University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

Health Application for Natural Products Information System for Plants (HANAPIN-SP) 2.0: Using Ontologies for Knowledge Representation and Data Storage

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Lim, Chelcy Jill Espiritu
2006-06145

October, 2011

ACCEPTANCE SHEET

The special problem entitled "*Health Application for Natural Products Information System for Plants (HANAPIN-SP) 2.0: Using Ontologies for Knowledge Representation and Data Storage*" prepared and submitted by *Chelcy Jill E. Lim* in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

_____
Richard Bryann L. Chua, M.Sc.
Adviser

EXAMINERS	APPROVED	DISAPPROVED
1. Gregorio B. Baes, Ph.D. (candidate)	_____	_____
2. Avegail D. Carpio, M.Sc.	_____	_____
3. Aldrich Colin K. Co., M.Sc. (candidate)	_____	_____
4. Vincent Peter C. Magboo, M.D., M.Sc.	_____	_____
5. Ma. Sheila A. Magboo, M.Sc.	_____	_____
6. Geoffrey A. Solano, M.Sc.	_____	_____
7. Bernie B. Terrado, M.Sc. (candidate)	_____	_____

_____
Date

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science

_____
Geoffrey A. Solano, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

_____
Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences and Mathematics

_____
Reynaldo H. Imperial, Ph.D.
Dean
College of Arts and Sciences

Abstract

Health Application for Natural Products Information System for Plants (HANAPIN-SP) 1.0 provides a central repository for storing, sharing and searching projects on natural products, but the limitations of relational databases, which is the storage schema of HANAPIN-SP 1.0, are lack of flexibility and lack of adaptability. An ontology-driven information system is an information system that uses an ontology to store both the data structure and the data itself, and addresses the limitations of relational databases. Thus, HANAPIN-SP 2.0 changed its storage from relational database-based to ontology-based. Plant natural products were also added to the existing Natural Products Ontology by Batista-Navarro, Manansala, Mendoza, and Ananiadou.

Contents

1 Introduction

1.1 Background of the Study

A natural product is a chemical compound or a substance produced by living organisms. Natural products are often used in the treatment of life-threatening conditions, such as cancer or malaria. The study of natural products in the Philippines is an active field, due to the archipelago's abundance and variety of plants and organisms [1].

Consolidation and archiving of studies on natural products is an ongoing and problematic task for researchers. The structures of the data on the study and experimentation of natural products are often varying from researcher to researcher. More importantly, there is no centralized repository for such studies. Studying about existing research usually involves personally and individually contacting the respective researcher/s, instead of simply searching a centralized database, resulting to more time and effort costs. Research work is often duplicated because researchers do not know that the research has already been carried out.

In answer to the stated problems, Health Application for Natural Products Information System for Plants (HANAPIN-SP) 1.0 by Custodio provides a centralized repository and interactive system for storing, sharing and searching projects on natural products [1]. It uses a simple single-class ontology that allows for flexible plant fields in the system.

1.2 Statement of the Problem

While HANAPIN-SP 1.0 has addressed the central repository problem, the natural limitation of the architecture used—relational databases—is lack of flexibility and adaptability. Changes to the structure of one of the database tables will propagate to the view component, because the structure of the data is hardcoded in the system. Also, although HANAPIN-SP 1.0 has implemented an ontology for dictating data structure of natural products, the ontology is a simple one-class type; there

is no hierarchy of relationships, perhaps due to lack of available data. This results to a cluttered ontology, which defeats the purpose of building an ontology.

An ontology-driven information system—an example of which is BioPortal [2]—uses ontologies both to dictate the structure of the data and store the data itself. The advantages of using such architecture are the flexibility and the adaptability. A change in the structure of the data in the storage layer will be reflected in the whole system with minimal recoding needed. The use of ontologies also allows the system to reuse existing ontologies, such as that created by Batista-Navarro et al. [3].

In an interview with Dr. Maria Constancia Carillo [4], she emphasized the need for having a repository for available research results to reduce the repetition of researches. She also mentioned the data fields that are important for this kind of repository, and who the intended users are.

1.3 Objectives of the Study

1. Extend the Natural Products Ontology by Batista-Navarro et al. [3] to include plant natural products or phytochemicals, with the following root classes as defined in the existing ontology.
    (a) Natural Products (Phytochemicals) – contains terms for classifying the compounds of natural products according to their chemical structure.
    (b) Organisms (Species) – contains terms for describing the organism which is the source of a natural product, basically a definition of the taxonomy of pertinent sources.
2. Implement an information system, called HANAPIN-SP 2.0, using the Natural Products Ontology created in objective 1 to store both data structure and content. The system will:
    (a) Allow unregistered users to:
        i. Browse, search, and view the species in the ontology.

ii. Browse, search, and view the phytochemicals in the ontology.

(b) Allow the curator to:

i. Perform the tasks unregistered users can do.

ii. Add, edit, and delete species, including details such as local names, phytochemicals found, traditional knowledge, locations, and images.

iii. Add and edit taxonomic categories of species.

iv. Add, edit, and delete phytochemicals, including details such as unique identifiers, formula, molecular weight, IUPAC (International Union of Pure and Applied Chemistry) name, and synonym.

v. Add and edit categories of phytochemicals.

vi. Import and export the ontology of the system.

vii. Edit account details.

(c) Allow the administrator to:

i. Perform the tasks unregistered users can do.

ii. Add, edit, and delete users.

iii. Edit account details.

1.4 Significance of the Project

The resulting Natural Products Ontology from this project contributes greatly to the natural products community of researchers, even if used independently, i.e., without HANAPIN-SP 2.0. After all, this is the first Philippine Plant Natural Products Ontology. Such an ontology can help in classifying natural products and aid in human and machine search. The Natural Products Ontology can actually be reused in another information system. It can also be continuously improved as more data become available.

Another significance of this project is that it adds flexibility and adaptability to the original HANAPIN-SP. As already proven by the first ontology-driven in-

formation system developed, changing the schema or adding a new feature did not entail as many recodings as a database-driven system would need [2].

HANAPIN-SP 2.0 is one of the few Philippine ontology-based information systems. As the Semantic Web (a "web of data" that enables machines to understand the semantics, or meaning, of information in the World Wide Web, of which an ontology is one of the important concepts) continues to grow, ontology-based information systems such as this can promote this development.

1.5 Scope and Limitations

The Plant Natural Products Ontology is built with the help of Dr. Maria Constancia Carillo, an expert on this field, and the accuracy and completeness is based on Dr. Carillo's input. It includes the ontology of plant natural products arranged by structural type, and the classes of organisms that produce such natural products.

Since storing data directly in the ontology is a new approach—there is only one existing large-scale information system that uses ontology as data storage [2]—the study of the running times of insertion, deletion, and searching of terms in the ontology is not yet extensively studied. Using an ontology, the running times may not be as fast as using relational databases. The running times are not studied in this project.

There is no user registration in the system. To become a curator, an unregistered user must contact the administrator of the site and request curator privileges. All curators have equal privileges in adding and editing data.

HANAPIN-SP 2.0 is designed to be used by one team of researchers or collaborating teams of researchers, that is, all users agree to share their information to all potential users of the system.

2 Review of Related Literature

2.1 HANAPIN-SP 1.0

HANAPIN-SP 1.0 by Custodio [1] features the following fully-working modules: Registered User Services, References, Active Compounds, and Plant Source. The Registered User Services module allows users to register and login, and access specific data open to their access level. Administrators are able to do system management. The References module allows users to view or download references, and higher level users to edit references for projects. The Active Compounds module allows users to view and edit a project's associated active compounds. This module uses an ontology to manage the active compound data structure, thus allowing the flexible changing of data fields as necessary. The Plant Source module allows users to view and edit plant source information of a project.

The problem with HANAPIN-SP 1.0 is that although it uses an ontology, it is a single-class ontology. The concept of an ontology was not included in the development of HANAPIN-SP 1.0, so there was little ontology development and only one XML file, describing the attributes of an active compound (thus we presume the class is named `Compound`), serves as the ontology. Classifying the compounds or natural products according to their structural type would result to a more organized and easy-to-search ontology. Also, there is no ontology for the plant source. Moreover, the ontology usage in the system is very minimal, used only to describe the compounds, not to store them—the data is stored in a relational database. As discussed in detail in Section 2.5, using an ontology for the internal structure and data of the system gives flexibility and adaptability benefits, which is not offered by how the ontology is minimally used in HANAPIN-SP 1.0.

2.2 Ontologies

Due to the ability of an ontology to explicitly describe data semantics in a common way, and sharing these among heterogeneous information systems, development

of ontologies for different domains have been pursued. For example, Saad et al. [5] have developed an ontology in the field of Islamic knowledge, which is used to answer questions regarding definition of concepts, acceptable rules for worship, and anything relating to Islamic knowledge.

Brusa et al. [6] have developed an ontology for a budgetary and financial system of the government. They have also tried the approach of adapting phases from different ontology engineering methodologies and succeeded with building an ontology by following a custom methodology fitted to their needs. This approach will be imitated for the further development of the Natural Products Ontology, adapting phases and guidelines from different methodologies that are suited to the ontology's specifications.

Because many ontologies are being developed, ontology integration is a growing area of research. Ontology integration is especially imporant between different ontologies of the same domain. Different ontologies may structure data differently, or even contain conflicting data. Hooijmaijers and Stumptner [7] proposed a method of ontology integration based on trust in the authors of the resources. Orgun et al. [8] list numerous approaches for providing interoperability among heterogeneous domain ontologies, by ontology merging, ontology mapping, or ontology alignment. They highlight the importance yet absence of a domain-independent fully automated framework for semantic interoperability.

The mentioned papers show that the development and use of ontologies can be applied to many domains, and that further research on ontology integration is an active field, indicating the need and benefit of sharing domain knowledge that will be useful for the domain community.

2.3 Natural Products Ontology

The Natural Products Ontology by Batista-Navarro et al. [3] is an ontology that includes the following major classes: Natural Product, Biological Activity (biological roles associated with natural products), Organism, Ecological Condition

(which affect the production of natural products by its source origins), Disease (with which the biological activities are associated), and Drug Target (names of the proteins linked to natural products). (See Figure 1.)

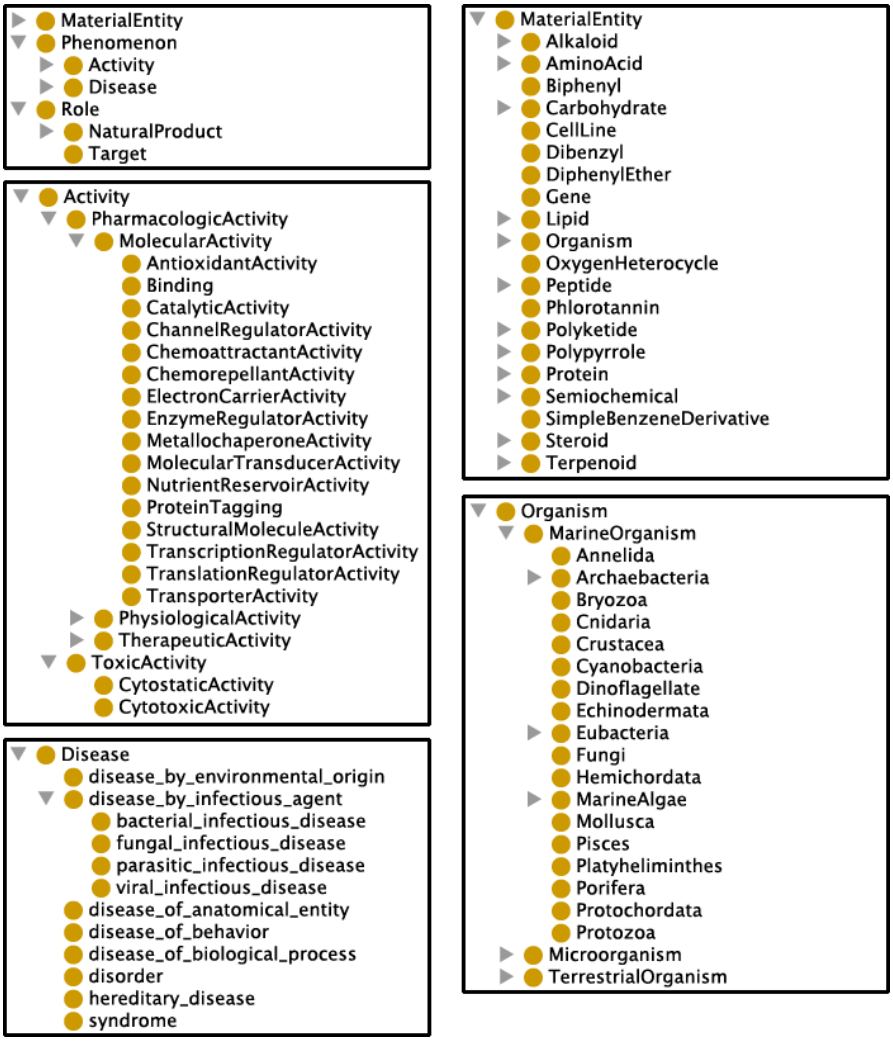

Figure 1: Some classes in the Natural Products Ontology

2.4 Ontology Engineering Methodologies

There exist several ontology engineering methodologies, all with their own processes, applications, and advantages and disadvantages. Several studies [9, 10, 11] have also been carried out to compare the capabilities of the more popular engineering methodologies.

The Cyc method has been used to build the Cyc KB, an ontology and knowl-

edge base of common sense knowledge [10]. It is an iterative process of studying and verifying knowledge before adding it into the ontology. Its processes eventually rely on machine learning tools to search for new common sense knowledge in the ontology [11].

Uschold and King's method is business-oriented and focuses on the end-application—how the ontology will be used in what kind of application [11]. Although it provides an outline of its four processes, Fernández-López in [9] notes that it does not detail the techniques, methods, and principles for each process, and that it lacks pre-development and post-development processes. Gómez-Pérez et al. in [11] further observed that it lacks a conceptualization process, which would enable domain experts to understand the ontology without having to learn any specific ontology-encoding language.

Grüninger and Fox's methodology is based on the experience of developing the Toronto Virtual Enterprise (TOVE) project ontology, which is also in the business domain [9]. One feature of this methodology is the encoding of knowledge in a formal logical language. This can prove to be both an advantage and a disadvantage. While the high degree of formality assures that the ontology is written in a universally understood standard, it requires the encoder and the user to learn and understand first-order logic to comprehend anything from the ontology. The predicates and axioms are quite complicated [11]. Considering that the ontology to be built should be easily understood by domain experts—who are, in this case, biology experts—using first-order logic will be unnecessary.

The KACTUS project's objectives were to investigate the feasibility of knowledge reuse in complex technical systems and the role of ontologies to support it [9]. The KACTUS method begins with the process of specifying the application—again, this method is application-based. Three ontologies have been developed to support the feasibility of this method; these have proven that the ontologies were reusable with each other as the application grew more complex [11]. However, this can be hardly considered this specific method's advantage, because it is an

ontology's basic nature to be reusable. The KACTUS method, as many methods, has no engineering life cycle. It also has not been used to build many ontologies and applications.

METHONTOLOGY is perhaps the most complete ontology engineering methodology—it enumerates the processes and the tasks for each process, and includes a project life cycle [11]. One more feature of this method is that it does not build its ontology around a specific application; it enables the construction of methodologies at the knowledge level [9]. Its processes, in some way, mirror the software development process and knowledge engineering methodologies, so it has the advantage of being complete from pre-development to post-development [11]. Moreover, it is recommended by the Foundation for Intelligent Physical Agents (FIPA) and is supported by many tools [10].

Perhaps the SENSUS method is the most unique of all the methodologies, because it promotes knowledge shareability by using a single base ontology to develop ontologies in specific domains [11]. This means linking important terms to the SENSUS ontology; all concepts are linked in a path from itself to the root of SENSUS. This would imply that any ontology developed using the SENSUS method will be immediately compatible with other SENSUS-based ontologies, requiring no further editing. But it is actually unknown how many developers link their ontologies to the SENSUS root. Also, the method is largely business-oriented and has no life cycle [9].

Lastly, the On-To-Knowledge methodology identifies more processes than the rest of the approaches [10]. It also details the techniques and principles for each of its processes, and indicates the relationships between such processes [11]. Like METHONTOLOGY, it also has a set of supporting tools.

Some researchers have proposed methodologies on accomplishing specific processes of the ontology engineering methodology. For example, some well-known ontology evaluation methodologies are OntoClean, OntoMetric, and OntoAbsolute [12]. OntoAbsolute aims to evaluate ontologies based on criteria such as simplicity,

unity, singleness, simpleness, connectivity, integration, and relativeness. Another evaluation method by Ouyang and Qu [13] evaluates ontologies based on coverage, cohesion, and coupling. The number of proposed ontology evaluation methods is growing due to the fact that the number of available ontologies is also growing. The goal of ontology evaluation is to select a suitable ontology for a particular application with respect to the user requirements in a given context [13]. This is especially important in domains where many ontologies have been developed.

Noy and McGuinness in [14] note that there is no one correct way of building an ontology. One reason is that the methodologies discussed have usually been developed for building ontologies in a specific domain or for a specific purpose (except perhaps a few, like METHONTOLOGY and the SENSUS method). Because none of the approaches cover all the processes involved in ontology building, Fernández-López in [9] suggests taking this series of methodologies as a reference point and develop several methodologies adaptable to different ontology types in different settings. This approach has been taken to develop an ontology in the budgetary and financial domain [6].

2.5 Information Systems Using Ontologies

Here we consider some information systems where ontologies are extensively used.

The BioExtract Server [15] is a publicly accessible Web-based data integration application of biological sequence databases. The project needed an ontology because different databases had different data sets, fields, and formatting. What was needed was a querying tool that could query any number of the supported biological sequence databases simultaneously. This would have been a problem because databases are heterogeneous: what may be called a `primaryKey` in one database may be called `primaryID` in another. And this is a simple example; in the field of biological sequences, many terms can be synonyms of each other. Therefore, they used an ontology to map the different databases to a universal name defined in a global ontology in the system, which would allow a single query

to correctly query all specified databases. The results are then returned in a unified format so analysis can be immediately carried out.

The BioExtract Server project has been a success—it allows researchers to simultaneously query different sequence databases to yield integrated result sets and apply analytic tools without requiring programming expertise.

Jia et al. [16] have designed an ontology-based cross-domain privilege management system, which solves security problems by preventing unauthorized clients from accessing certain Web services. Through the use of privilege management ontologies, cross-domain service requests are mapped from one domain to another to determine the privilege of a certain user. The system was implemented using OWL for ontologies and Jena for parsing and loading the ontologies.

BioPortal [2]—a database of biomedical ontologies—takes using ontologies to a system-wide level: it uses ontologies and ontology instances to represent essentially all data that the application requires, from declarative high-level descriptions to internal system data. First, the developers replaced the databases in the storage layer with an ontology. Then they replaced the database calls with ontology calls, and finally moved the data from the databases to instances in the ontology.

They also validated the flexibility and adaptability, which are the supposed benefits of an ontology-driven information system, by adding a new feature: views, which are basically specific subsets of bigger ontologies. According to observations and discussions with developers, implementing this new feature was easier and far more efficient in the ontology-based system than the table-based system. It took less work because they could reuse existing parts of the ontology—since a view is essentially an ontology itself, only with additional metadata, like which ontology it is from and what area in the domain it is trying to describe. Thus, we expect that we can get these same advantages from using an ontology for both structure and data.

2.6 From Relational Database Tables to Ontologies

As the Semantic Web becomes more popular, so does the use of ontologies over database tables. Thus there exist several simple techniques to convert relational database tables to an ontology [17, 18, 19].

RDB2Onto [17] and D2OMapper [19] are both automated tools that take care of mapping database concepts to ontology concepts. RDB2Onto [17] has a simple algorithm of transforming an SQL query to an OWL template. It requires, of course, building the SQL query first. The output is the code for the ontology in XML format. D2OMapper [19] is based on the complex definitions of a database schema and an OWL ontology as tuples. Once the database schema has been defined correctly, the tool can then present its mapping as an OWL ontology.

Finally, Astrova and Stantic [18] propose a method of switching from relational databases to ontologies by analyzing HTML forms. This is a rather feasible solution, since most interaction with the database (adding and editing data) deals with HTML forms. The three basic steps are:

1. Analyzing HTML forms to extract a form model schema that expresses semantics of a relational database behind the forms;

2. Transforming the form model schema into an ontology ("schema transformation"); and

3. Creating ontological instances from data contained in the forms ("data migration").

3 Theoretical Framework

3.1 Plant Natural Products

In this project, plant natural products or plant phytochemicals refer to natural products that are harvested from terrestrial plants, as opposed to, for example, natural products harvested from marine organisms (which are called marine natural products [3]). Whenever the term "plant natural products" is used in this project, it will always refer to the natural products from terrestrial plants, the addition of which to the ontology is one of the main objectives of this project.

Plant natural products provide the basis for drug discovery and the inspiration for creating synthetic drugs, providing cures from simple illnesses to life-threatening diseases. Some plant extracts have been used traditionally but the exact compound that addresses the targeted illness is not directly known. Therefore natural products research is an active research field in the Philippines, especially given the abundance and variety of organisms in the country [20].

3.2 Ontology

An ontology is a formal, explicit specification of concepts in a domain of discourse; the concepts are usually arranged in a hierarchy [21, 14]. These concepts are sometimes called *classes*; properties of each concept that describe various features and attributes of the concept are called *slots*, *attributes*, *roles*, or *properties*; and restrictions on slots are called *facets* or *role restrictions* [14]. Ontologies are reusable and sharable artifacts that have to be developed in a machine interpretable language [21]. An ontology together with a set of individual instances of classes constitutes a knowledge base, but the difference between an ontology and a knowledge base is hardly discernible in reality [14].

An ontology can be about any topic—from wines to poker to general common sense. For example, Cyc[1] is an everyday common sense knowledge base. The

[1] http://www.cyc.com/

Gene Ontology (GO)[2] is a major bioinformatics initiative with the aim of standardizing the representation of gene and gene product attributes across species and databases. BioPortal[3] lists over 250 biomedical ontologies.

Several reasons to develop an ontology are to share common understanding of the structure of information among people or software agents, to enable reuse of domain knowledge, to make domain assumptions explicit, to separate domain knowledge from the operational knowledge, and to analyze domain knowledge [14]. Not to mention there is also the growing trend of the Semantic Web and thus the use of ontology to store information system data.

3.3 Ontology Engineering Methodologies

To come up with a final methodology that is used to develop the Plant Natural Products Ontology, we discuss here in detail the relevant parts of some methodologies.

The On-To-Knowledge methodology takes into account how the ontology will be used in further applications [9]. It outlines the following five processes [11]:

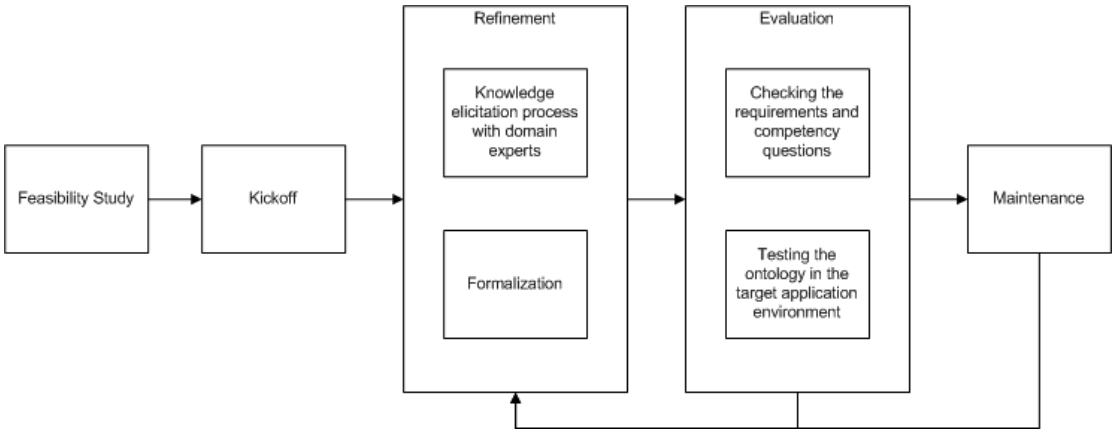

Figure 2: On-To-Knowledge Methodology Processes

1. Feasibility Study – applied to the complete application and, therefore, should be carried out before developing the ontologies.

[2]http://www.geneontology.org/
[3]http://bioportal.bioontology.org/

2. Kickoff – results in the ontology requirements specification document that describes the following: the domain and goal of the ontology; the design guidelines (for instance, naming conventions); available knowledge sources (books, magazines, interviews, etc.); potential users and use cases as well as applications supported by the ontology. Competency questions can be useful to elaborate the requirements specification document. The first draft of the ontology is called the "baseline ontology". The most important concepts and relations are identified on an informal level. The developers should look for potentially reusable ontologies already developed.

3. Refinement – produce a mature and application-oriented "target ontology" according to the specification given in the kickoff process. Includes the following activities:

   (a) Knowledge elicitation process with domain experts – the baseline ontology is refined by means of interaction with experts in the domain. Axioms are identified and modeled. During the elicitation, the concepts are gathered on one side and the terms to label the concepts on the other. Then, terms and concepts are mapped. A complementary way to enrich the ontology is to use it as seed in an ontology learning process.

   (b) Formalization – the ontology is implemented using an ontology language. Such language is selected according to the specific requirements of the envisaged application. On-To-Knowledge recommends the OntoEdit ontology editor, which generates automatically the ontology code in several languages.

4. Evaluation – proof of the usefulness of the developed ontologies and their associated software environment. The product obtained is called ontology based application. Includes the following activities:

   (a) Checking the requirements and competency questions – the developers

check whether the ontology satisfies the requirements and "can answer" the competency questions.

(b) Testing the ontology in the target application environment – this evaluation process is closely linked to the refinement process. In fact, several cycles are needed until the target ontology reaches the envisaged level.

5. Maintenance – refers to both ontology and system software maintenance. The ontology and the system software are corrected and updated as needed.

METHONTOLOGY has supporting (pre-development and post-development) processes that are absent from other methodologies. The processes are as follows [11]:

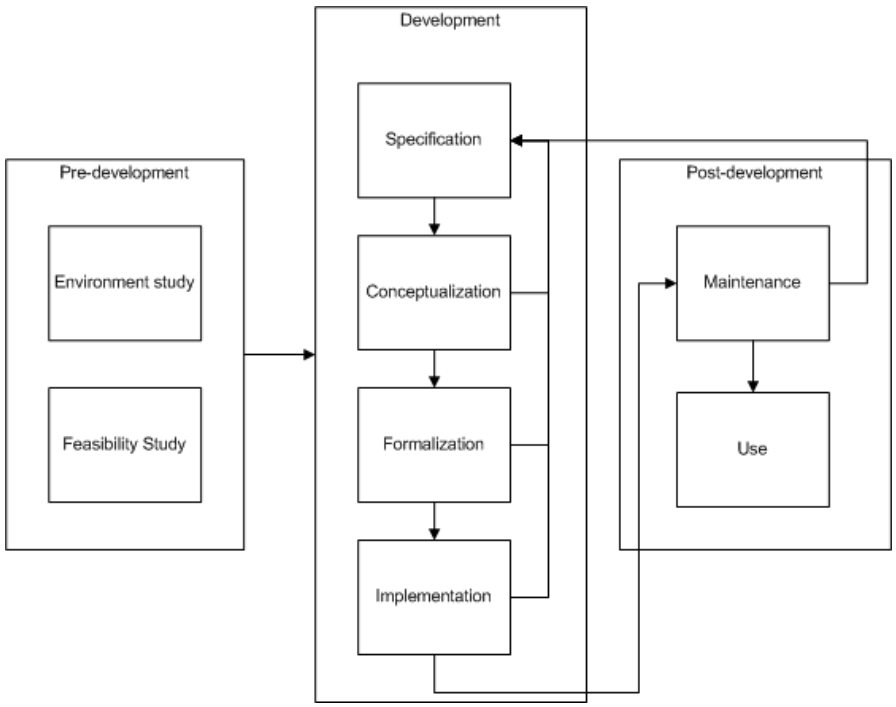

Figure 3: METHONTOLOGY Processes

- Pre-development

1. Environment study – identifies the problem to be solved with the ontology, the applications where the ontology will be integrated, and so forth.

2. Feasibility study – answers questions like: is it possible to build the ontology?; is it suitable to build the ontology?; and so forth.

- Development

    1. Specification – states why the ontology is being built, what its intended uses are and who the end-users are.
    2. Conceptualization – structures the domain knowledge as meaningful models either from scratch or reusing existing models; is implementation-language independent.
    3. Formalization – transforms the conceptual model into a formal or semi-computable model.
    4. Implementation – builds computable models in an ontology language.

- Post-development

    1. Maintenance – updates and corrects the ontology if needed.
    2. Use – refers to being used by other ontologies or applications.

METHONTOLOGY also provides specific guidelines, especially for the conceptualization process of development. The effectiveness can be positively deduced from the large number of ontologies that have been built using this methodology [11].

3.4 OWL

The OWL Web Ontology Language is a W3C Recommendation designed for use by applications that need to process the content of information instead of just presenting information to humans [22]. OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. This representation of terms and their interrelationships is called an ontology. Basically, OWL is one of the languages available to represent an ontology.

OWL is built on top of RDF (Resource Description File), a way for describing Web resources, and is written in XML (Extensible Markup Language) format so that it can be easily shared between different types of computers and application languages. RDF can be queried using SPARQL (SPARQL Protocol and RDF Query Language).

```
<rdfs:Class rdf:ID="WINE">
   <rdfs:subClassOf rdf:resource="#POTABLE-LIQUID"/>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#MAKER"/>
         <daml:minCardinality>
            1
         </daml:minCardinality>
      </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#MAKER"/>
         <daml:toClass rdf:resource="#WINERY"/>
      </daml:Restriction>
   </rdfs:subClassOf>
   <rdfs:subClassOf>
      <daml:Restriction>
         <daml:onProperty rdf:resource="#GRAPE-SLOT"/>
         <daml:minCardinality>
            1
   .........
</rdfs:Class>
```

Figure 4: An example OWL file. This is an excerpt from the wine ontology, defining the concept of a wine, taken from the freely accessible Wine Agent (http://onto.stanford.edu:8080/wino/).

OWL provides three sublanguages:

- OWL Lite supports classification hierarchy and simple constraints.
- OWL DL supports maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time).
- OWL Full supports maximum expressiveness and the syntactic freedom of RDF with no computational guarantees.

From an interview with Velez [23], Spivack advised that for people starting on the Semantic Web, OWL Lite is enough. According to him, reasoning and the other features of OWL DL/Full are complicated and not being used for anything yet. OWL Lite is a good starting point as it is simple, and should a developer wish to upgrade, a legal OWL Lite ontology is a legal OWL DL and OWL Full ontology [22].

3.5 Protégé

Protégé[4] is a free, open-source, Java-based ontology editor. It can export its ontologies into formats applicable for use in Web development. Users can easily extend Protégé with its plug-in architecture. Protégé actually uses relational databases to store its ontology, but the structure is hidden from the user; we do know it is ontology-independent. A certain flavor of Protégé, called Protégé-OWL, allows users to load and save OWL files and execute reasoners.

Protégé has a core API, which is used to access basic Protégé functionalities. The Protégé API can be used directly by external applications to access Protégé knowledge bases. It allows applications to query and edit ontologies, without requiring the Protégé program to be actively running. The API seems to be limited; according to the FAQ section of the Protégé Website, querying is currently a simple matching. An alternative would be to save the Protégé ontology as an OWL ontology, which can then be queried in a different way, such as by using Ruby on Rails[5] with the ActiveRDF[6] plug-in, or Jena [23].

3.6 Jena

Jena[7] is a Java framework that provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. It is used

[4]http://protege.stanford.edu/
[5]http://rubyonrails.org/
[6]http://activerdf.org/
[7]http://openjena.org/

to read, write, and query RDF statements. Since OWL is built on top of RDF, Jena can be also used to programatically access OWL ontologies.

3.7 Ontology-Driven Information System

An ontology-driven information system is an information system that uses an ontology to store both the data structure and the data itself. This is in contrast to most information systems which use relational databases (the most popular being MySQL[8]) to store data. In using relational databases, the structure of the data is hardcoded into the table structure. This means that changing the structure of the data, which happens when the data structure is uncertain or when adding new features, will require many changes along the development stack. Using ontologies, however, change is limited to the ontology and the view layer, if needed.

For example, if adding a new feature would require adding new database tables, this would usually mean changing the structure of other related tables to reflect foreign links between the tables. And as web applications grow in the number of layers and tiers they possess, changes will need to be done in almost every layer and tier. We avoid this problem by using an ontology—changes need to be done only on the ontology itself and the view layer.

The first large-scale ontology-driven information system, BioPortal [2], has been successful in proving the feasibility and the extensibility of the infrastructure. Though this kind of infrastructure is novel, it is growing quickly, as seen from the increasing number of Semantic Web applications [23] and methods to convert relational databases to ontologies [17, 18, 19].

[8] http://www.mysql.com/

4 Design and Implementation

4.1 Plant Natural Products Ontology

To develop the Plant Natural Products Ontology, the ontology by Batista-Navarro et al. [3] is used. The ontology is not yet complete, but the resulting ontology is comprised of the following major classes:

- Natural Product
- Biological Activity
- Organism
- Ecological Condition
- Disease
- Drug Target

These classes are currently under development. The current subclasses of the Organism class are `alga` and `prokaryote`.

The Plant Natural Products Ontology is part of this already existing ontology. All the classes except Natural Product and Organism do not need much updating; the biological activities, ecological conditions, and other corresponding concepts for plant and marine natural products are roughly the same. Only a few changes and updates to these classes are needed. The main concern of this project is to add terrestrial plants to the Organism class, and the natural products or phytochemicals from these plants to the Natural Product class. Subclasses are added under Organism to refer to terrestrial plants.

4.2 Proposed Ontology Engineering Methodology

This proposed ontology engineering methodology follows the processes of the On-To-Knowledge methodology, uses the guidelines from [14], and borrows some activities from METHONTOLOGY for the pre-development and post-development

processes. The On-To-Knowledge methodology is used because it takes into account the application to be developed, thus resulting to an application-dependent ontology [9]. This ensures that the ontology works with the already existing HANAPIN-SP 1.0 (soon to be 2.0). Noy and McGuinness [14] discuss general issues and solutions, describes an iterative approach, and provides more specific guidelines for each of the processes (except for the post-development process).

1. Feasibility Study – also the pre-development process. It identifies the problem to be solved with the ontology, the applications where the ontology will be integrated, and so forth. It also answers questions like: is it possible to build the ontology?; is it suitable to build the ontology?; and so forth [11]. The following guide questions are from [14]:

    - What is the domain that the ontology will cover?
    - For what we are going to use the ontology?
    - For what types of questions the information in the ontology should provide answers?
    - Who will use and maintain the ontology?

2. Kickoff – will give the following output [11]:

    - Ontology requirements specification document that describes the domain and goal of the ontology, the design guidelines (for instance, naming conventions), available knowledge sources (books, magazines, interviews, etc.), and potential users and use cases as well as applications supported by the ontology.
    - Competency questions, which can be useful to elaborate the requirements specification document.
    - The baseline ontology or the first draft of the ontology. The most important concepts and relations are identified on an informal level.

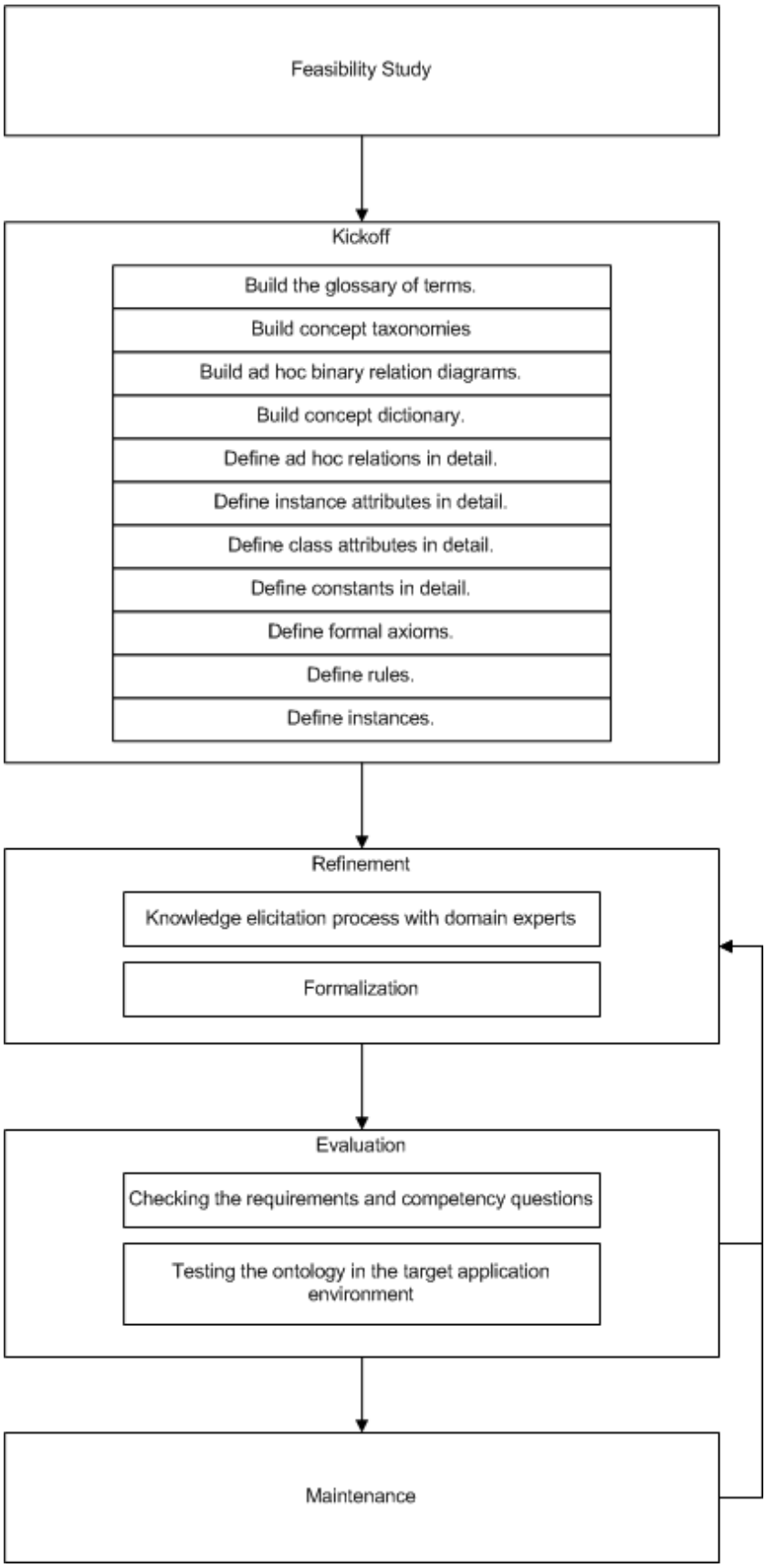

Figure 5: Proposed Ontology Engineering Methodology

The Kickoff process has the following tasks, borrowed from METHONTOLOGY [11]:

(a) Build the glossary of terms.

(b) Build concept taxonomies.

(c) Build ad hoc binary relation diagrams.

(d) Build concept dictionary.

(e) Define ad hoc binary relations in detail.

(f) Define instance attributes in detail.

(g) Define class attributes in detail.

(h) Define constants in detail.

(i) Define instances.

To aid in activity (b), the following taxonomic relations are defined in METHONTOLOGY [11]:

- A concept C1 is a *Subclass-Of* another concept C2 if and only if every instance of C1 is also an instance of C2.
- A *Disjoint-Decomposition* of a concept C is an infinite set of subclasses of C that do not have common instances. For example, in an ontology on the domain of traveling, the set of flight numbers is a disjoint-decomposition of a flight company; all flight numbers cannot possibly be listed, and there is no flight with two flight numbers.
- An *Exhaustive-Decomposition* of a concept C is a finite set of subclasses of C that may have common instances and subclasses. For example, the set of subclasses `Economy Trip`, `Business Trip`, and `Luxury Trip` is an exhaustive-decomposition of `Travel Package`; there is no other type of travel package, and a travel package may be both a business and luxury trip.

- A *Partition* of a concept C is a finite set of subclasses of C that do not have common instances. For example, the set of subclasses `International Flight` and `Domestic Flight` is a partition of `Flight`; there is no other type of flight, and a flight may either be international or domestic, but not both.

We also use the following class hierarchy guide from [14] as basis to build the hierarchy of classes which have no taxonomic hierarchy.

- A subclass of a class represents a concept that is a "kind of" the concept that the superclass represents.
- A single object is not a subclass of all objects. That is, there should be no distinction between the plural and singular versions of the same concept, with the singular version being a subclass of the plural version.
- Transitivity of the hierarchical relations: If B is a subclass of A and C is a subclass of B, then C is a subclass of A.
- Classes and their names: Classes represent concepts in the domain and not the words that denote these concepts. The name of a class may change if we choose a different terminology, but the term itself represents the objective reality in the world. Synonyms for the same concept do not represent different classes.
- Avoid class cycles.
- Siblings in a class hierarchy: All the siblings in the hierarchy (except for the ones at the root) must be at the same level of generality. If a class has only one direct subclass there may be a modeling problem or the ontology is not complete. If there are more than a dozen subclasses for a given class then additional intermediate categories may be necessary. However, if no natural classes exist to group concepts in the long list of siblings, there is no need to create artificial classes.

3. Refinement – includes the activities *Knowledge elicitation process with domain experts* and *Formalization*, as defined in Section 3.3. Noy and McGuinness [14] discuss when to introduce a new class or not:

    - Subclasses of a class usually (1) have additional properties that the superclassdoes not have, or (2) restrictions different from those of the superclass, or (3) participate in different relationships than the superclasses. In practical terms, each subclass should either have new attributes added to it, or have new attribute values defined, or override some details for the inherited attributes.
    - Classes in terminological hierarchies do not have to introduce new properties. Sometimes it may be useful to create new classes even if they do not introduce any new properties. For example, some ontologies include large reference hierarchies of common terms used in the domain.
    - If the concepts with different attribute values become restrictions for different attributes in other classes, then we should create a new class for the distinction. Otherwise, we represent the distinction in an attribute value.
    - If a distinction is important in the domain and we think of the objects with different values for the distinction as different kinds of objects, then we should create a new class for the distinction.
    - A class to which an individual instance belongs should not change often.
    - Individual instances are the most specific concepts represented in a knowledge base.
    - If concepts form a natural hierarchy, then we should represent them as classes.

4. Evaluation – includes the activities *Checking the requirements and competency questions* and *Testing the ontology in the target application environment*, as defined in Section 3.3.

5. Maintenance – also the post-development process. Here, updates and corrections are done to the ontology and the system software if needed [11].

4.3 HANAPIN-SP 2.0

4.3.1 Context Diagram

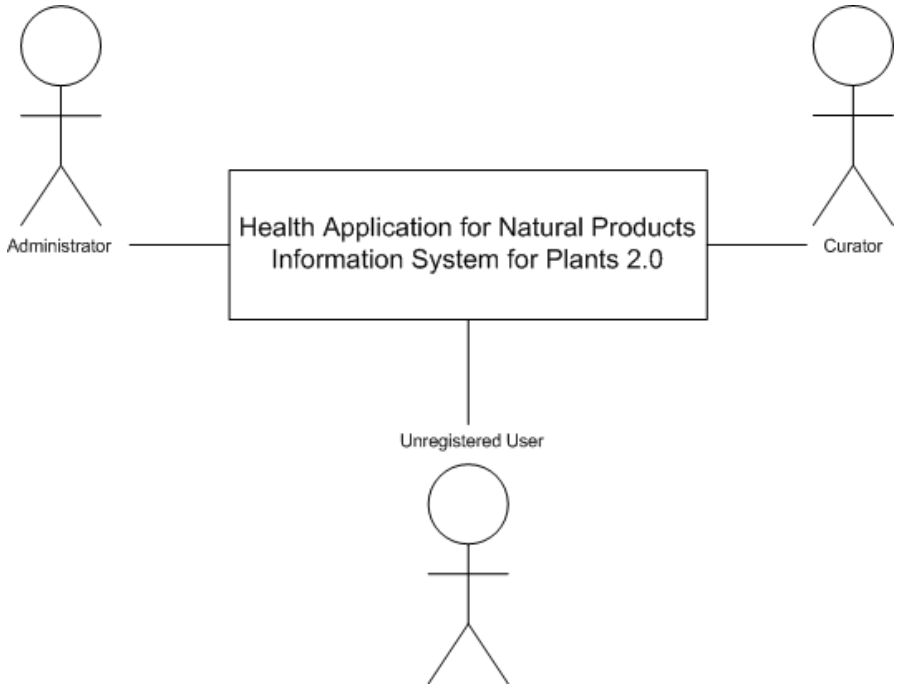

Figure 6: Context diagram of HANAPIN-SP 2.0

4.3.2 Use Case Diagram

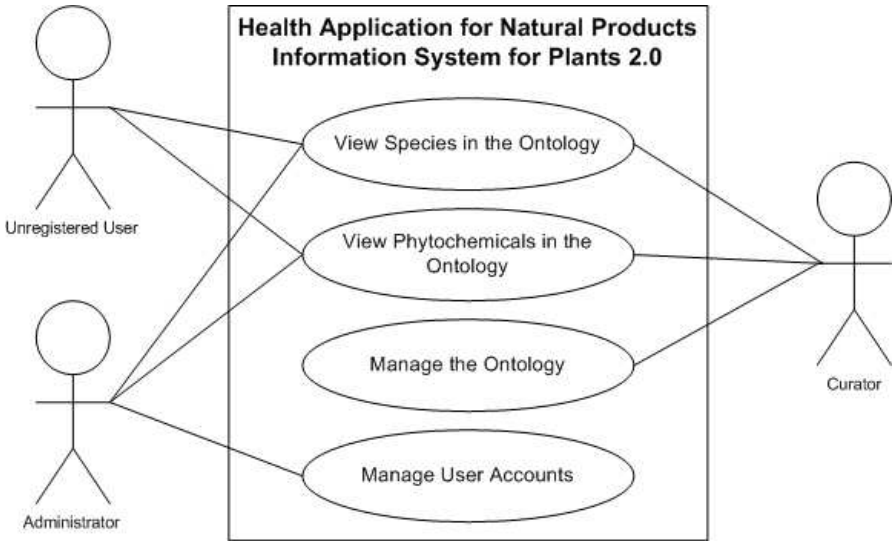

Figure 7: Top-level use case diagram of HANAPIN-SP 2.0

1. View Species in the Ontology

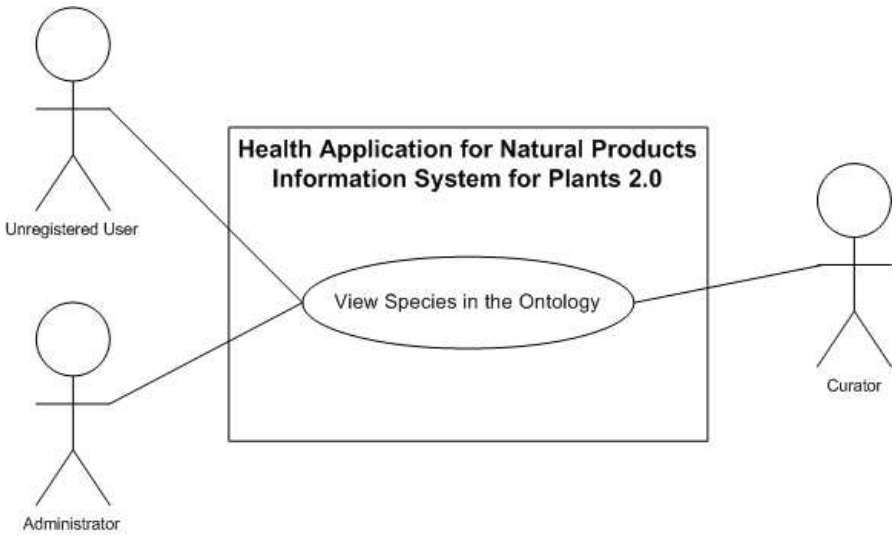

Figure 8: View Species in the Ontology use case diagram of HANAPIN-SP 2.0

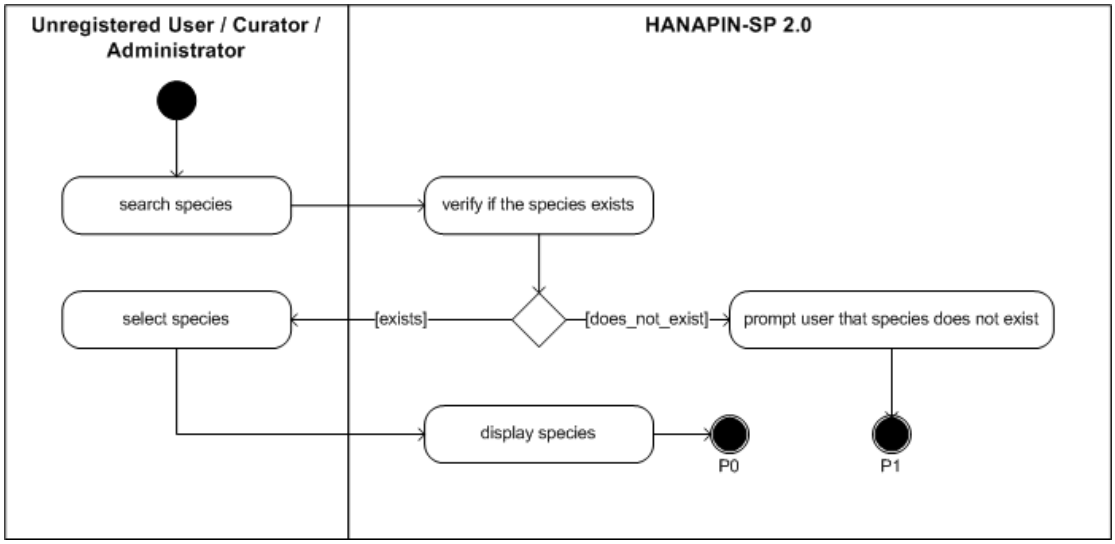

Figure 9: Search and view species activity diagram of HANAPIN-SP 2.0

2. View Phytochemicals in the Ontology

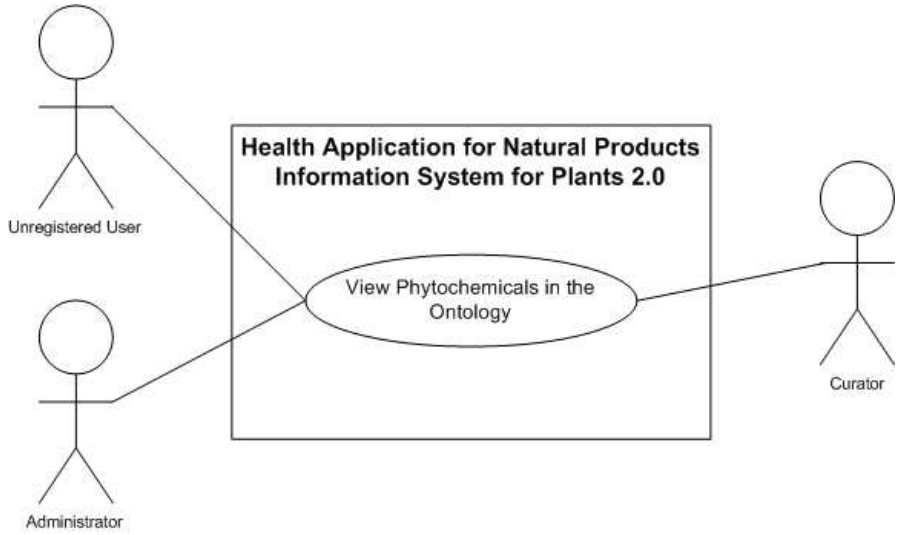

Figure 10: View Phytochemicals in the Ontology use case diagram of HANAPIN-SP 2.0

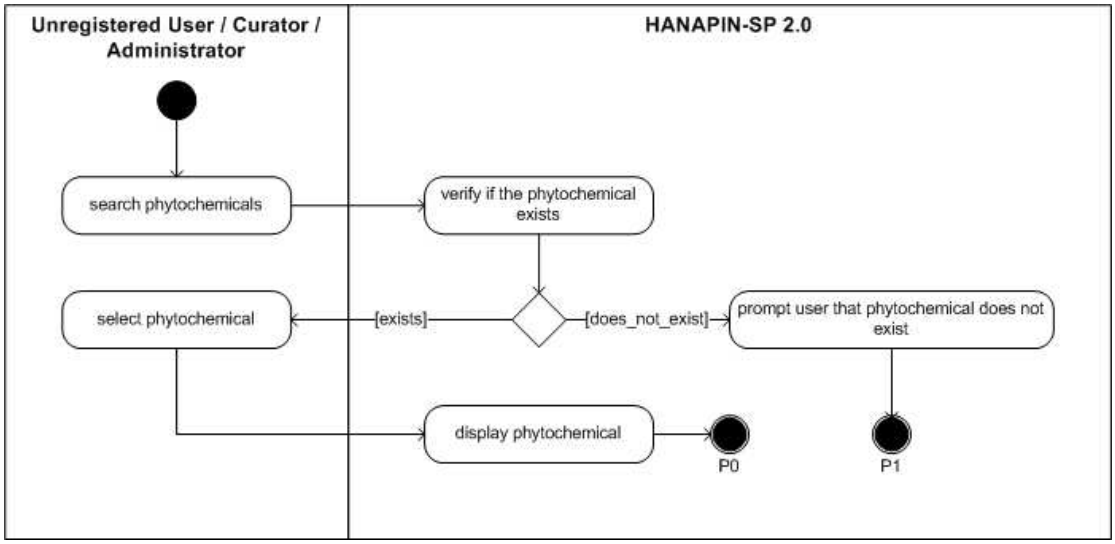

Figure 11: Search and view phytochemicals activity diagram of HANAPIN-SP 2.0

3. Manage the Ontology

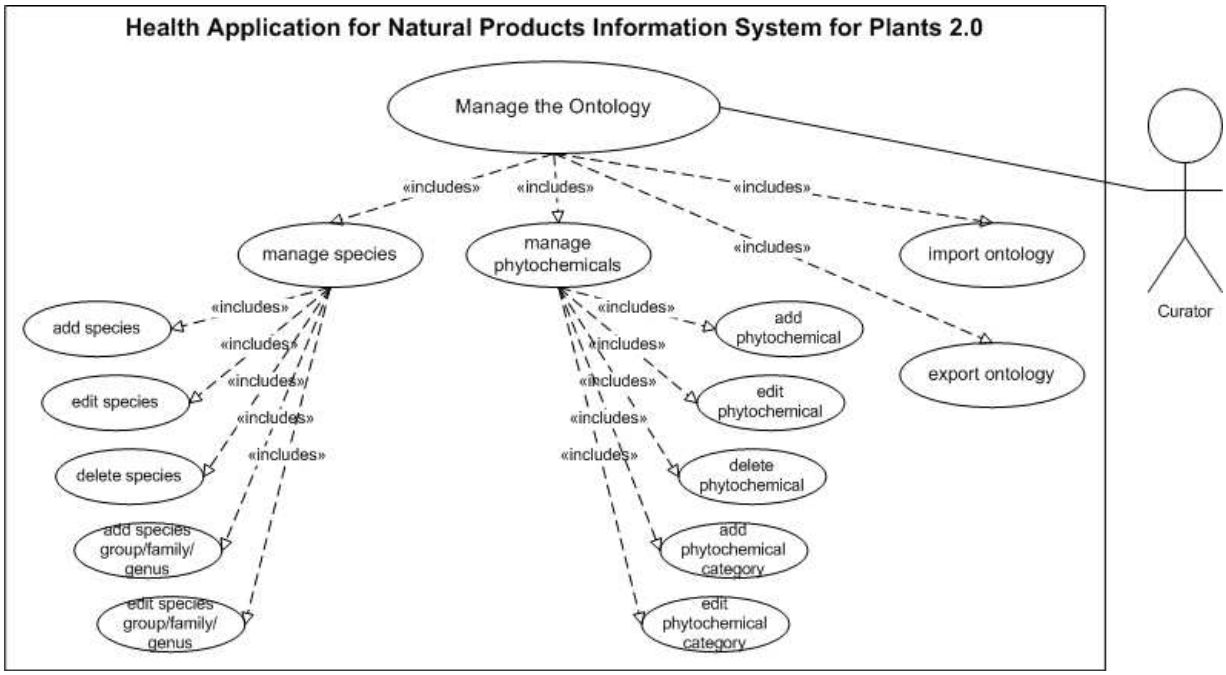

Figure 12: Manage the Ontology use case diagram of HANAPIN-SP 2.0

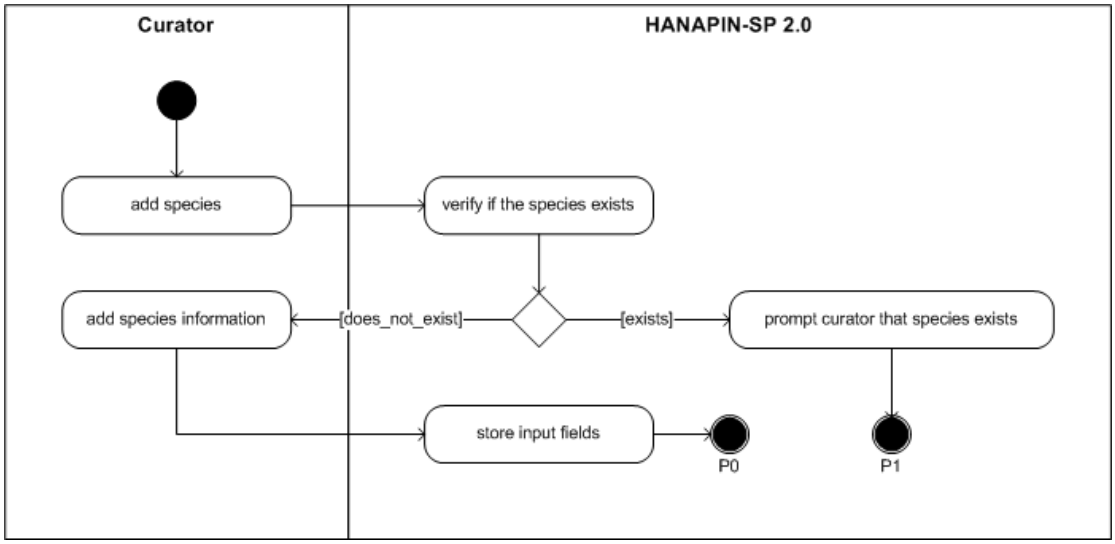

Figure 13: Add species activity diagram of HANAPIN-SP 2.0

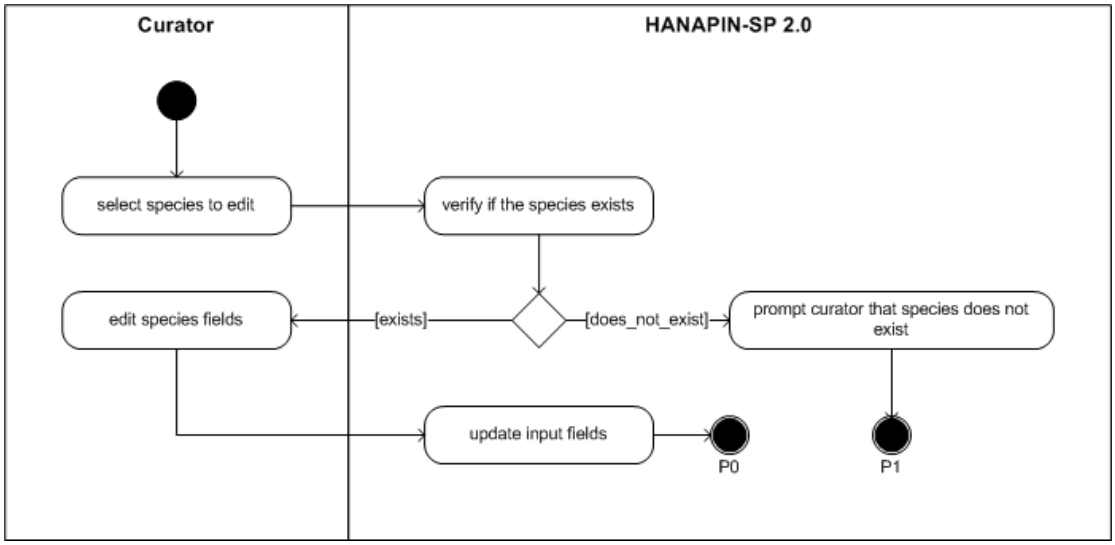

Figure 14: Edit species activity diagram of HANAPIN-SP 2.0

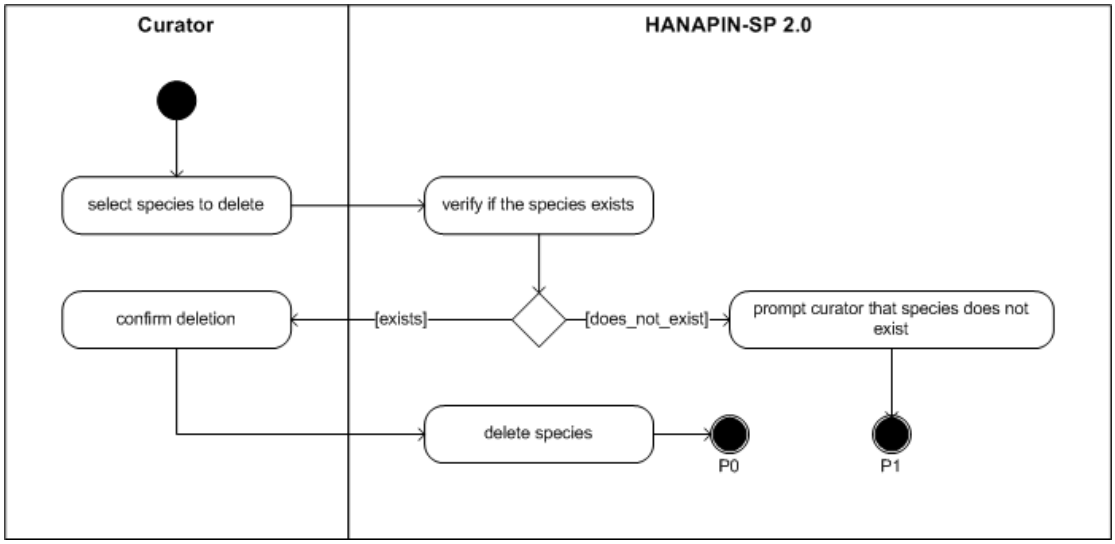

Figure 15: Delete species activity diagram of HANAPIN-SP 2.0

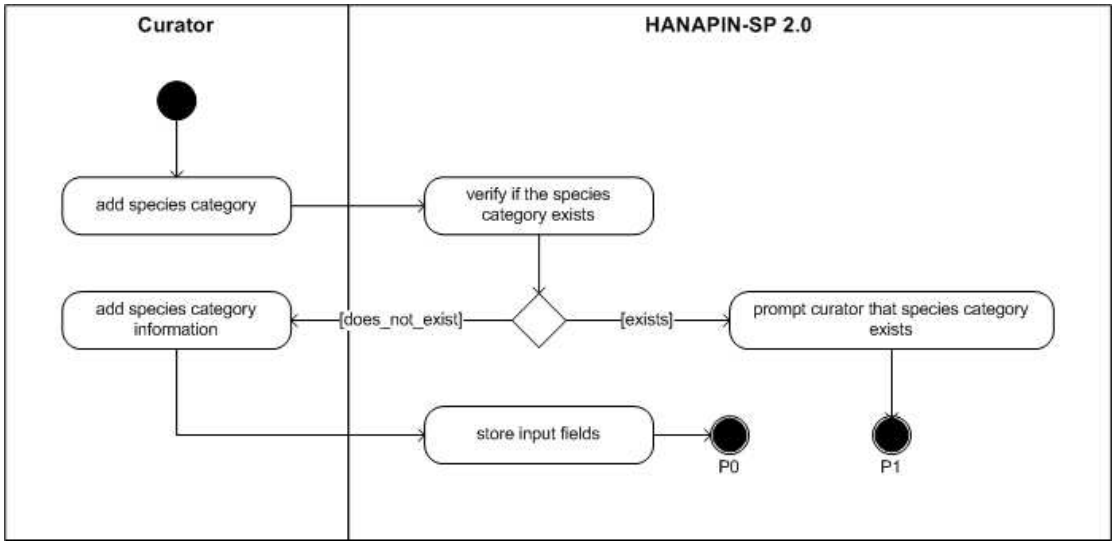

Figure 16: Add species group/family/genus activity diagram of HANAPIN-SP 2.0

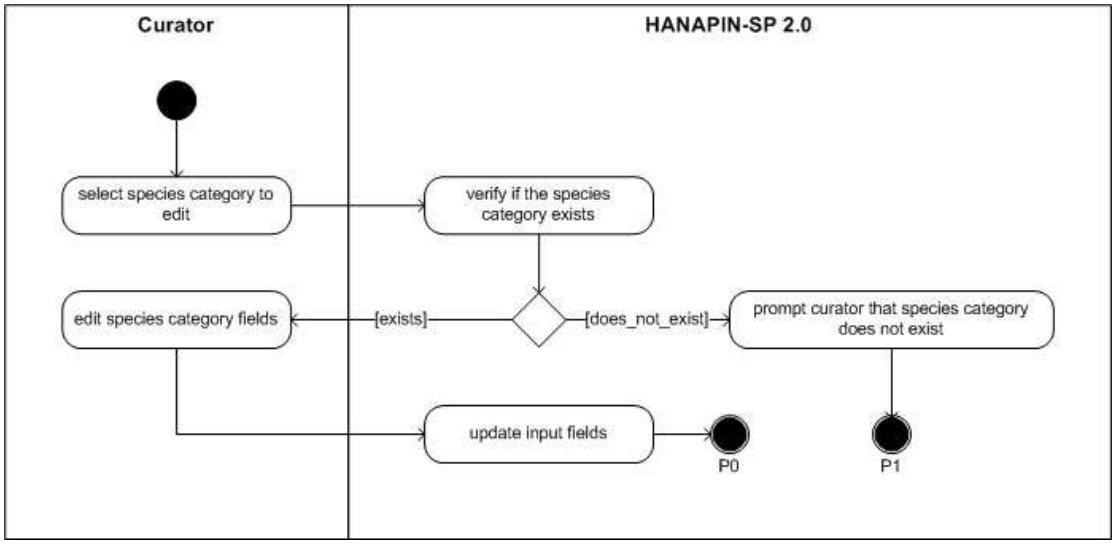

Figure 17: Edit species group/family/genus activity diagram of HANAPIN-SP 2.0

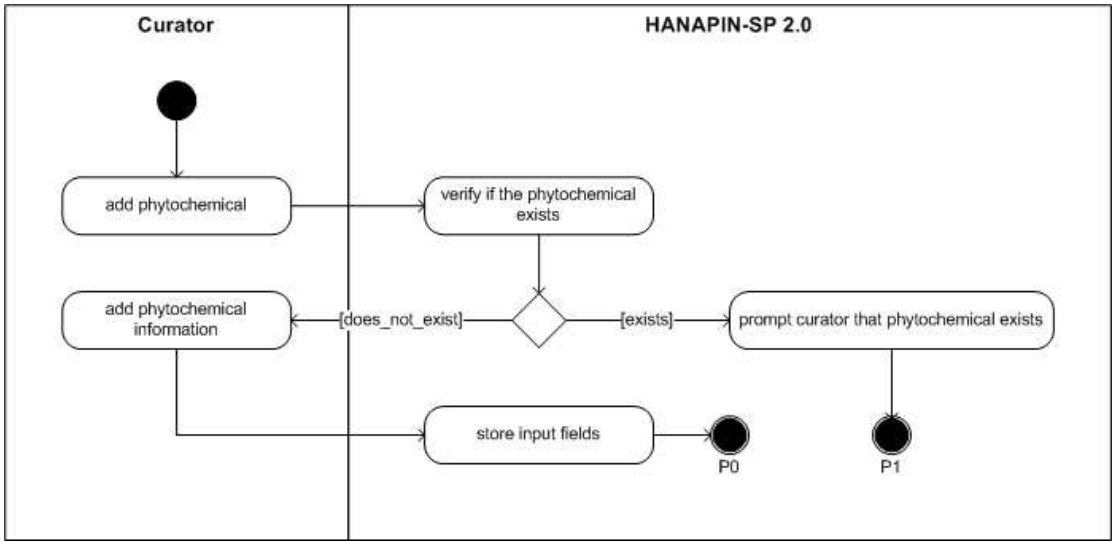

Figure 18: Add phytochemical activity diagram of HANAPIN-SP 2.0

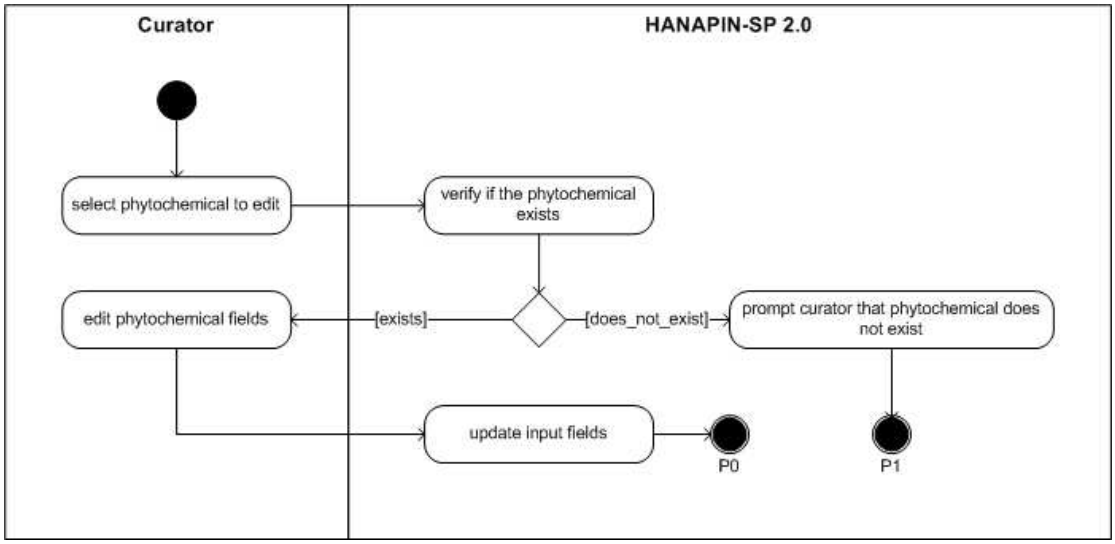

Figure 19: Edit phytochemical activity diagram of HANAPIN-SP 2.0

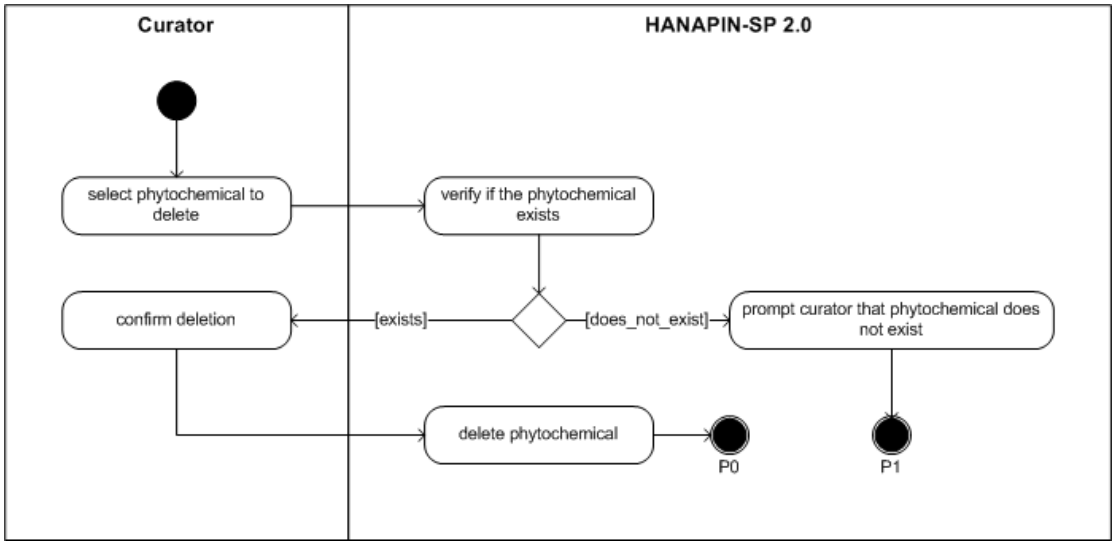

Figure 20: Delete phytochemical activity diagram of HANAPIN-SP 2.0

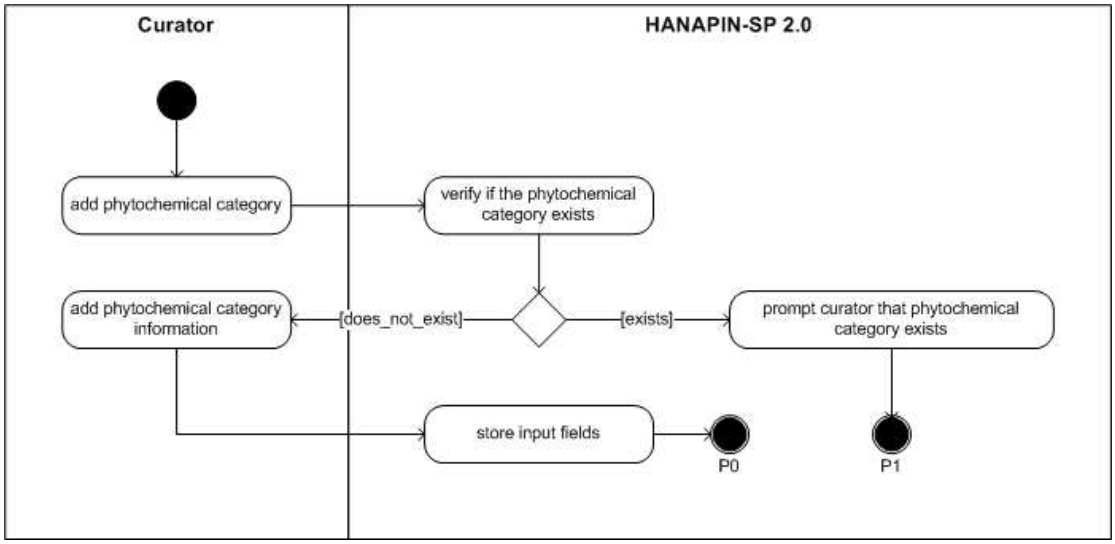

Figure 21: Add phytochemical category activity diagram of HANAPIN-SP 2.0

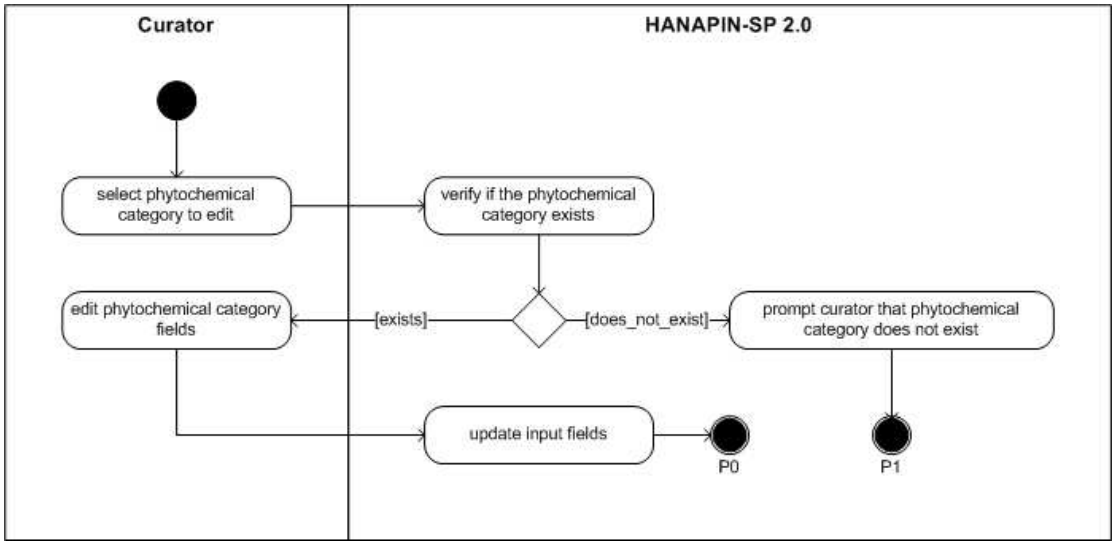

Figure 22: Edit phytochemical category activity diagram of HANAPIN-SP 2.0

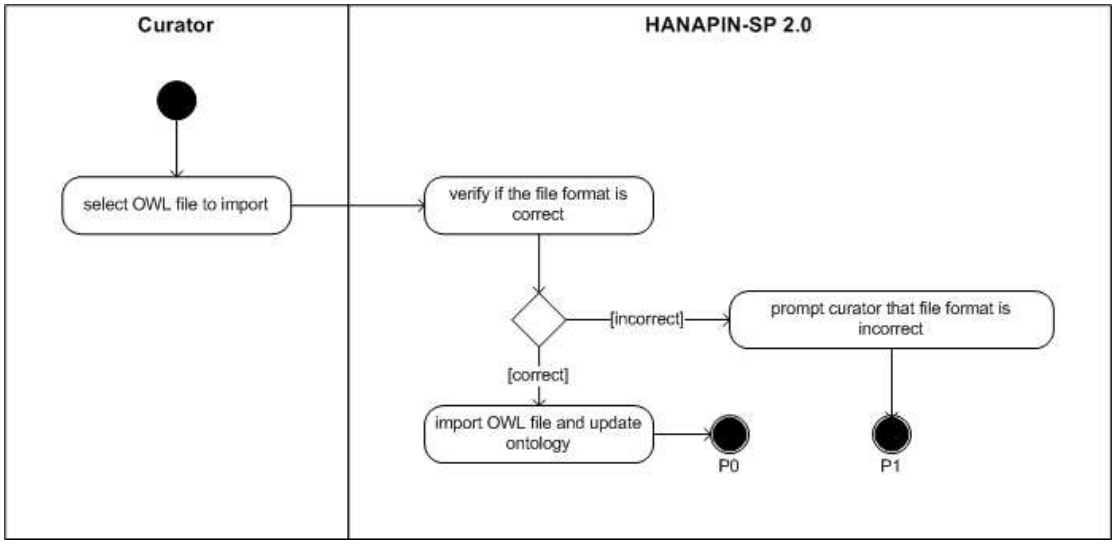

Figure 23: Import ontology activity diagram of HANAPIN-SP 2.0

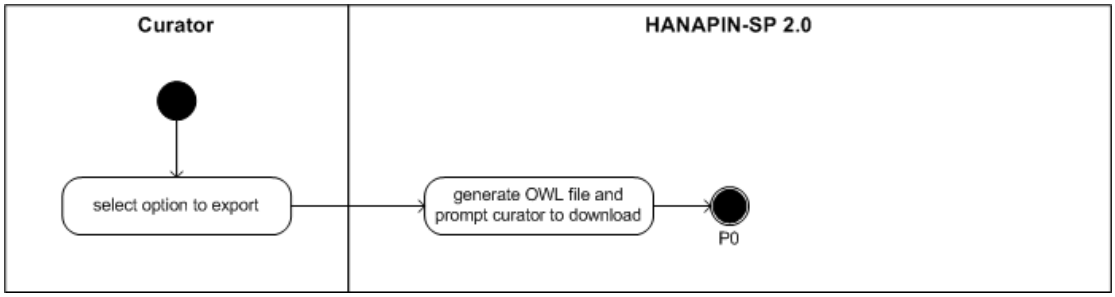

Figure 24: Export ontology activity diagram of HANAPIN-SP 2.0

4. Manage User Accounts

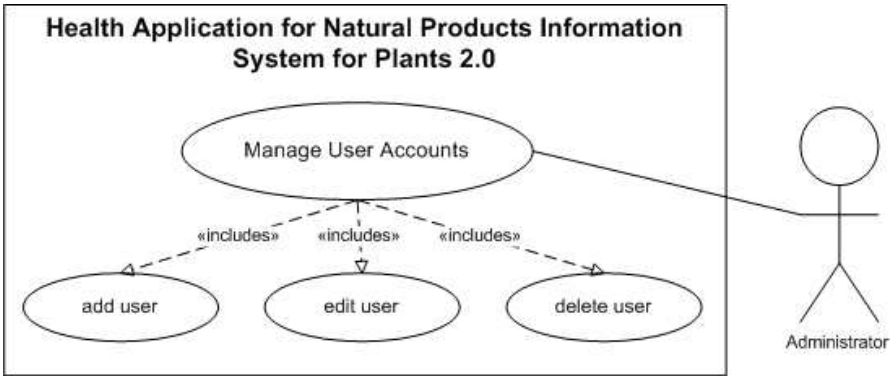

Figure 25: Manage User Accounts use case diagram of HANAPIN-SP 2.0

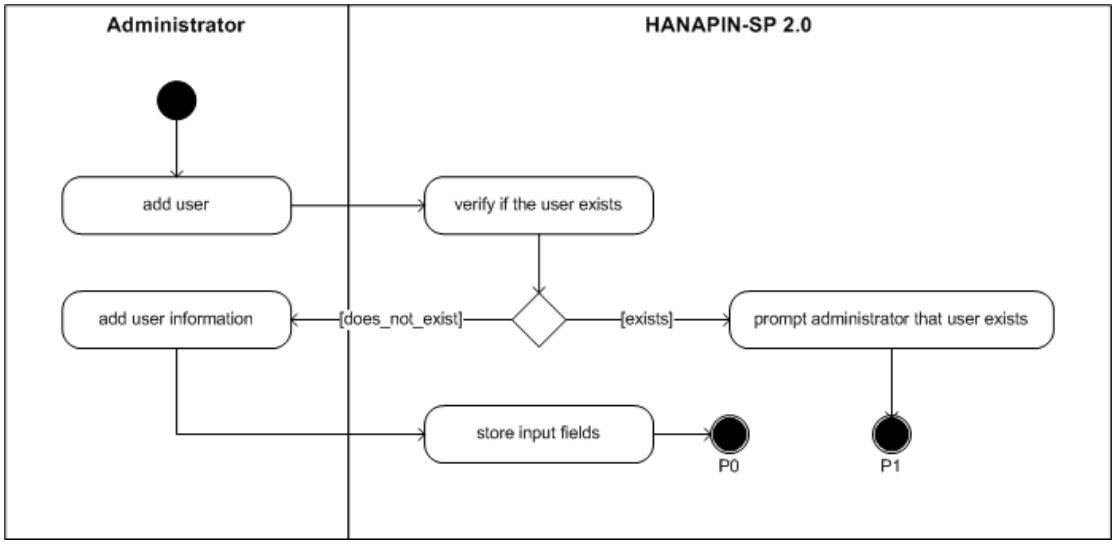

Figure 26: Add user activity diagram of HANAPIN-SP 2.0

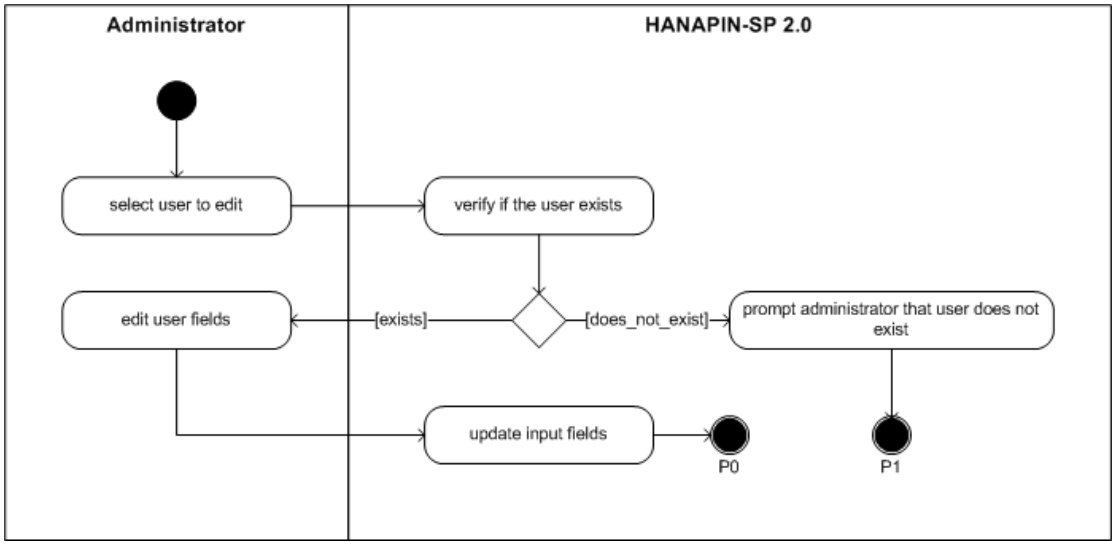

Figure 27: Edit user activity diagram of HANAPIN-SP 2.0

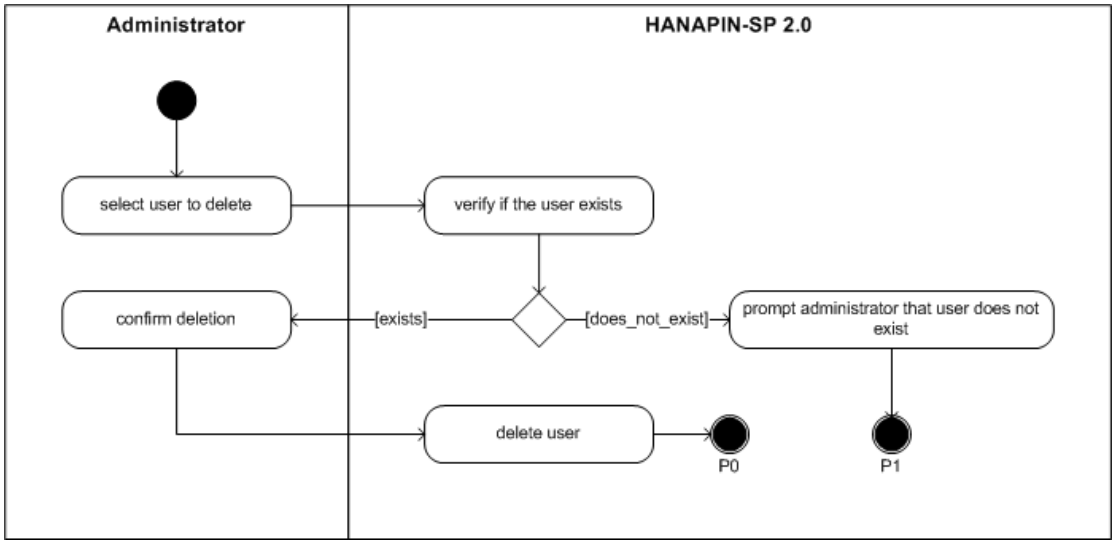

Figure 28: Delete user activity diagram of HANAPIN-SP 2.0

4.3.3 Entity Relationship Diagram

HANAPIN-SP 2.0 is comprised of two sets of tables: one for users, and one for the ontology. The ontology tables are automatically generated by Jena, and the system queries the ontology tables through Jena, without knowing the underlying structure of the tables. Thus, only the user tables are shown in the entity relationship diagram in Figure 29.

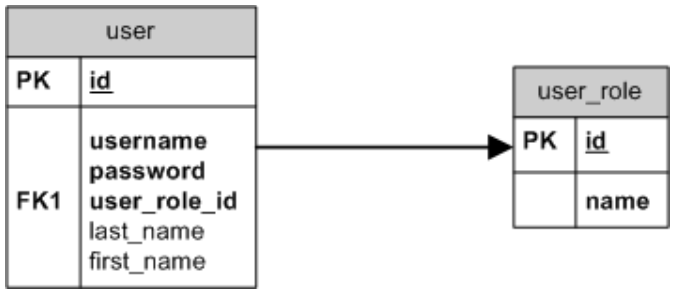

Figure 29: Entity relationship diagram of HANAPIN-SP 2.0

4.3.4 Data Dictionary

Attribute	Data Type	Description
id	int(20)	ID of the user
username	varchar(20)	User name of the user
password	varchar(32)	MD5-hashed password of the user
user_role_id	int(20)	ID of the role of the user
last_name	varchar(30)	Last name of the user
first_name	varchar(30)	First name of the user

Table 1: `user` table

Attribute	Data Type	Description
id	int(20)	ID of the user role
name	varchar(20)	Name of the user role

Table 2: `user_role` table

4.3.5 System Architecture

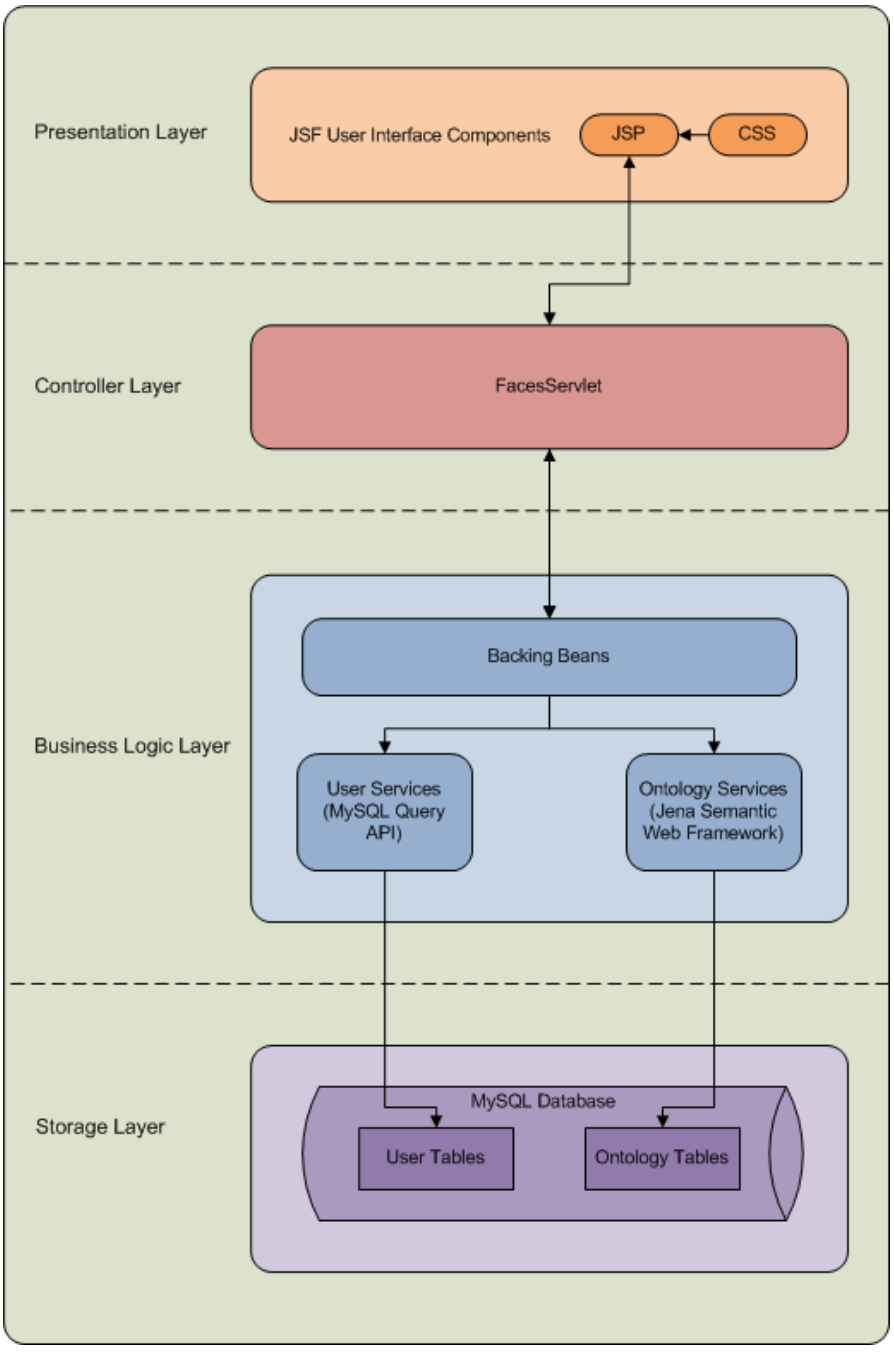

Figure 30: Architecture of HANAPIN-SP 2.0

HANAPIN-SP 2.0 stores its data in a relational database, wherein one part (the User Services) is queried by a MySQL Query API (Java Database Connectivity driver 5.1.16), and the other part (the Ontology Services) functions as an ontology that is to be queried by Jena (version 2.6.4).

For the User Services, there are only two tables: `user` and `user_role`. There are three user levels: unregistered user, curator, and administrator. A plain relational database storage is preferred (instead of an ontology), since the structure of the data is relatively static.

For the Ontology Services, the table and table structure was created by Jena automatically, by converting the OWL file to a format that can be stored in a relational database. The ontology is defined in OWL, because it is already used by Batista-Navarro et al. [3] and it is the standard for defining ontologies. To facilitate the editing of the OWL ontology, the ontology editor Protégé is used. The backing beans (written in Java) uses the Jena framework to access the ontology—this includes reading from, querying, and writing to the ontology.

HANAPIN-SP 2.0 uses the JavaServer Faces (JSF) framework version 2.1. The PrimeFaces[9] library version 2.2.1 is used to generate the JSF tree and autocomplete components. The imgscalr[10] Java image scaling library version 3.2 is used to generate thumbnails of uploaded images.

[9] http://www.primefaces.org/

[10] http://www.thebuzzmedia.com/software/imgscalr-java-image-scaling-library/

5 Technical Architecture

To run the system server, the following are required:

- Apache Tomcat 6.0
- MySQL 5
- Any operating system that supports the aforementioned software

The server has been tested to run on a machine with the following specifications: Windows 7 operating system, a quad-core processor at 2.66 GHz, and 2.00 GB of memory.

To run the system client, which is web-based, only a browser is required. JavaScript support is recommended. The following browsers have been tested to run the system: Microsoft Internet Explorer, Mozilla Firefox, and Google Chrome.

6 Results

6.1 Plant Natural Products Ontology

The plant and plant natural product (or phytochemical) classes have been developed, as seen in Figure 31 and 32, respectively. The plant classification system follows the system used in The Plant List[11], a collaborative venture coordinated by the Royal Botanic Gardens, Kew and the Missouri Botanical Garden and involving collaborators worldwide. The classification systems of both classes have been approved by Dr. Carillo.

The fields of each species are the following: scientific name, scientific-name synonyms, local names, phytochemicals found, traditional knowledge, locations, and images.

[11] http://www.theplantlist.org/

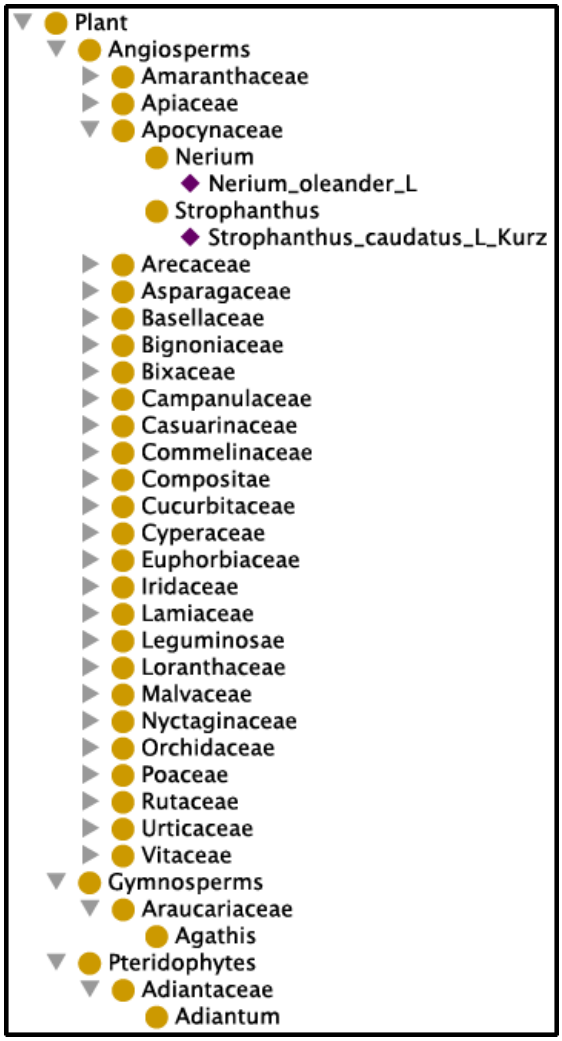

Figure 31: Plant Ontology. Some sections have been expanded to show the hierarchy. A yellow circle indicates a class, while a purple diamond indicates an instance.

The fields of each phytochemical are the following: phytochemical name, KEGG (Kyoto Encyclopedia of Genes and Genomes) identifier, ChEBI (Chemical Entities of Biological Interest) identifier, formula, molecular weight, IUPAC (International Union of Pure and Applied Chemistry) name, and synonyms.

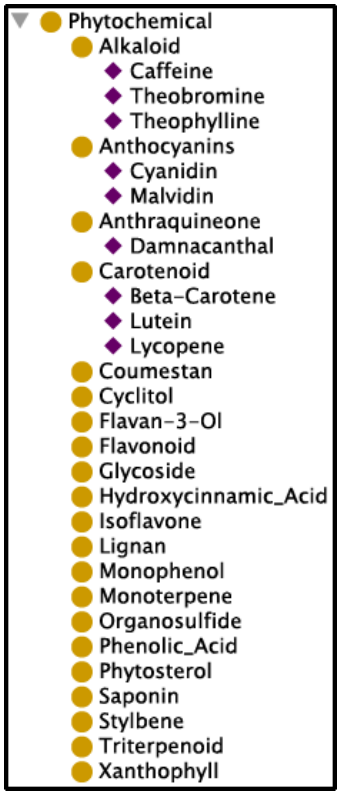

Figure 32: Plant Natural Products Ontology. Some sections have been expanded to show the hierarchy.

The resulting ontology has defined the overall structure, and more plants and phytochemicals may be added in the future. The ontology does not aim to be complete, because more plant and phytochemical links are continuously being discovered in the Philippines, and because it is the nature of ontologies to change constantly.

6.2 HANAPIN-SP 2.0

The following screenshots show what all HANAPIN-SP 2.0 users (unregistered users, curators, and administrators) are allowed to do in the system.

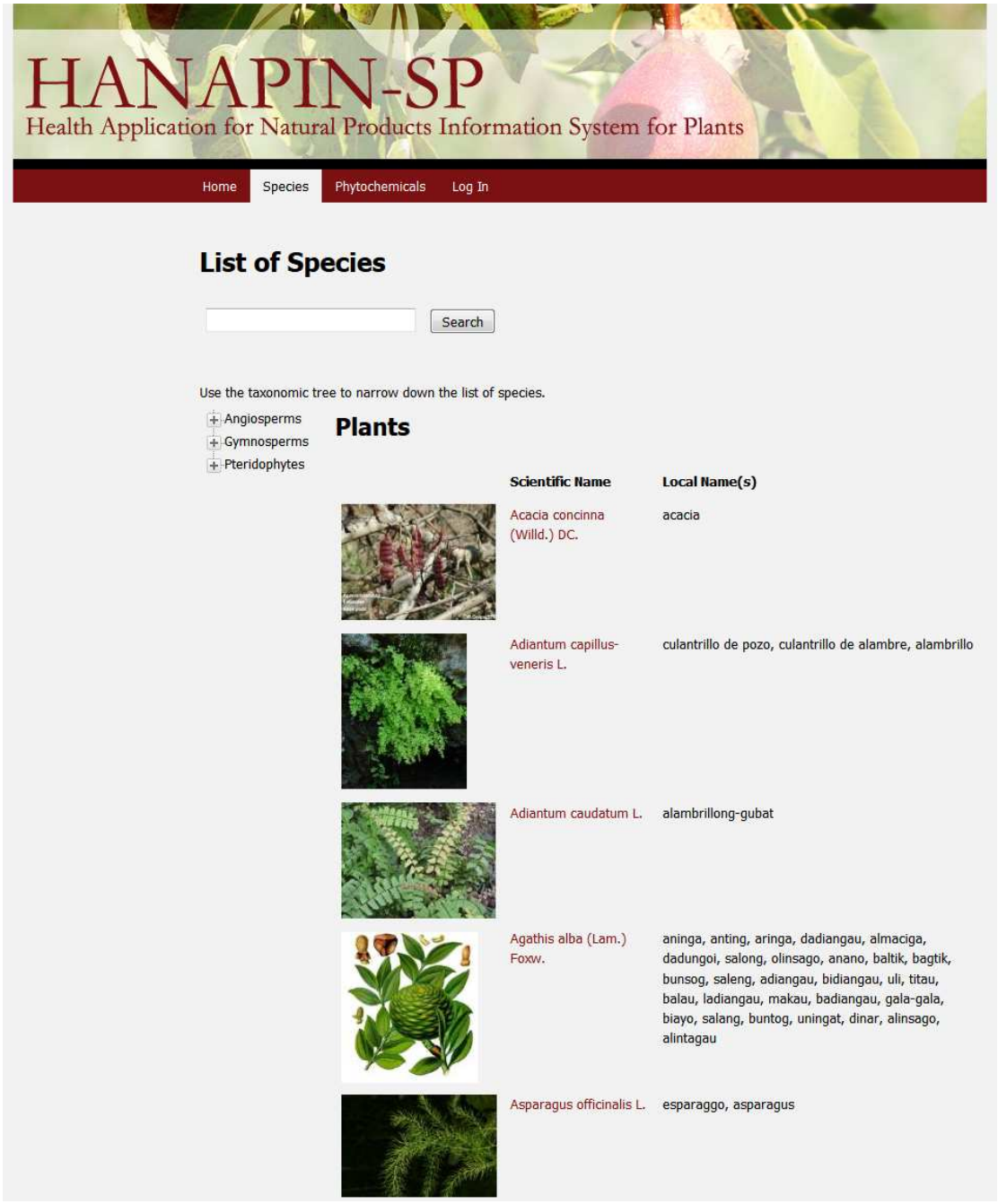

Figure 33: Browse the list of species.

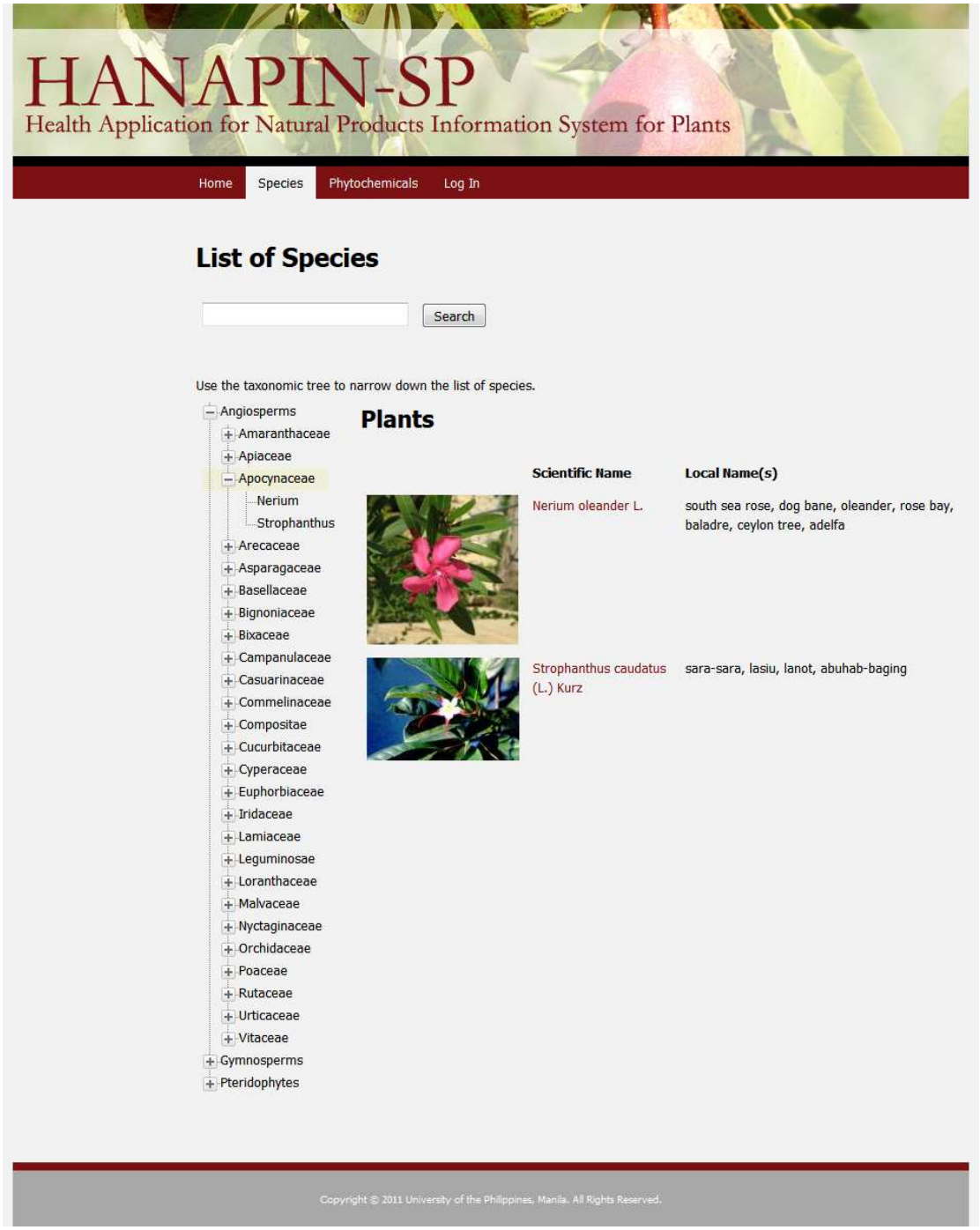

Figure 34: Use the tree at the left to narrow down the list of species.

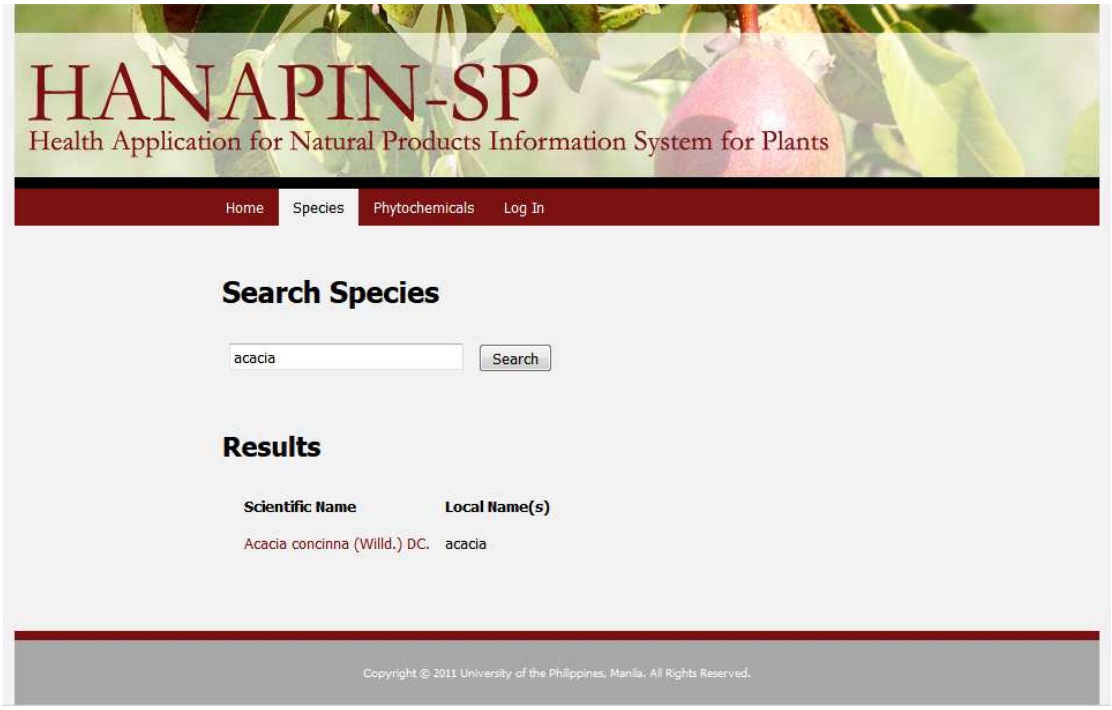

Figure 35: Search the list of species by typing a keyword.

Figure 36: View a species.

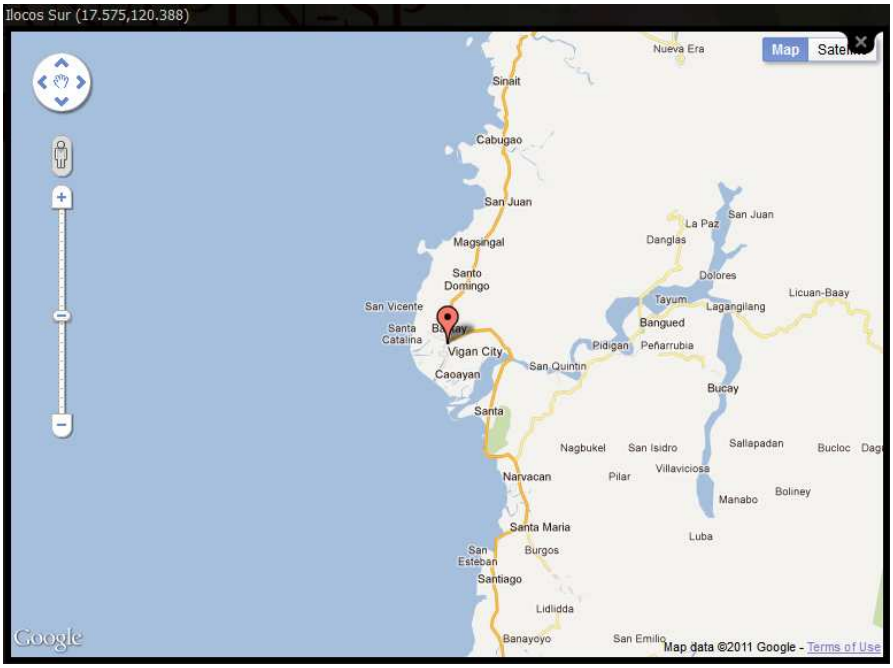

Figure 37: Clicking a location brings out a window inside the page that shows its position using Google Maps.

Figure 38: Clicking an image brings out a window inside the page that shows the image in its original size.

Figure 39: Browse the list of phytochemicals.

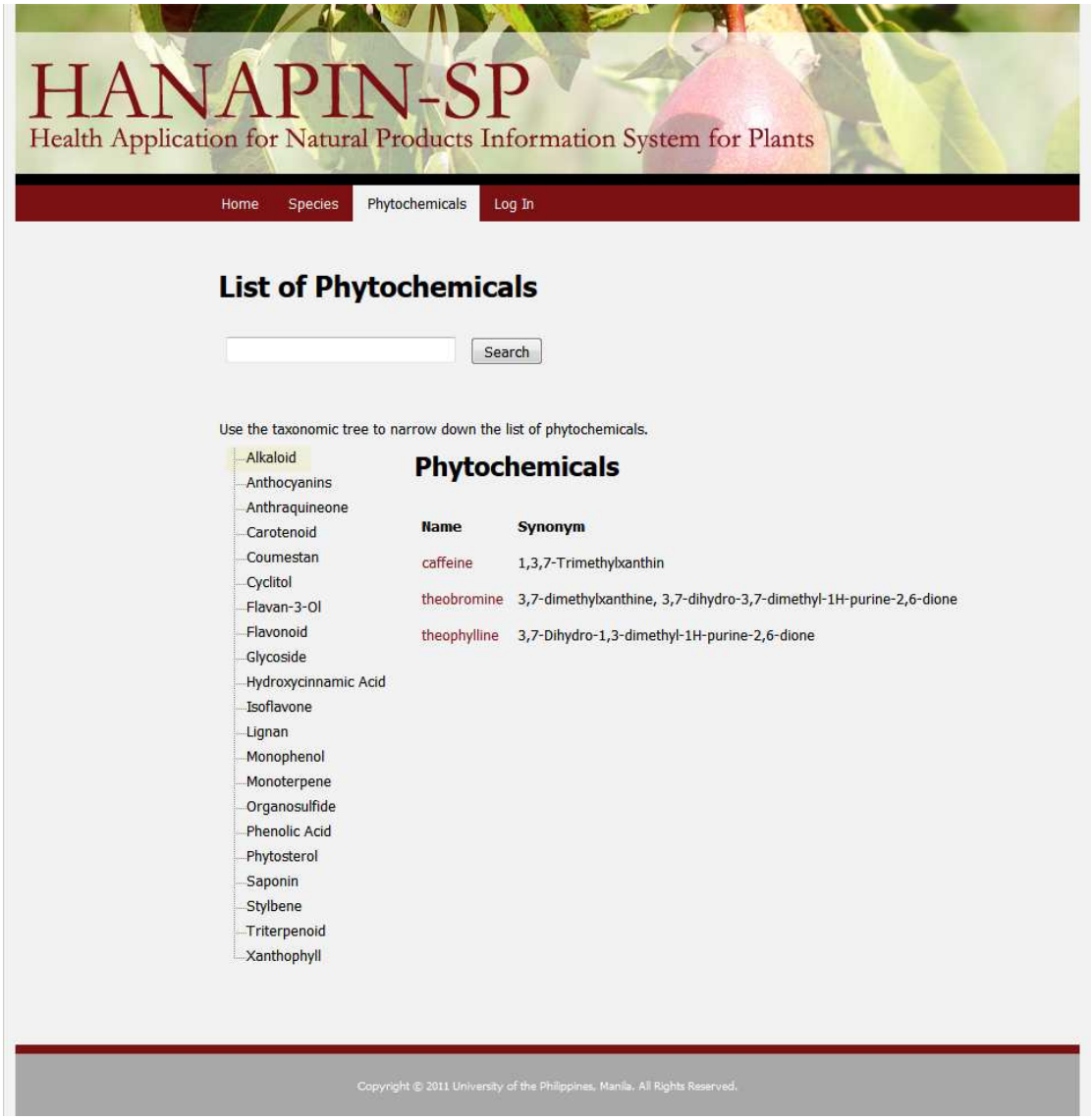

Figure 40: Use the tree at the left to narrow down the list of phytochemicals.

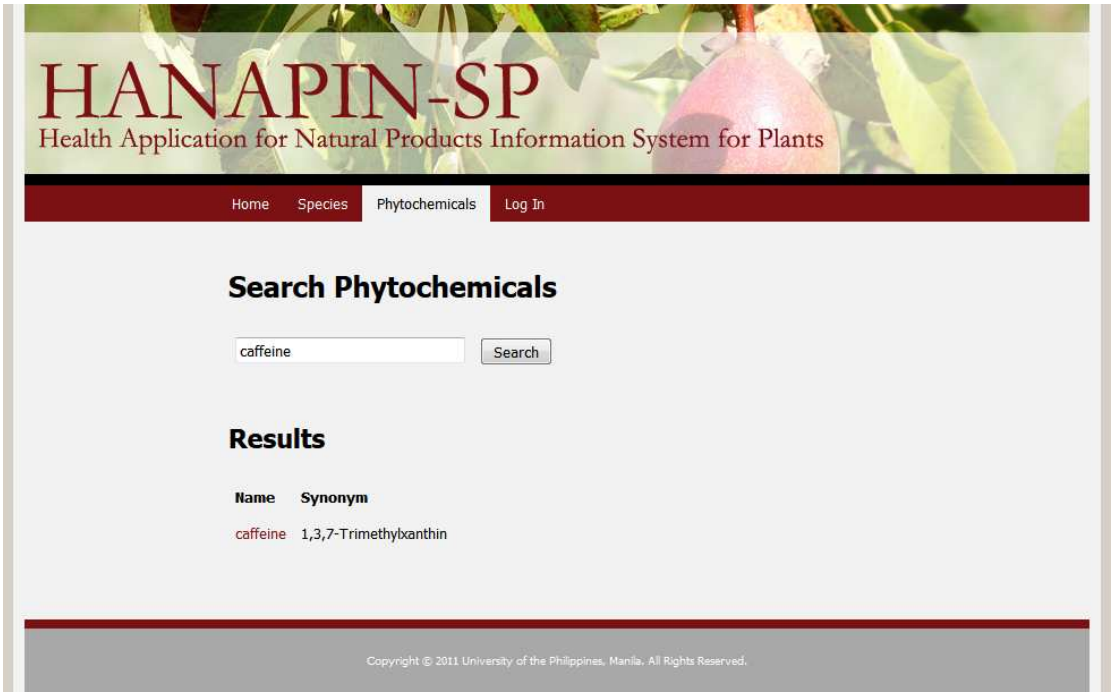

Figure 41: Search the list of phytochemicals by typing a keyword.

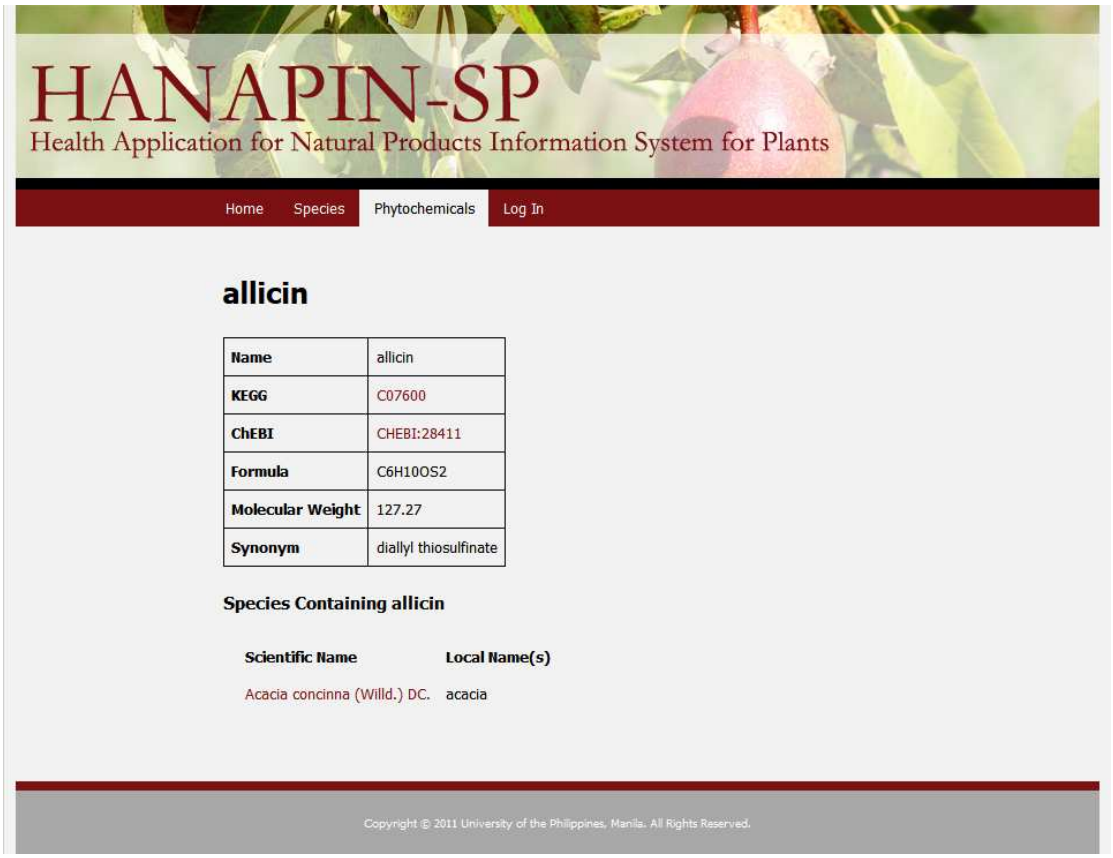

Figure 42: View a phytochemical.

The following screenshots show what only registered users are allowed to do in the system.

Figure 43: Log in to the system.

Figure 44: View account details.

Figure 45: Edit account details.

The following screenshots show what only curators are allowed to do in the system.

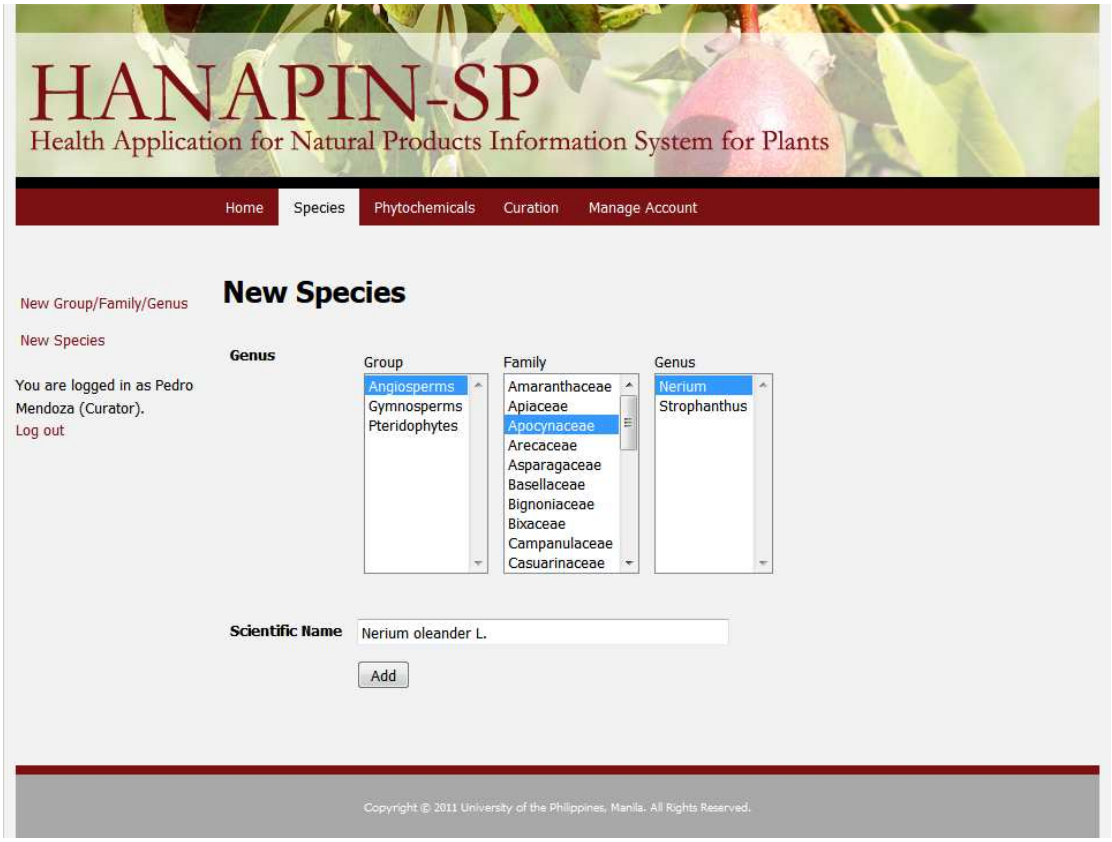

Figure 46: Add a new species.

Figure 47: Edit the scientific name of a species.

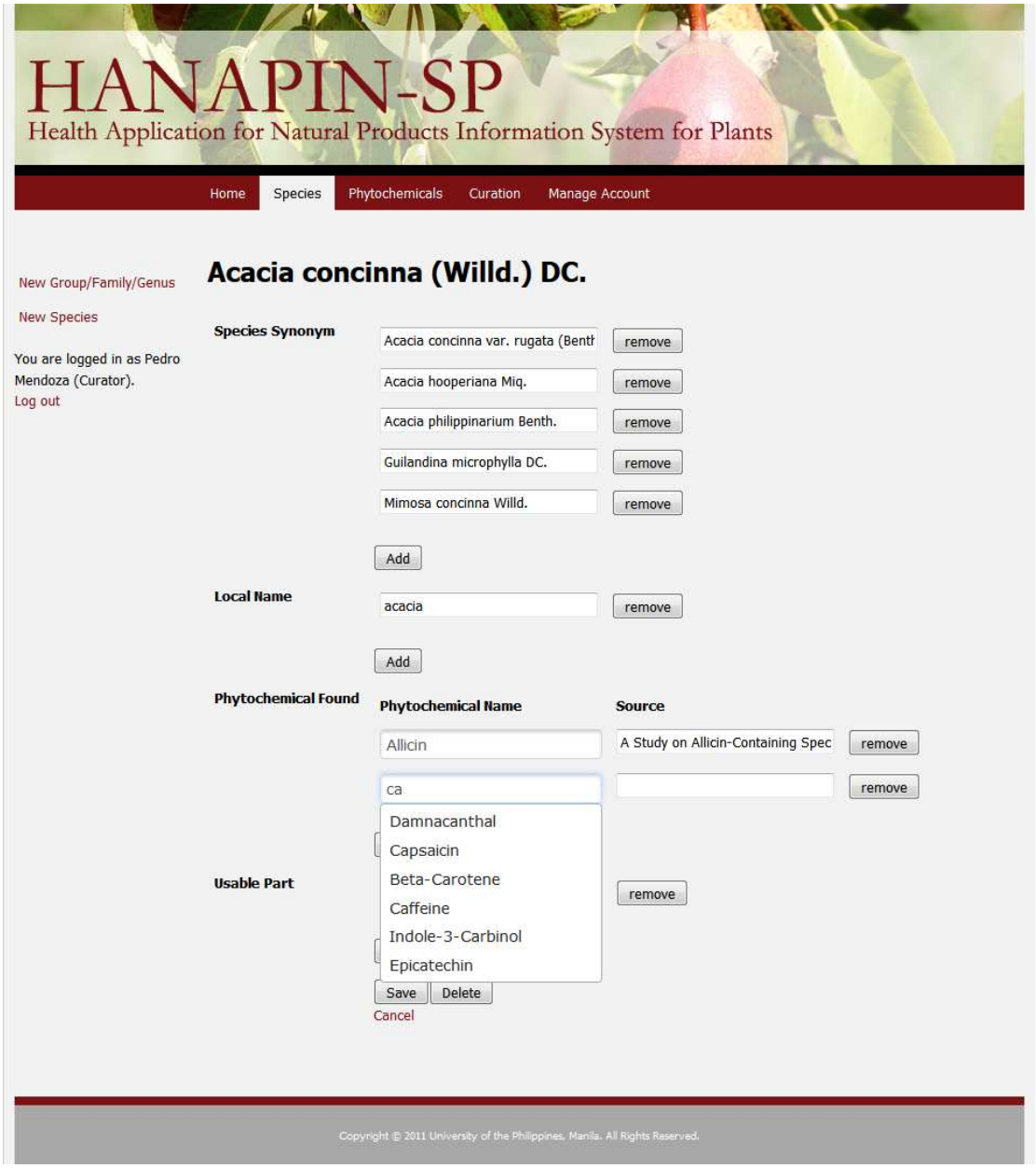

Figure 48: Edit the synonyms, local names, phytochemicals found, and usable parts of a species. The screenshot shows how the curator can use the dropdown box in the phytochemical field so that only an existing phytochemical can be chosen.

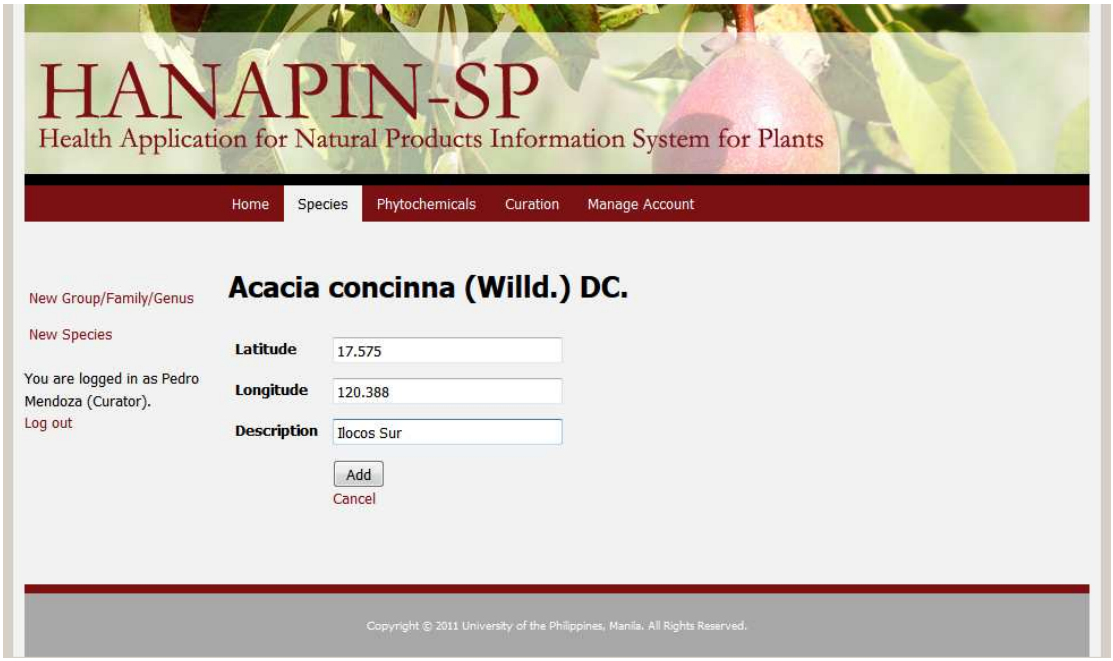

Figure 49: Add a new location to a species.

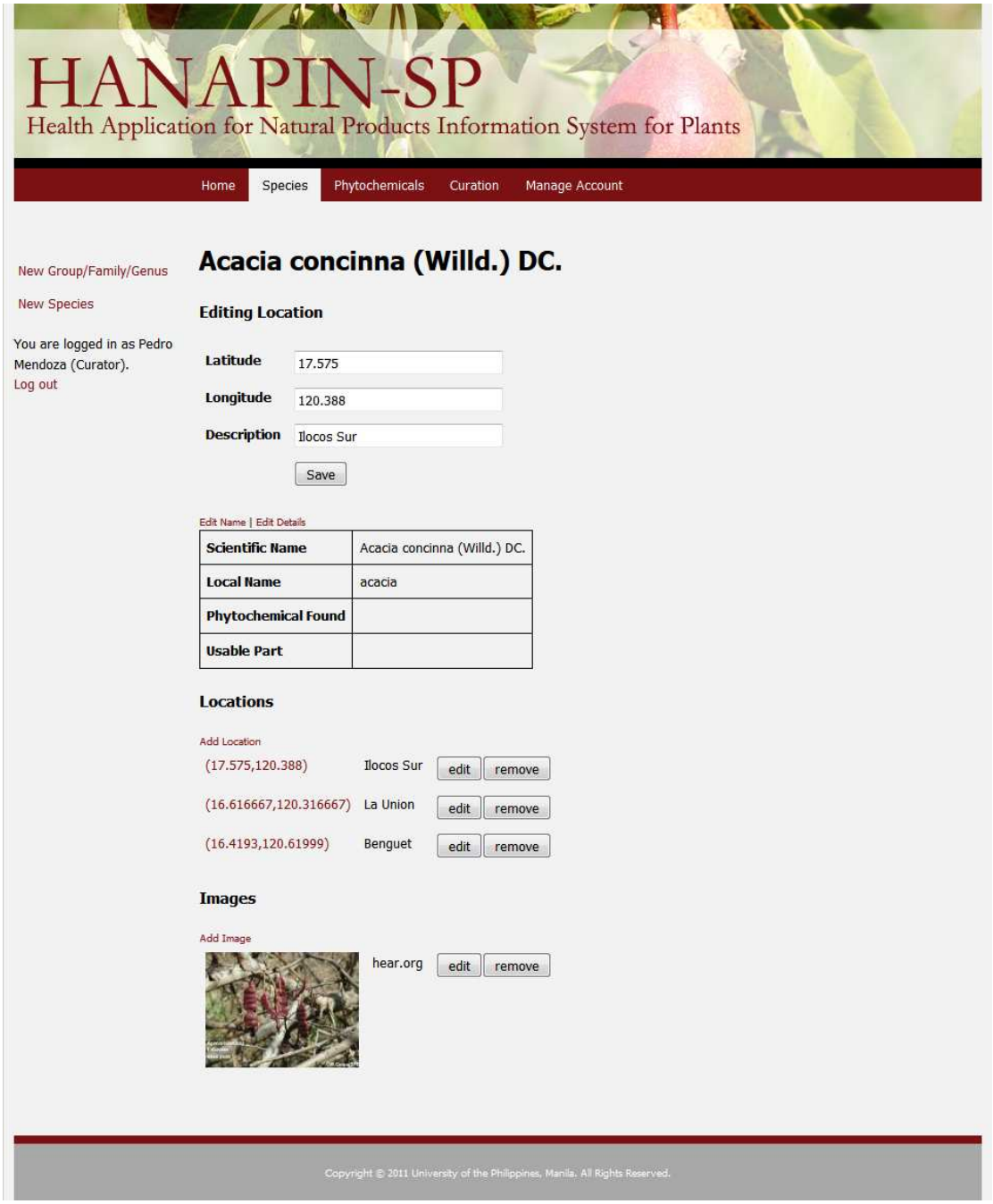

Figure 50: Edit an existing location of a species.

Figure 51: Add a new image to a species.

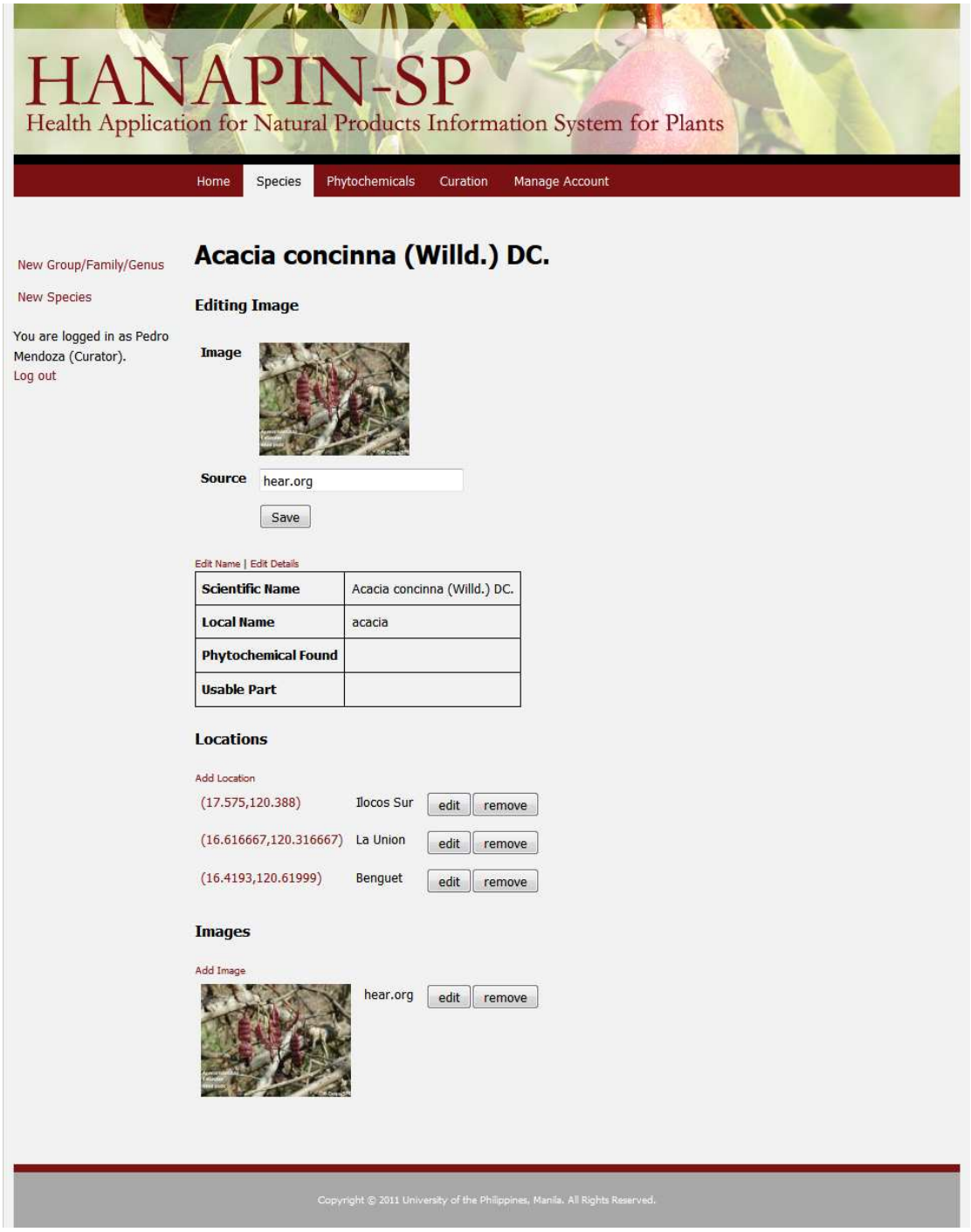

Figure 52: Edit an existing image of a species.

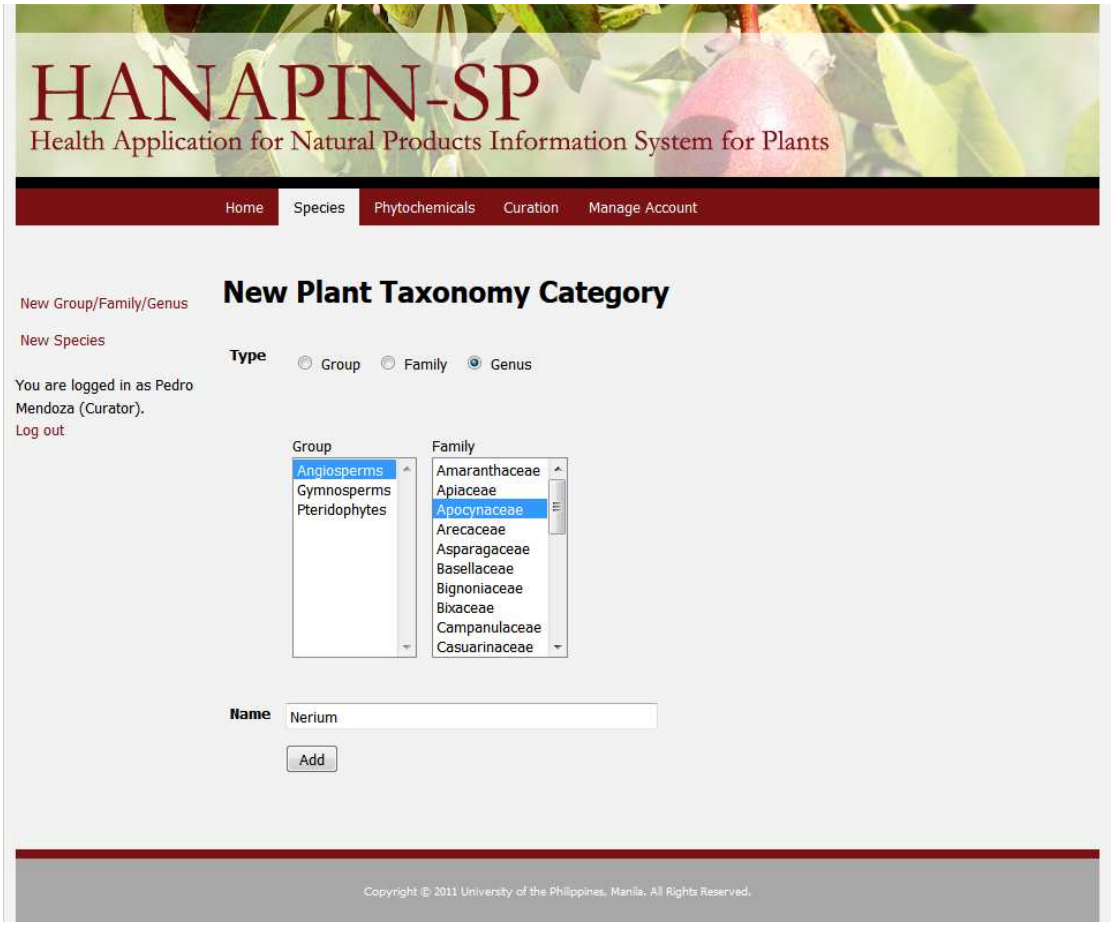

Figure 53: Add a new group, family, or genus to the plant taxonomy.

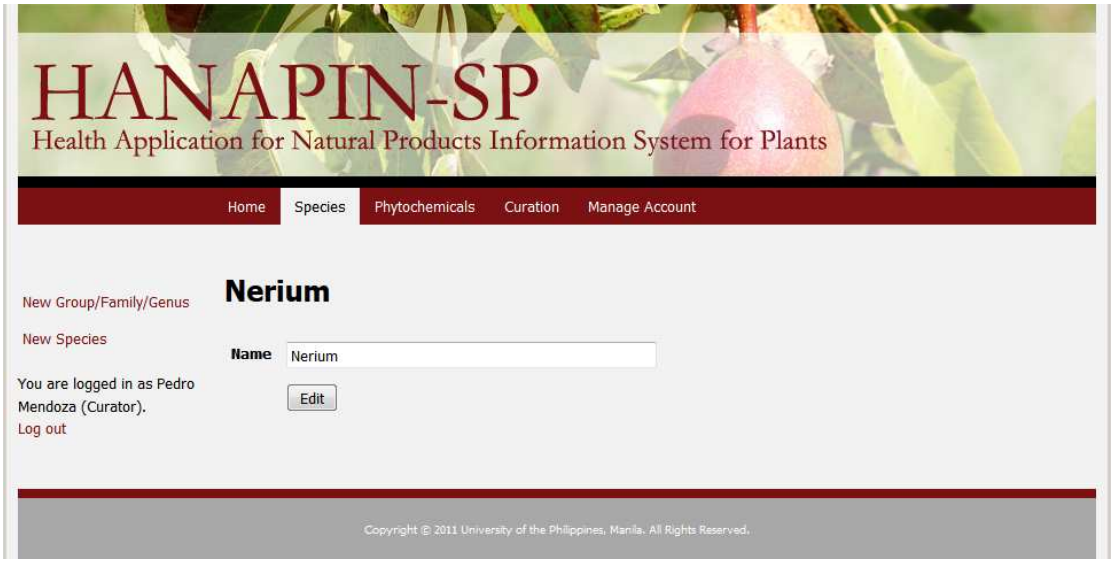

Figure 54: Edit an existing group, family, or genus of the plant taxonomy.

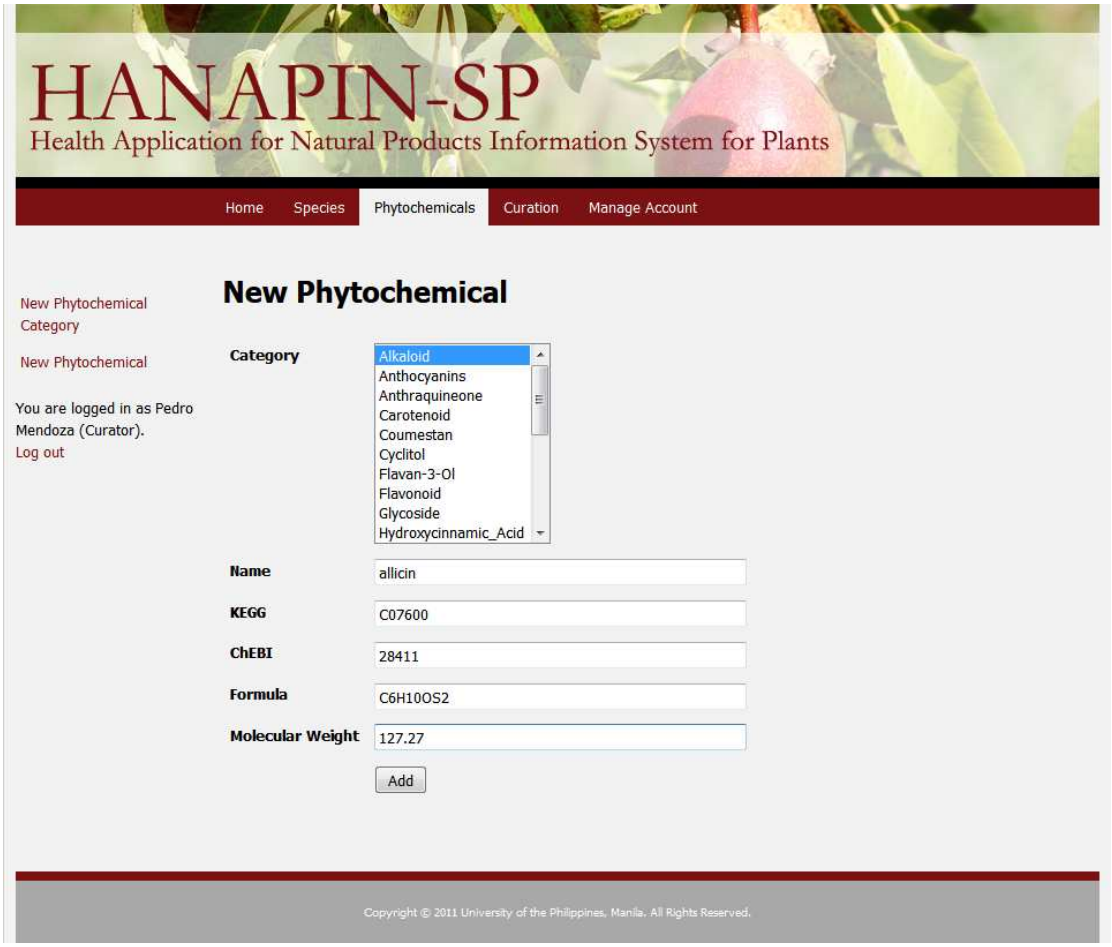

Figure 55: Add a new phytochemical.

Figure 56: Edit the name of a phytochemical.

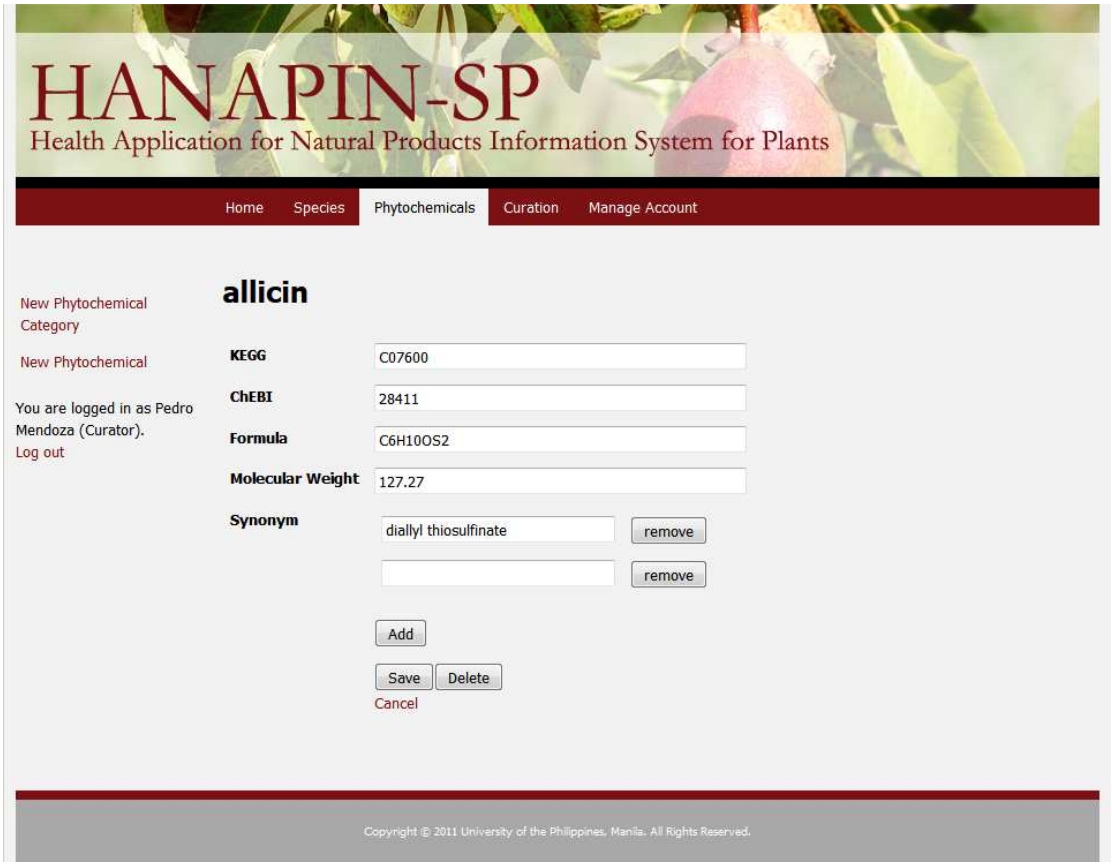

Figure 57: Edit the KEGG identifier, ChEBI identifier, formula, molecular weight, and synonyms of a species.

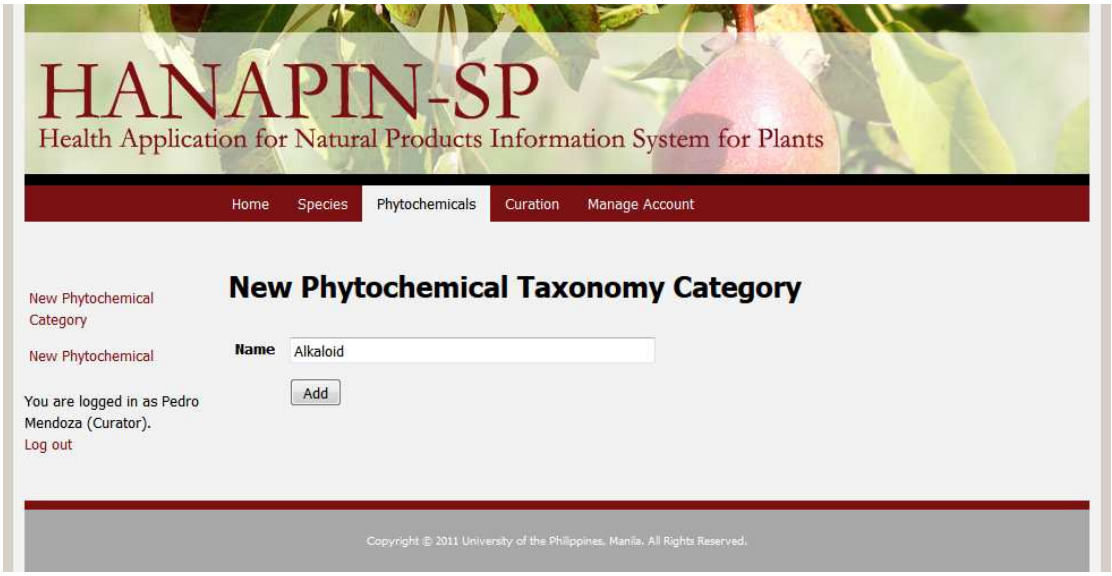

Figure 58: Add a category to the phytochemical taxonomy.

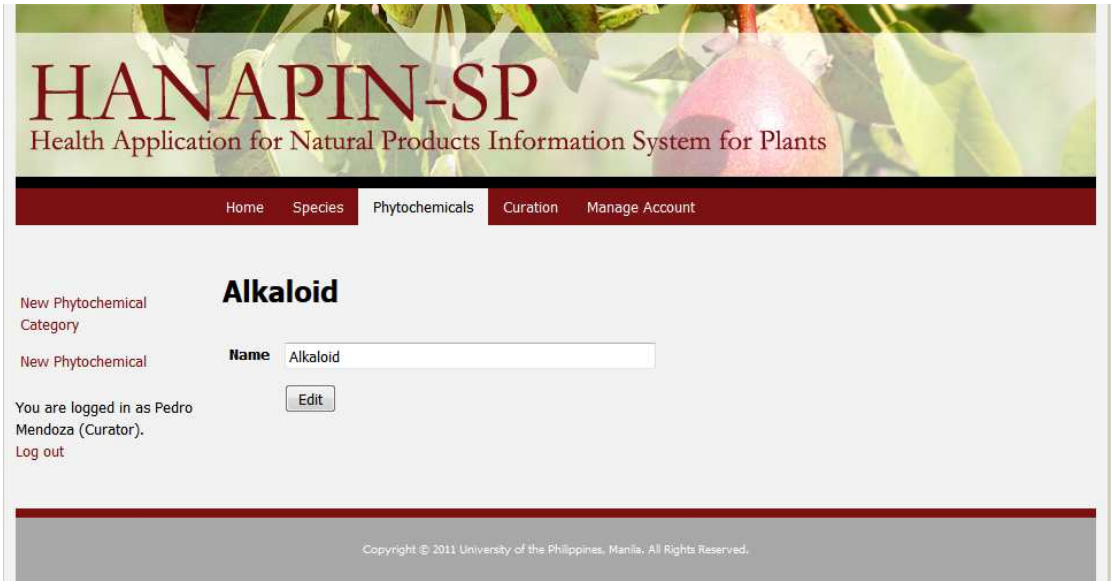

Figure 59: Edit an existing category of the phytochemical taxonomy.

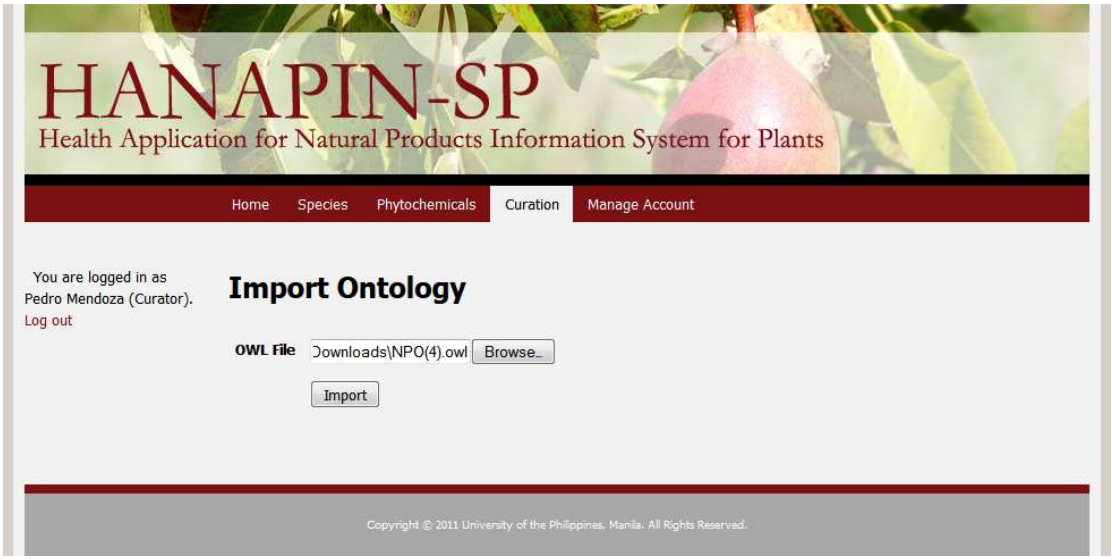

Figure 60: Import an ontology into the system.

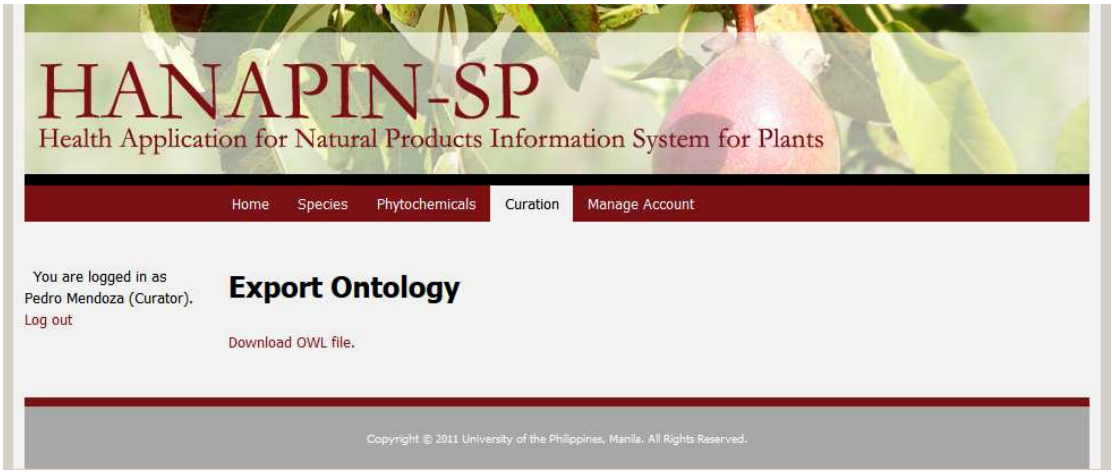

Figure 61: Export the ontology of the system.

The following screenshots show what only the administrator is allowed to do in the system.

Figure 62: View the list of users.

Figure 63: Register a new user.

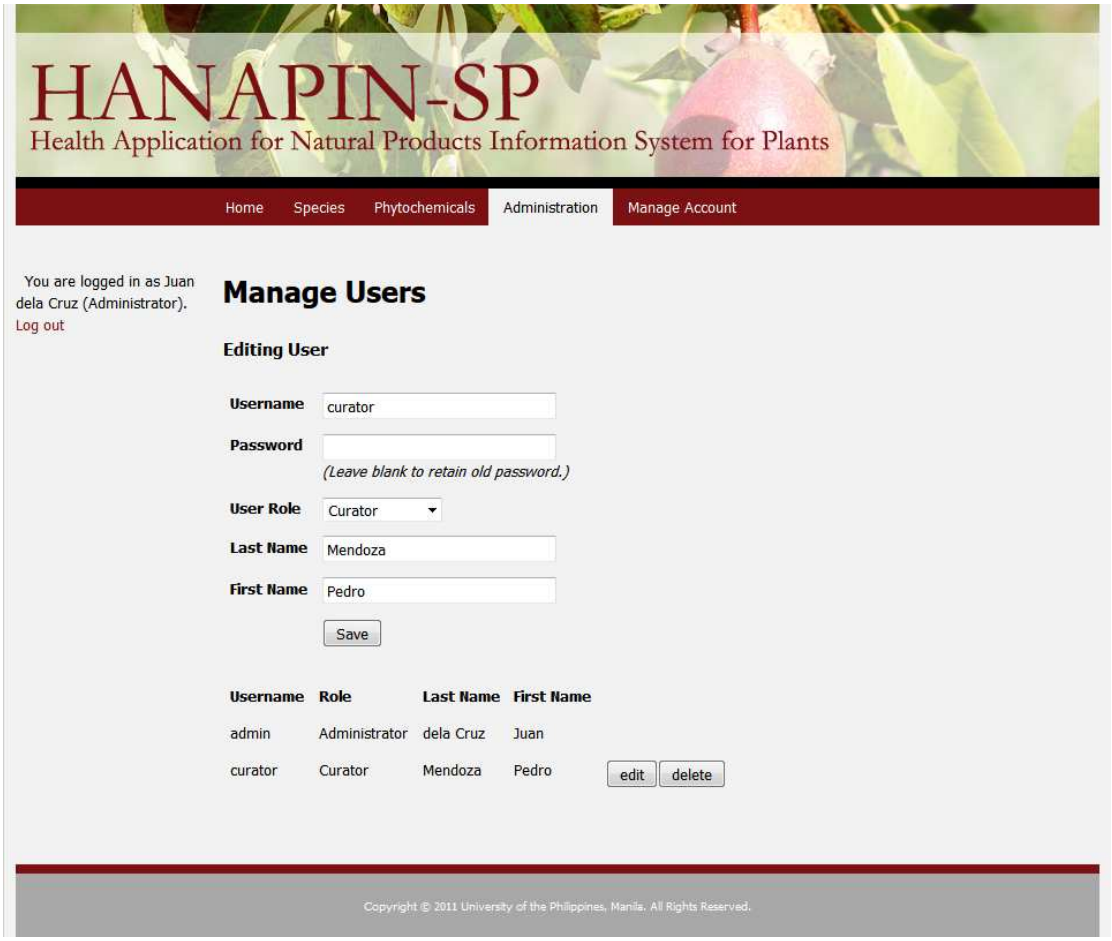

Figure 64: Edit an existing user.

7 Discussion

The general structure of the ontology has already been studied and defined by Batista-Navarro et al. [3], so the project can go directly to the third process, which is refinement. This includes the knowledge elicitation process with domain experts and formalization. After researching on different plant taxonomy classification systems and consulting with Dr. Carillo and other experts in the field, the use of the three-level classification system was chosen: the classes of groups contain subclasses of families, which contain subclasses of genera, which contain instances of species.

Using Protégé, the initial structure of the ontology was defined, such as the data structure (fields) for species and phytochemicals, and some test classes and instances. Then the OWL ontology was imported into HANAPIN-SP 2.0 using Jena into a relational database format. Although the OWL file itself can be read from and written to, the ontology was moved to a relational database backend, where they are stored as RDF statements. A relational database backend was preferred because it is already proven as faster in querying, while there is no extensive study on the speed of querying an ontology directly, especially if it contains a large amount of data.

The Jena framework has a function for automatically converting ontologies to relational database tables (using the `createModel` method) and vice versa (using the `write` method). Although the data is stored in a relational database, the business logic layer manages the data as an ontology. When the system queries the ontology, the query passes through Jena, which builds the appropriate SQL statement. Therefore, in building queries, it is not necessary to know the structure of the relational database tables that Jena created, as Jena automatically translates from ontology query to relational database query.

The system is able to display the tree-like structure reflecting the species taxonomy by making the appropriate Jena call (`listSubClasses`) at every level. Each Jena call outputs the direct subclasses or instances of the current class, so

the function to output the tree is a recursive function, stopping when there are no further subclasses at a given level. The result must also be processed to obtain the actual data value, since the raw output from Jena includes meta-data.

Additional work was done to ensure that no same resource (class or instance) has the same name as another resource. Whenever the name of a resource is changed, the system queries the ontology if a resource with the same name exists. If the name is already in use, the user is requested to provide another name; otherwise, the renaming is successful. As a rule, an ontology must have unique names for all of its resources. Jena has no built-in process of checking duplicate names, which, if left unchecked, can lead to two resources with the same name and thus ambiguous results—when that particular resource is called, any of the resources with the same name may be returned.

Jena considers its data to be immutable—they cannot be edited. Therefore, to edit data, the previous data must be deleted and then added again with the new value. Jena does this automatically in renaming a resource. It automatically deletes the previous resource, creates a new resource with the new name, and updates all references to this resource. However, editing fields within the resource is not automatic. Thus the system accomplishes this step manually but in the same way: by removing the previous value of the field, and then adding a new field value.

Searching for a resource by a keyword is simple if only the field values are to be queried. However, searching by category (for example, species can be searched by group, family, or genus) gives the additional benefit of easier searching even when the user does not know the exact hierarchy of what he is searching for. But this is difficult for Jena because the method to retrieve the parent of a resource (`getProperty` with argument of `type` or `subClassOf`) returns only the immediate parent. Therefore, to query for all the parent classes, either direct or indirect, required a recursive function that called `getProperty`. Likewise, to display the classification hierarchy on an individual species page needed the same

algorithm of recursive calls.

The ontology can be exported to RDF format (using the `write` method) and edited in an ontology editor (such as Protégé) outside of HANAPIN-SP 2.0. New classes and instances can be added, edited, and deleted, and new fields can be added to the structure of the species and phytochemicals. Then the ontology can be imported back into the system (using the `createModel` method), replacing the existing ontology. New and deleted classes and instances are reflected in the system without any changes necessary.

Field (or structure) changes are partially supported in HANAPIN-SP 2.0. Outside changes to the classification structure (that is, edits on the file directly or through an ontology editor that is outside the system) can be displayed immediately in the system. New fields may be added without affecting the system. To display new fields, they must be added to a comma-separated list in a configuration file in the system (for example, `Scientific_Name, Species_Synonym, Local_Name, Phytochemical_Found, Usable_Part`). This list is also used to determine the order in which the fields are displayed. The new fields will be displayed without the need for any additional coding or even a recompilation. This is possible, because instead of hard-coding the fields to be fetched using Jena (`getProperty` with argument of the property that is required), they are specified in a configuration file. Thus the system fetches the value of the specified fields, and displays the value to the user. Omitting the configuration file is not possible, because the system cannot control the order in which the fields are displayed, since Jena returns the data in the order it was entered. Two fields can also be displayed as a pair, that is, one field supplements information for another field (for example, the phytochemical found is paired with another field describing the source reference, see Figure 36). Fields can be paired without recompilation by specifying the paired fields in the configuration file (`npo.plant.prop1.fields = Phytochemical_Found, Usable_Part` and `npo.plant.prop2.fields = Phytochemical_Source, Usable_Part_Use`). Values of new fields are dis-

played as plain text; if they are to be displayed with additional details (for example, as a link), a few lines of coding and a recompilation are needed.

Changes in the classification system and new fields added to the OWL file directly cannot be edited. The nature of forms in the JSF framework is that each field in the form corresponds to a hard-coded variable in the backing bean (a Java file). Therefore, new fields must be added in the form and as new variables in the backing bean, which entails a recompilation. Specifically, the new field must be coded in the object class with the appropriate getter and setter, in the JSF form as another input text box, and in the data loading and editing functions of the backing bean. An example can be seen below.

Listing 1: The configuration file

```
...

npo.plant.fields = Scientific_Name,Species_Synonym,Local_Name,
    Phytochemical_Found,Usable_Part,New_Field

...
```

Listing 2: The object class file

```
private String newField;

public String getNewField() {
  return newField;
}

public void setNewField(String newField) {
  this.newField = newField;
}
```

Listing 3: The JSF form

```
<h:inputText value=‘‘#{backingBean.object.newField}‘‘ />
```

Listing 4: The backing bean

```
private void loadObject() {
  ...

  String newField = ontology.getPropertyValue(ontologyResource, ''
     New_Field'');
  object.setNewField(newField);

  ...
}

public String edit() {
  ...

  Property newFieldProperty = ontology.getProperty(ontology.
     getProperty(''New_Field''));
  ontologyResource.removeAll(newFieldProperty);
  ontologyResource.addLiteral(newFieldProperty, object.getNewField
     ());

  ...
}
```

Jena creates its own MySQL database connection, but the connection expires after a set period of time (the wait timeout, configured in the server itself) and Jena does not automatically reconnect. This is inherently a Jena problem, and may be fixed outside of Jena by restarting the server periodically, ideally before the wait timeout period is completed. `cron`, a Unix time-based job scheduler, can be used to automatically restart the server. For example, if the wait timeout period is twenty-four hours, the restart can be scheduled every midnight: `1 0 * * * shutdown.sh|startup.sh`.

8 Conclusion

A Plant Natural Products Ontology has been created, based on the Marine Natural Products Ontology by Batista-Navarro et al. [3]. The ontology has defined the classification hierarchy and fields for both plants and plant natural products. This ontology has been incorporated into a system, HANAPIN-SP 2.0, which is not completely ontology-based, but can read and write data to the ontology using the Jena framework. The system allows displaying of new fields and changes in the classification structure without the need for recompilation.

9 Recommendation

The ontology development process is a process of continuous refining. Therefore, the Natural Products Ontology can be further improved and enriched by additional input from domain experts. More species are always available to be added to the system as more plant natural products are discovered in the Philippines. Details of existing species may be improved by adding locations, images, and other data.

Using the Ordered List Ontology[12] or something similar can be used to specify the order of the fields in the ontology itself, which would eliminate the need for maintaining a configuration file.

To further benefit from the adaptability and flexibility of ontology-driven information systems, the process of importing the ontology, or of displaying the data, or of both can be modified so that new fields added externally to the ontology may be edited inside the system without or less modification needed, preferably without the need for recompilation. A few recommendations would be to emulate an ontology editor inside the system, or to move to a different web framework. A possible alternative is Ruby on Rails, because one can use ActiveRDF, a library for accessing RDF data from Ruby programs.

While the current functionalities of HANAPIN-SP 2.0 are sufficient for what its users (mainly, the curators) need to store data, progress in the field of natural products research would entail a need for a more robust system. Specifically, focus can be given on different user privileges, and support for projects and collaboration.

While Jena is a robust Semantic Web framework, some problems were found in the course of developing the system. For example, Jena can add checking for duplicates and then throwing an exception if a resource with the same name exists. It can also automate the process of editing fields, instead of letting the user delete the previous entry and then adding a new entry in its place manually. The querying for the parent or super class of a certain resource is also very limited;

[12] http://smiy.sourceforge.net/olo/spec/orderedlistontology.html

Jena only returns the immediate parent, not the entire hierarchy. Providing an option for obtaining the entire hierarchy can be added to improve Jena. Lastly, Jena does not renew a connection after it has timed out; this problem can also be addressed, considering it is an essential part of the framework.

References

[1] G. J. A. Custodio, "Health Application for NAtural Products Information System for Plants (HANAPIN-SP)." University of the Philippines Manila, 2010.

[2] C. Nyulas, N. F. Noy, M. Dorf, N. Griffith, and M. A. Musen, "Ontology-driven software: What we learned from using ontologies as infrastructure for software." Stanford University, 2009.

[3] K. Manansala, R. T. Batista-Navarro, and E. Mendoza, "Towards the development of a natural product ontology." University of the Philippines Diliman, 2009.

[4] M. C. Carillo, "Personal interview." 19 May, 2011.

[5] S. Saad, N. Salim, H. Zainal, and Z. Muda, "A process for building domain ontology: An experience in developing Solat ontology," *2011 International Conference on Electrical Engineering and Informatics*, 2011.

[6] G. Brusa, M. L. Caliusco, and O. Chiotti, "A process for building a domain ontology: An experience in developing a government budgetary ontology," *Conferences in Research and Practice in Information Technology*, vol. 72, 2006.

[7] D. Hooijmaijers and M. Stumptner, "Trust based ontology integration for the community services sector," *Conferences in Research and Practice in Information Technology*, vol. 72, 2006.

[8] B. Orgun, M. Dras, A. Nayak, and G. James, "Approaches for semantic interoperability between domain ontologies," *Conferences in Research and Practice in Information Technology*, vol. 72, 2006.

[9] M. Fernández-López, "Overview of methodologies for building ontologies," *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5)*, vol. 18, 1999.

[10] M. Fernández-López and A. Gómez-Pérez, "A survey on methodologies for developing, maintaining, evaluating and reengineering ontologies." OntoWeb Consortium, 2002.

[11] A. Gómez-Pérez, M. Fernández-López, and O. Corcho, *Ontological engineering with examples from the areas of knowledge management, e-commerce, and the semantic web*. London, UK: Springer-Verlag London Limited, 2004.

[12] M. Amirhosseini and J. Salim, "Ontoabsolute as a ontology evaluation methodology in analysis of the structural domains in upper, middle and lower level ontologies," *2011 International Conference on Semantic Technology and Information Retrieval*, 2011.

[13] L. Ouyang and M. Qu, "A method of ontology evaluation based on coverage, cohesion and coupling," *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery*, 2011.

[14] N. F. Noy and D. L. McGuinness, "Ontology development 101: A guide to creating your first ontology." Stanford University, 2001.

[15] C. Lushbough, M. K. Bergman, C. J. Lawrence, D. Jennewein, and V. Brendel, "BioExtract server—an integrated workflow-enabling system to access and analyze heterogeneous, distributed biomolecular data," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 7, 2010.

[16] B. Jia, S. Xie, and J. Li, "Cross-domain privilege management system based on ontology in electronic commerce environment," *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery*, 2011.

[17] M. Laclavík, "RDB2Onto: Relational database data to ontology individuals mapping." Institute of Informatics, Slovak Academy of Sciences, 2008.

[18] I. Astrova and B. Stantic, "Reverse engineering of relational databases to ontologies: An approach based on an analysis of html forms." Tallinn University of Technology and Griffith University, 2007.

[19] Z. Xu, S. Zhang, and Y. Dong, "Mapping between relational database schema and owl ontology for deep annotation," *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, 2006.

[20] M. C. Carrillo, R. B. Chua, G. Solano, N. Quiming, J. Panibe, R. T. Batista-Navarro, K. Manansala, and E. Mendoza, "Health Application for NAtural Products INformation - System for Plants (HANAPIN-SP): An information database system for plant-derived natural products targeted for philippine researchers." University of the Philippines Manila, 2009.

[21] J. Cardoso, *Semantic Web Services: Theory, Tools and Applications*. IGI Global, 2007.

[22] D. L. McGuinness and F. van Harmelen, "OWL web ontology language overview." W3C, 2004.

[23] G. Velez, "Breaking into the semantic web, part I." The Voice of Semantic Web Business (semanticweb.com), 2008.

ConnectionPool.java

```
package dao;

import java.sql.Connection;
import java.sql.SQLException;

import javax.sql.DataSource;

import org.apache.commons.dbcp.ConnectionFactory;
import
org.apache.commons.dbcp.DriverManagerConnectionFac
tory;
import
org.apache.commons.dbcp.PoolableConnectionFactory;
import org.apache.commons.dbcp.PoolingDataSource;
import org.apache.commons.pool.impl.GenericObjectPool;
import org.apache.log4j.Logger;

public class ConnectionPool {

   private static Logger logger =
Logger.getLogger(ConnectionPool.class);

   private static ConnectionPool instance = null;

   private static DataSource ds = null;
   private static GenericObjectPool pool = null;

   private ConnectionPool() {
      try {
         DAOProperties properties = new
DAOProperties("hanapinsp2.jdbc");

         Class.forName(properties.getProperty("driver",
true));

         pool = new GenericObjectPool();
         pool.setMinIdle(5);
         pool.setMaxActive(15);
         pool.setTestOnBorrow(true);

         ConnectionFactory connectionFactory = new
DriverManagerConnectionFactory(
               properties.getProperty("url", true),
               properties.getProperty("username", true),
               properties.getProperty("password", true));

         @SuppressWarnings("all")
         PoolableConnectionFactory
poolableConnectionFactory = new
PoolableConnectionFactory(
               connectionFactory, pool, null, "SELECT 1;",
false, true);

         ds = new PoolingDataSource(pool);

         if (pool == null)
            System.out.println("pool null");
         if (ds == null)
            System.out.println("ds null");

      } catch (ClassNotFoundException e) {
         logger.error("The database driver could not be
found.");
      }
   }

   public static ConnectionPool getInstance() {
      if (instance == null) {
         synchronized (ConnectionPool.class) {
            if (instance == null) {
               instance = new ConnectionPool();
            }
         }
      }
      return instance;
   }

   public Connection getConnection() throws
SQLException {
      return ds.getConnection();
   }

}
```

DAOConfigurationException.java

```
package dao;

public class DAOConfigurationException extends
RuntimeException {

   private static final long serialVersionUID = 1L;

   public DAOConfigurationException(String message) {
      super(message);
   }

   public DAOConfigurationException(Throwable cause) {
      super(cause);
   }

   public DAOConfigurationException(String message,
Throwable cause) {
      super(message, cause);
   }

}
```

DAOException.java

```
package dao;

public class DAOException extends Exception {

   private static final long serialVersionUID = 1L;

   public DAOException(String message) {
      super(message);
   }

   public DAOException(Throwable cause) {
      super(cause);
   }

   public DAOException(String message, Throwable
cause) {
      super(message, cause);
   }

}
```

DAOFactory.java

```
package dao;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;


public abstract class DAOFactory {

   private static final String PROPERTY_URL = "url";
```

```
    private static final String PROPERTY_DRIVER =
"driver";
    private static final String PROPERTY_USERNAME =
"username";
    private static final String PROPERTY_PASSWORD =
"password";
    private static final String JNDI_ROOT =
"java:comp/env/";

    public static DAOFactory getInstance(String name)
throws DAOConfigurationException {
        if (name == null) {
            throw new DAOConfigurationException("Database
name is null.");
        }

        DAOProperties properties = new
DAOProperties(name);
        String url = properties.getProperty(PROPERTY_URL,
true);
        String driverClassName =
properties.getProperty(PROPERTY_DRIVER, false);
        String password =
properties.getProperty(PROPERTY_PASSWORD, false);
        String username =
properties.getProperty(PROPERTY_USERNAME,
password != null);
        DAOFactory instance;

        if (driverClassName != null) {
            try {
                Class.forName(driverClassName);
            } catch (ClassNotFoundException e) {
                throw new DAOConfigurationException(
                    "Driver class '" + driverClassName + "' is
missing in classpath.", e);
            }
            instance = new DriverManagerDAOFactory(url,
username, password);
        }

        else {
            DataSource dataSource;
            try {
                dataSource = (DataSource) new
InitialContext().lookup(JNDI_ROOT + url);
            } catch (NamingException e) {
                throw new DAOConfigurationException(
                    "DataSource '" + url + "' is missing in JNDI.",
e);
            }
            if (username != null) {
                instance = new
DataSourceWithLoginDAOFactory(dataSource,  username,
password);
            } else {
                instance = new
DataSourceDAOFactory(dataSource);
            }
        }

        return instance;
    }

    abstract Connection getConnection() throws
SQLException;

    public UserDAO getUserDAO() {
        return new UserDAO(this);
    }

    public UserRoleDAO getUserRoleDAO() {
    return new UserRoleDAO(this);
    }
}

class DriverManagerDAOFactory extends DAOFactory {
    private String url;
    private String username;
    private String password;

    DriverManagerDAOFactory(String url, String username,
String password) {
        this.url = url;
        this.username = username;
        this.password = password;
    }

    Connection getConnection() throws SQLException {
        return DriverManager.getConnection(url, username,
password);
    }
}

class DataSourceDAOFactory extends DAOFactory {
    private DataSource dataSource;

    DataSourceDAOFactory(DataSource dataSource) {
        this.dataSource = dataSource;
    }

    Connection getConnection() throws SQLException {
        return dataSource.getConnection();
    }
}

class DataSourceWithLoginDAOFactory extends
DAOFactory {
    private DataSource dataSource;
    private String username;
    private String password;

    DataSourceWithLoginDAOFactory(DataSource
dataSource, String username, String password) {
        this.dataSource = dataSource;
        this.username = username;
        this.password = password;
    }

    Connection getConnection() throws SQLException {
        return dataSource.getConnection(username,
password);
    }
}
```

DAOProperties.java

```
package dao;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

public class DAOProperties {

    private static final String PROPERTIES_FILE =
"dao.properties";
    private static final Properties PROPERTIES = new
Properties();

    static {
        ClassLoader classLoader =
Thread.currentThread().getContextClassLoader();
        InputStream propertiesFile =
classLoader.getResourceAsStream(PROPERTIES_FILE);

        if (propertiesFile == null) {
            throw new DAOConfigurationException(
```

```
            "Properties file '" + PROPERTIES_FILE + "' is
missing in classpath.");
        }

        try {
            PROPERTIES.load(propertiesFile);
        } catch (IOException e) {
            throw new DAOConfigurationException(
                "Cannot load properties file '" +
PROPERTIES_FILE + "'.", e);
        }
    }

    private String specificKey;

    public DAOProperties(String specificKey) throws
DAOConfigurationException {
        this.specificKey = specificKey;
    }

    public String getProperty(String key, boolean
mandatory) throws DAOConfigurationException {
        String fullKey = specificKey + "." + key;
        String property = PROPERTIES.getProperty(fullKey);

        if (property == null || property.trim().length() == 0) {
            if (mandatory) {
                throw new
DAOConfigurationException("Required property '" +
fullKey + "'"
                    + " is missing in properties file '" +
PROPERTIES_FILE + "'.");
            } else {
                property = null;
            }
        }

        return property;
    }

}
```

DAOUtil.java

```
package dao;

import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public final class DAOUtil {

    private DAOUtil() {
    }

    public static PreparedStatement prepareStatement
        (Connection connection, String sql, boolean
returnGeneratedKeys, Object... values)
            throws SQLException
    {
        PreparedStatement preparedStatement =
connection.prepareStatement(sql,
            returnGeneratedKeys ?
Statement.RETURN_GENERATED_KEYS :
Statement.NO_GENERATED_KEYS);
        setValues(preparedStatement, values);
        return preparedStatement;
    }

    public static void setValues(PreparedStatement
preparedStatement, Object... values)
        throws SQLException
    {
        for (int i = 0; i < values.length; i++) {
            preparedStatement.setObject(i + 1, values[i]);
        }
    }

    public static void close(Connection connection) {
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e) {
                System.err.println("Closing Connection failed: "
+ e.getMessage());
                e.printStackTrace();
            }
        }
    }

    public static void close(Statement statement) {
        if (statement != null) {
            try {
                statement.close();
            } catch (SQLException e) {
                System.err.println("Closing Statement failed: " +
e.getMessage());
                e.printStackTrace();
            }
        }
    }

    public static void close(ResultSet resultSet) {
        if (resultSet != null) {
            try {
                resultSet.close();
            } catch (SQLException e) {
                System.err.println("Closing ResultSet failed: " +
e.getMessage());
                e.printStackTrace();
            }
        }
    }

    public static void close(Connection connection,
Statement statement) {
        close(statement);
        close(connection);
    }

    public static void close(Connection connection,
Statement statement, ResultSet resultSet) {
        close(resultSet);
        close(statement);
        close(connection);
    }

    public static String hashMD5(String string) {
        byte[] hash;

        try {
            hash =
MessageDigest.getInstance("MD5").digest(string.getBytes
("UTF-8"));
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException("MD5 should be
supported?", e);
        } catch (UnsupportedEncodingException e) {
            throw new RuntimeException("UTF-8 should be
supported?", e);
        }

        StringBuilder hex = new StringBuilder(hash.length *
2);
```

```
        for (byte b : hash) {
            if ((b & 0xff) < 0x10) hex.append("0");
            hex.append(Integer.toHexString(b & 0xff));
        }
        return hex.toString();
    }

}
```

UserDAO.java

```
package dao;

import static dao.DAOUtil.*;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import webapp.Config;

import model.user.User;
import model.user.UserRole;


public final class UserDAO {

    private static final String SQL_FIND_BY_ID =
        "SELECT id, username, password, user_role_id,
last_name, first_name FROM user WHERE id = ?";
    private static final String
SQL_FIND_BY_USERNAME_AND_PASSWORD =
        "SELECT id, username, password, user_role_id,
last_name, first_name FROM user WHERE username = ?
AND password = ?";
    private static final String
SQL_LIST_ORDER_BY_USERNAME =
        "SELECT id, username, password, user_role_id,
last_name, first_name FROM user ORDER BY username";
    private static final String SQL_INSERT =
        "INSERT INTO user (username, password,
user_role_id, last_name, first_name) VALUES
(?, ?, ?, ?, ?)";
    private static final String SQL_UPDATE =
        "UPDATE user SET username = ?, password = ?,
user_role_id = ?, last_name = ?, first_name = ? WHERE
id = ?";
    private static final String SQL_DELETE =
        "DELETE FROM user WHERE id = ?";
    private static final String SQL_EXIST_USERNAME =
        "SELECT id FROM user WHERE username = ?";
    private static final String SQL_EXIST_EMAIL =
        "SELECT id FROM user WHERE email = ?";

    private DAOFactory daoFactory;

    UserDAO(DAOFactory daoFactory) {
        this.daoFactory = daoFactory;
    }

    public User find(Long id) throws DAOException {
        return find(SQL_FIND_BY_ID, id);
    }

    public User find(String username, String password)
throws DAOException {
        return
find(SQL_FIND_BY_USERNAME_AND_PASSWORD,
username, hashMD5(password));
    }

    private User find(String sql, Object... values) throws
DAOException {
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;
        User user = null;

        try {
            connection = daoFactory.getConnection();
            preparedStatement =
prepareStatement(connection, sql, false, values);
            resultSet = preparedStatement.executeQuery();
            if (resultSet.next()) {
                user = mapUser(resultSet);
            }
        } catch (SQLException e) {
            throw new DAOException(e);
        } finally {
            close(connection, preparedStatement, resultSet);
        }

        return user;
    }

    public List<User> list() throws DAOException {
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;
        List<User> users = new ArrayList<User>();

        try {
            connection = daoFactory.getConnection();
            preparedStatement =
connection.prepareStatement(SQL_LIST_ORDER_BY_U
SERNAME);
            resultSet = preparedStatement.executeQuery();
            while (resultSet.next()) {
                users.add(mapUser(resultSet));
            }
        } catch (SQLException e) {
            throw new DAOException(e);
        } finally {
            close(connection, preparedStatement, resultSet);
        }

        return users;
    }

    public void create(User user) throws
IllegalArgumentException, DAOException {
        if (user.getId() != null) {
            throw new IllegalArgumentException("User is
already created, the user ID is not null.");
        }

        Object[] values = {
            user.getUsername(),
            hashMD5IfNecessary(user.getPassword()),
            user.getUserRole().getId(),
            user.getLastName(),
            user.getFirstName()
        };

        Connection connection = null;
        PreparedStatement preparedStatement = null;
        ResultSet generatedKeys = null;

        try {
            connection = daoFactory.getConnection();
            preparedStatement =
prepareStatement(connection, SQL_INSERT, true,
values);
            int affectedRows =
preparedStatement.executeUpdate();
            if (affectedRows == 0) {
```

```
            throw new DAOException("Creating user failed,
no rows affected.");
        }
        generatedKeys =
preparedStatement.getGeneratedKeys();
        if (generatedKeys.next()) {
            user.setId(generatedKeys.getLong(1));
        } else {
            throw new DAOException("Creating user failed,
no generated key obtained.");
        }
    } catch (SQLException e) {
        throw new DAOException(e);
    } finally {
        close(connection, preparedStatement,
generatedKeys);
    }
}

public void update(User user) throws DAOException {
    if (user.getId() == null) {
        throw new IllegalArgumentException("User is not
created yet, the user ID is null.");
    }

    Object[] values = {
        user.getUsername(),
        hashMD5IfNecessary(user.getPassword()),
        user.getUserRole().getId(),
        user.getLastName(),
        user.getFirstName(),
        user.getId()
    };

    Connection connection = null;
    PreparedStatement preparedStatement = null;

    try {
        connection = daoFactory.getConnection();
        preparedStatement =
prepareStatement(connection, SQL_UPDATE, false,
values);
        int affectedRows =
preparedStatement.executeUpdate();
        if (affectedRows == 0) {
            throw new DAOException("Updating user failed,
no rows affected.");
        }
    } catch (SQLException e) {
        throw new DAOException(e);
    } finally {
        close(connection, preparedStatement);
    }
}

public void save(User user) throws DAOException {
    if (user.getId() == null) {
        create(user);
    } else {
        update(user);
    }
}

public void delete(User user) throws DAOException {
    Object[] values = { user.getId() };

    Connection connection = null;
    PreparedStatement preparedStatement = null;

    try {
        connection = daoFactory.getConnection();
        preparedStatement =
prepareStatement(connection, SQL_DELETE, false,
values);
        int affectedRows =
preparedStatement.executeUpdate();
        if (affectedRows == 0) {
            throw new DAOException("Deleting user failed,
no rows affected.");
        } else {
            user.setId(null);
        }
    } catch (SQLException e) {
        throw new DAOException(e);
    } finally {
        close(connection, preparedStatement);
    }
}

public boolean existUsername(String username) throws
DAOException {
    return exist(SQL_EXIST_USERNAME, username);
}

public boolean existEmail(String email) throws
DAOException {
    return exist(SQL_EXIST_EMAIL, email);
}

private boolean exist(String sql, Object... values) throws
DAOException {
    Connection connection = null;
    PreparedStatement preparedStatement = null;
    ResultSet resultSet = null;
    boolean exist = false;

    try {
        connection = daoFactory.getConnection();
        preparedStatement =
prepareStatement(connection, sql, false, values);
        resultSet = preparedStatement.executeQuery();
        exist = resultSet.next();
    } catch (SQLException e) {
        throw new DAOException(e);
    } finally {
        close(connection, preparedStatement, resultSet);
    }

    return exist;
}

private static String hashMD5IfNecessary(String
password) {
    return !password.matches("^[a-f0-9]{32}$") ?
hashMD5(password) : password;
}

private static User mapUser(ResultSet resultSet) throws
SQLException, DAOException {
    final UserRoleDAO userRoleDAO =
Config.getInstance().getDAOFactory().getUserRoleDAO();

    return new User(
        resultSet.getLong("id"),
        resultSet.getString("username"),
        resultSet.getString("password"),
        new UserRole(resultSet.getLong("user_role_id"),
userRoleDAO.find(resultSet.getLong("user_role_id")).getN
ame()),
        resultSet.getString("last_name"),
        resultSet.getString("first_name")
    );
}

}
```

UserRoleDAO.java

```
package dao;
```

```
import static dao.DAOUtil.close;
import static dao.DAOUtil.prepareStatement;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import model.user.UserRole;

public class UserRoleDAO {

    private static final String SQL_FIND_BY_ID =
        "SELECT id, name FROM user_role WHERE id = ?";
    private static final String SQL_LIST_ORDER_BY_ID =
        "SELECT id, name FROM user_role ORDER BY id";

    private DAOFactory daoFactory;

    UserRoleDAO(DAOFactory daoFactory) {
        this.daoFactory = daoFactory;
    }

    public UserRole find(Long id) throws DAOException {
        return find(SQL_FIND_BY_ID, id);
    }

    private UserRole find(String sql, Object... values) throws
DAOException {
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;
        UserRole userRole = null;

        try {
            connection = daoFactory.getConnection();
            preparedStatement =
prepareStatement(connection, sql, false, values);
            resultSet = preparedStatement.executeQuery();
            if (resultSet.next()) {
                userRole = mapUserRole(resultSet);
            }
        } catch (SQLException e) {
            throw new DAOException(e);
        } finally {
            close(connection, preparedStatement, resultSet);
        }

        return userRole;
    }

    public List<UserRole> list() throws DAOException {
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;
        List<UserRole> userRoles = new
ArrayList<UserRole>();

        try {
            connection = daoFactory.getConnection();
            preparedStatement =
connection.prepareStatement(SQL_LIST_ORDER_BY_ID
);
            resultSet = preparedStatement.executeQuery();
            while (resultSet.next()) {
                userRoles.add(mapUserRole(resultSet));
            }
        } catch (SQLException e) {
            throw new DAOException(e);
        } finally {
            close(connection, preparedStatement, resultSet);
        }
        return userRoles;
    }

    private static UserRole mapUserRole(ResultSet
resultSet) throws SQLException {
    return new UserRole(
            resultSet.getLong("id"),
            resultSet.getString("name")
        );
    }

}
```

AbstractForm.java

```
package model;

import java.text.MessageFormat;
import java.util.ResourceBundle;

import javax.faces.application.FacesMessage;
import javax.faces.application.FacesMessage.Severity;
import javax.faces.component.UIComponent;
import javax.faces.component.UIForm;
import javax.faces.context.FacesContext;

public abstract class AbstractForm {

    public static final String EXIST_MESSAGE_ID =
"model.AbstractForm.EXIST";

    public static final String INVALID_MESSAGE_ID =
"model.AbstractForm.INVALID";

    public static final String INEQUAL_MESSAGE_ID =
"model.AbstractForm.INEQUAL";

    private static ResourceBundle messageBundle =
ResourceBundle.getBundle(

FacesContext.getCurrentInstance().getApplication().getMe
ssageBundle());

    private UIForm form;
    private boolean success;

    public UIForm getForm() {
        return form;
    }

    public boolean isSuccess() {
        return success;
    }

    public void setForm(UIForm form) {
        this.form = form;
    }

    public void setSuccess(boolean success) {
        this.success = success;
    }

    public String getMessage(String messageId, Object...
params) {
        return
MessageFormat.format(messageBundle.getString(messag
eId), params);
    }

    public void setSuccessMessage(String message) {
        setSuccess(true);
        addMessage(form, message,
FacesMessage.SEVERITY_INFO);
    }
```

```
    public void setErrorMessage(String message) {
        addMessage(form, message,
FacesMessage.SEVERITY_ERROR);
    }

    public void setErrorMessage(Exception exception) {
        String message =
getMessage(exception.getClass().getName(),
exception.getMessage());
        addMessage(form, message,
FacesMessage.SEVERITY_ERROR);
        exception.printStackTrace(); // Or use logger.
    }

    public void setErrorMessage(UIComponent component,
String message) {
        addMessage(component, message,
FacesMessage.SEVERITY_ERROR);
    }

    public static void addMessage(UIComponent
component, String message, Severity severity) {
        FacesContext facesContext =
FacesContext.getCurrentInstance();

facesContext.addMessage(component.getClientId(facesC
ontext),
            new FacesMessage(severity, message, null));
    }

    public static String getLabel(UIComponent component)
{
        return (String) component.getAttributes().get("label");
    }

}
```

ExistsValidator.java

```
package model;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

import com.hp.hpl.jena.ontology.OntResource;

import ontology.Ontology;
import ontology.OntologyInstance;

public class ExistsValidator implements Validator {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    @Override
    public void validate(FacesContext context,
UIComponent component, Object value)
            throws ValidatorException {

        String name = (String) value;
        OntResource ontResource =
onto.getOntResource(name.trim(), true);

        if (ontResource != null) {
            throw new ValidatorException(new
FacesMessage("A resource with the same name already
exists."));
        }
    }

}
```

OntologyBean.java

```
package model;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import javax.faces.bean.ManagedBean;
import javax.faces.context.ExternalContext;
import javax.faces.context.FacesContext;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.io.FilenameUtils;
import org.apache.commons.io.IOUtils;
import
org.apache.myfaces.custom.fileupload.UploadedFile;

import ontology.Ontology;
import ontology.OntologyInstance;

@ManagedBean
public class OntologyBean {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private UploadedFile uploadedFile;

    public void actionImport() {
        String prefix =
FilenameUtils.getBaseName(uploadedFile.getName());
        String suffix =
FilenameUtils.getExtension(uploadedFile.getName());

        File file = null;
        OutputStream output = null;

        try {
            file = File.createTempFile(prefix + "_", "." + suffix,
                new
File(FacesContext.getCurrentInstance().getExternalConte
xt().getRealPath("/")));
            output = new FileOutputStream(file);
            IOUtils.copy(uploadedFile.getInputStream(),
output);
        } catch (IOException e) {
            if (file != null)
                file.delete();

            e.printStackTrace();
        } finally {
            IOUtils.closeQuietly(output);
        }

        onto.importOntology(file.getAbsolutePath());
    }

    public void actionExport() throws IOException {
        FacesContext facesContext =
FacesContext.getCurrentInstance();
        ExternalContext externalContext =
facesContext.getExternalContext();
        HttpServletResponse response =
(HttpServletResponse) externalContext.getResponse();

        response.reset();
        response.setContentType("application/xml");
        response.setHeader("Content-disposition",
"attachment; filename=\"NPO.owl\"");

        BufferedOutputStream output = null;
```

```
        try {
            output = new
BufferedOutputStream(response.getOutputStream());
            onto.exportOntology(output);
        } finally {
        output.close();
        }

        facesContext.responseComplete();
    }

    public UploadedFile getUploadedFile() {
        return uploadedFile;
    }

    public void setUploadedFile(UploadedFile uploadedFile)
{
        this.uploadedFile = uploadedFile;
    }

}
```

OntologyUploadValidator.java

```
package model;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

import
org.apache.myfaces.custom.fileupload.UploadedFile;

public class OntologyUploadValidator implements
Validator {

    @Override
    public void validate(FacesContext context,
UIComponent component, Object value)
            throws ValidatorException {

        UploadedFile uploadedFile = (UploadedFile) value;
        String filename = uploadedFile.getName();

        if (!filename.endsWith(".owl")
&& !filename.endsWith(".OWL")) {
            throw new ValidatorException(new
FacesMessage("Error: Unsupported file type, please
select OWL only."));
        }
    }

}
```

PasswordValidator.java

```
package model;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

public class PasswordValidator implements Validator {

    public void validate(FacesContext context,
UIComponent component, Object value)
        throws ValidatorException
    {
        String password = (String)
component.getAttributes().get("password");
        String confirm = (String) value;

        if (!password.equals(confirm)) {
            throw new ValidatorException(new
FacesMessage("Passwords do not match."));
        }

    }

}
```

StringDouble.java

```
package model;

public class StringDouble {

    private String string1;
    private String string2;

    public StringDouble() {

    }

    public StringDouble(String string1, String string2) {
        this.string1 = string1;
        this.string2 = string2;
    }

    public String getString1() {
        return string1;
    }

    public String getString2() {
        return string2;
    }

    public void setString1(String string1) {
        this.string1 = string1;
    }

    public void setString2(String string2) {
        this.string2 = string2;
    }

}
```

StringSingle.java

```
package model;

public class StringSingle {

    private String string;

    public StringSingle() {

    }

    public StringSingle(String string) {
        this.string = string;
    }

    public String getString() {
        return string;
    }

    public void setString(String string) {
        this.string = string;
    }

}
```

StringTriple.java

```
package model;

public class StringTriple {

    private String string1;
    private String string2;
    private String string3;

    public StringTriple() {

    }

    public StringTriple(String string1, String string2, String
string3) {
        this.string1 = string1;
        this.string2 = string2;
        this.string3 = string3;
    }

    public String getString1() {
        return string1;
    }

    public String getString2() {
        return string2;
    }

    public String getString3() {
        return string3;
    }

    public void setString1(String string1) {
        this.string1 = string1;
    }

    public void setString2(String string2) {
        this.string2 = string2;
    }

    public void setString3(String string3) {
        this.string3 = string3;
    }

}
```

StringUtil.java

```
package model;

import java.util.ArrayList;
import java.util.List;

public class StringUtil {

    public static List<StringSingle>
convertToStringSingleList(List<String> strings) {
        List<StringSingle> ss = new
ArrayList<StringSingle>();
        for (String string : strings) {
            ss.add(new StringSingle(string));
        }
        return ss;
    }

    public static List<StringDouble>
convertToStringDoubleList(List<String> strings1,
            List<String> strings2) {
        List<StringDouble> sd = new
ArrayList<StringDouble>();
        for (int i = 0; i < strings1.size(); i++) {
            sd.add(new StringDouble(strings1.get(i),
strings2.get(i)));
        }
        return sd;
    }

    public static List<StringTriple>
convertToStringTripleList(List<String> strings1,
            List<String> strings2, List<String> strings3) {
        List<StringTriple> st = new ArrayList<StringTriple>();
        for (int i = 0; i < strings1.size(); i++) {
            st.add(new StringTriple(strings1.get(i),
strings2.get(i), strings3.get(i)));
        }
        return st;
    }

}
```

TaxonomyRenameForm.java

```
package model;

import javax.faces.bean.ManagedBean;

import ontology.Ontology;
import ontology.OntologyInstance;

@ManagedBean
public class TaxonomyRenameForm extends
AbstractForm {

    protected Ontology onto =
OntologyInstance.getInstance().getOntology();

    protected String oldName;
    protected String newName;

    public String actionEdit() {
        onto.renameResource(oldName, newName);
        onto.commit();

        return "edit-success";
    }

    public String getOldName() {
        return oldName;
    }

    public String getNewName() {
        return newName;
    }

    public void setOldName(String oldName) {
        this.oldName = oldName;
        this.newName = oldName.replace("_", " ");
    }

    public void setNewName(String newName) {
        this.newName = newName;
    }

}
```

TaxonomyUtil.java

```
package model;

import java.util.List;

import ontology.Ontology;

import org.primefaces.model.DefaultTreeNode;
import org.primefaces.model.TreeNode;

public class TaxonomyUtil {

    public static void getSubClassesOf(Ontology onto,
String className, TreeNode parent) {
```

```
        List<String> classes =
onto.listSubClasses(className);
        for (String aClass : classes) {
            TreeNode node = new
DefaultTreeNode(aClass.replace("_", " "), parent);
            getSubClassesOf(onto, aClass, node);
        }
    }

}
```

Util.java

```
package model;

import java.util.List;

public class Util {

    public static List<String> reverseOrder(List<String> list)
{
        int size = list.size();
        for (int i = 0; i < Math.floor(size / 2); i++) {
            int mirror = size - 1 - i;
            String temp = list.get(mirror);
            list.set(mirror, list.get(i));
            list.set(i, temp);
        }

        return list;
    }

}
```

Config.java

```
package model.image;

import java.util.ResourceBundle;

public class Config {
    private static Config config;
    private ResourceBundle rb;
    private static final String SOURCE =
"model/image/config";

    private Config() {
    }

    public static Config getInstance() {
        if (config == null) {
            config = new Config();
            config.rb = ResourceBundle.getBundle(SOURCE);
        }
        return config;
    }

    public String getValue(String key) {
        return rb.getString(key);
    }

    public void setTemplate(String template) {

    }
}
```

ImageBean.java

```
package model.image;

import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;
import javax.faces.bean.ManagedBean;
import javax.imageio.ImageIO;

import model.plant.Plant;
import ontology.Consts;
import ontology.Ontology;
import ontology.OntologyInstance;

import org.apache.commons.io.FilenameUtils;
import org.apache.commons.io.IOUtils;
import
org.apache.myfaces.custom.fileupload.UploadedFile;

import com.hp.hpl.jena.ontology.OntResource;
import com.sun.image.codec.jpeg.JPEGCodec;
import com.sun.image.codec.jpeg.JPEGImageEncoder;
import com.thebuzzmedia.imgscalr.Scalr;

@ManagedBean
public class ImageBean {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private String currentResource;
    private UploadedFile uploadedFile;
    private String source;

    private Plant plant = new Plant();

    public String actionUpload() {

        String webappDir =
Config.getInstance().getValue("webapp.directory");
        String uploadsDir =
Config.getInstance().getValue("uploads.directory");
        String suffix =
FilenameUtils.getExtension(uploadedFile.getName());

        File file = null;
        OutputStream output = null;

        try {
            file = File.createTempFile(currentResource + "_",
"." + suffix,
                    new File(webappDir + uploadsDir + "/plants"));
            output = new FileOutputStream(file);
            IOUtils.copy(uploadedFile.getInputStream(),
output);

            BufferedImage srcImage = ImageIO.read(new
File(file.getAbsolutePath()));
            BufferedImage scaledImage =
Scalr.resize(srcImage, 150);

            OutputStream os = new FileOutputStream(
                    webappDir + uploadsDir + "/plants/thumbs/" +
file.getName());
            JPEGImageEncoder encoder =
JPEGCodec.createJPEGEncoder(os);
            encoder.encode(scaledImage);
            os.close();

        OntResource individual =
onto.getOntResource(currentResource);


    individual.addLiteral(onto.getProperty(Consts.IMAGE),
file.getName());

    individual.addLiteral(onto.getProperty(Consts.IMAGE_S
OURCE), file.getName() + source);

        } catch (IOException e) {
```

```
            if (file != null)
                file.delete();

            e.printStackTrace();
        } finally {
            IOUtils.closeQuietly(output);
        }

        return "show?faces-redirect=true&name=" +
currentResource;
    }

    public ImageBean() {
        loadPlant();
    }

    public String getCurrentResource() {
        return currentResource;
    }

    public UploadedFile getUploadedFile() {
        return uploadedFile;
    }

    public String getSource() {
        return source;
    }

    public Plant getPlant() {
        if (plant == null || plant.getOntologyLocalName() ==
null

    || !plant.getOntologyLocalName().equals(currentResour
ce)) {
            loadPlant();
        }

        return plant;
    }

   public void setCurrentResource(String currentResource)
{
        this.currentResource = currentResource;
    }

    public void setUploadedFile(UploadedFile uploadedFile)
{
        this.uploadedFile = uploadedFile;
    }

    public void setSource(String source) {
        this.source = source;
    }

    private void loadPlant() {
        if (currentResource != null
&& !currentResource.equals("")) {
            OntResource ontResource =
onto.getOntResource(currentResource);
            plant = new Plant(currentResource,
                    onto.getPropertyValue(ontResource,
Consts.SCIENTIFIC_NAME));
        }
    }

}
```

ImageServlet.java

```
package model.image;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.Closeable;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.URLDecoder;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ImageServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    private static final int DEFAULT_BUFFER_SIZE =
10240;

    private String imagePath;

    public void init() throws ServletException {

        String webappDir =
Config.getInstance().getValue("webapp.directory");
        String uploadsDir =
Config.getInstance().getValue("uploads.directory");
        this.imagePath = webappDir + uploadsDir;

    }

    protected void doGet(HttpServletRequest request,
HttpServletResponse response)
        throws ServletException, IOException
    {

        String requestedImage = request.getPathInfo();

        if (requestedImage == null) {
response.sendError(HttpServletResponse.SC_NOT_FOU
ND);
            return;
        }

        File image = new File(imagePath,
URLDecoder.decode(requestedImage, "UTF-8"));

        if (!image.exists()) {

response.sendError(HttpServletResponse.SC_NOT_FOU
ND);
            return;
        }

        String contentType =
getServletContext().getMimeType(image.getName());

        if (contentType == null
|| !contentType.startsWith("image")) {

response.sendError(HttpServletResponse.SC_NOT_FOU
ND);
            return;
        }

        response.reset();
        response.setBufferSize(DEFAULT_BUFFER_SIZE);
        response.setContentType(contentType);
        response.setHeader("Content-Length",
String.valueOf(image.length()));
        response.setHeader("Content-Disposition", "inline;
filename=\"" + image.getName() + "\"");

        BufferedInputStream input = null;
        BufferedOutputStream output = null;

        try {
```

```
            input = new BufferedInputStream(new
FileInputStream(image), DEFAULT_BUFFER_SIZE);
            output = new
BufferedOutputStream(response.getOutputStream(),
DEFAULT_BUFFER_SIZE);

            byte[] buffer = new
byte[DEFAULT_BUFFER_SIZE];
            int length;
            while ((length = input.read(buffer)) > 0) {
                output.write(buffer, 0, length);
            }
        } finally {
            close(output);
            close(input);
        }
    }

    private static void close(Closeable resource) {
        if (resource != null) {
            try {
                resource.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

}
```

ImageUploadValidator.java

```
package model.image;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.validator.Validator;
import javax.faces.validator.ValidatorException;

import
org.apache.myfaces.custom.fileupload.UploadedFile;

public class ImageUploadValidator implements Validator {

    @Override
    public void validate(FacesContext context,
UIComponent component, Object value)
            throws ValidatorException {

        UploadedFile uploadedFile = (UploadedFile) value;
        String filename = uploadedFile.getName();

        if (!filename.endsWith(".jpg")
&& !filename.endsWith(".JPG")
                && !filename.endsWith(".jpeg")
&& !filename.endsWith(".JPEG")
                && !filename.endsWith(".gif")
&& !filename.endsWith(".GIF")
                && !filename.endsWith(".png")
&& !filename.endsWith(".PNG")) {
            throw new ValidatorException(new
FacesMessage("Error: Unsupported image type, please
select either JPG, GIF or PNG only."));
        }

    }

}
```

ImageUtil.java

```
package model.image;

import java.io.File;

public class ImageUtil {

    public static void deleteImage(String filename) {
        String webappDir =
Config.getInstance().getValue("webapp.directory");
        String uploadsDir =
Config.getInstance().getValue("uploads.directory");

        File file = new File(webappDir + uploadsDir +
"/plants/" + filename);
        file.delete();

        file = new File(webappDir + uploadsDir +
"/plants/thumbs/" + filename);
        file.delete();
    }

}
```

config.properties

```
webapp.directory = /home/cjlim/chelcy/webapps
uploads.directory = /uploads
```

Location.java

```
package model.location;

public class Location {

    private String coordinates;
    private String description;

    public Location() {

    }

    public Location(String coordinates, String description) {
        this.coordinates = coordinates;
        this.description = description;
    }

    public String getCoordinates() {
        return coordinates;
    }

    public String getDescription() {
        return description;
    }

    public String getLatitude() {
        return coordinates.split(",")[0];
    }

    public String getLongitude() {
        return coordinates.split(",")[1];
    }

    public void setCoordinates(String coordinates) {
        this.coordinates = coordinates;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public void setLatitude(String latitude) {
        this.coordinates = latitude + "," + this.getLongitude();
    }

    public void setLongitude(String longitude) {
        this.coordinates = this.getLatitude() + "," + longitude;
    }
```

```
}
```

LocationBean.java

```
package model.location;

import javax.faces.bean.ManagedBean;

import model.plant.Plant;
import ontology.Consts;
import ontology.Ontology;
import ontology.OntologyInstance;

import com.hp.hpl.jena.ontology.OntResource;

@ManagedBean
public class LocationBean {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private String currentResource;
    private String latitude;
    private String longitude;
    private String description;

    private Plant plant = new Plant();

    public String actionAdd() {
        String coordinates = latitude + "," + longitude;

        OntResource individual =
onto.getOntResource(currentResource);

    individual.addLiteral(onto.getProperty(Consts.LOCATIO
N_COORDINATES), coordinates);

    individual.addLiteral(onto.getProperty(Consts.LOCATIO
N_STRING),
                coordinates + description);

        return "show?faces-redirect=true&name=" +
currentResource;
    }

    public LocationBean() {
        loadPlant();
    }

    public String getCurrentResource() {
        return currentResource;
    }

    public String getLatitude() {
        return latitude;
    }

    public String getLongitude() {
        return longitude;
    }

    public String getDescription() {
        return description;
    }

    public Plant getPlant() {
        if (plant == null || plant.getOntologyLocalName() ==
null

    || !plant.getOntologyLocalName().equals(currentResour
ce)) {
            loadPlant();
        }

        return plant;
    }

   public void setCurrentResource(String currentResource)
{
        this.currentResource = currentResource;
    }

    public void setLatitude(String latitude) {
        this.latitude = latitude;
    }

    public void setLongitude(String longitude) {
        this.longitude = longitude;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    private void loadPlant() {
        if (currentResource != null
&& !currentResource.equals("")) {
            OntResource ontResource =
onto.getOntResource(currentResource);
            plant = new Plant(currentResource,
                    onto.getPropertyValue(ontResource,
Consts.SCIENTIFIC_NAME));
        }
    }

}
```

Phytochemical.java

```
package model.phytochemical;

import java.util.List;

import model.StringSingle;

public class Phytochemical {

    private String ontologyLocalName;
    private String name;
    private String kegg;
    private String chebi;
    private String formula;
    private String molecularWeight;
    private String iupacName;
    private List<StringSingle> synonym;

    public Phytochemical() {

    }

    public Phytochemical(String ontologyLocalName, String
name) {
        this.setOntologyLocalName(ontologyLocalName);
        this.setName(name);
    }

    @Override
    public boolean equals(Object other) {
        if (this == other)
            return true;

        if (!(other instanceof Phytochemical))
            return false;

       Phytochemical otherPhytochemical = (Phytochemical)
other;
        if
(this.getOntologyLocalName().equals(otherPhytochemical.
getOntologyLocalName()))
            return true;
```

```
        else
            return false;
    }

    public String getOntologyLocalName() {
        return ontologyLocalName;
    }

    public String getName() {
        return name;
    }

    public String getKegg() {
        return kegg;
    }

    public String getChebi() {
        return chebi;
    }

    public String getFormula() {
        return formula;
    }

    public String getMolecularWeight() {
        return molecularWeight;
    }

    public String getIupacName() {
        return iupacName;
    }

    public List<StringSingle> getSynonym() {
        return synonym;
    }

    public void setOntologyLocalName(String
ontologyLocalName) {
        this.ontologyLocalName = ontologyLocalName;
    }

    public void setName(String name) {
        this.name = name;
    }

    public void setKegg(String kegg) {
        this.kegg = kegg;
    }

    public void setChebi(String chebi) {
        this.chebi = chebi;
    }

    public void setFormula(String formula) {
        this.formula = formula;
    }

    public void setMolecularWeight(String molecularWeight)
{
        this.molecularWeight = molecularWeight;
    }

    public void setIupacName(String iupacName) {
        this.iupacName = iupacName;
    }

    public void setSynonym(List<StringSingle> synonym) {
        this.synonym = synonym;
    }

}
```

PhytochemicalBean.java

```
package model.phytochemical;

import java.util.ArrayList;
import java.util.List;

import javax.faces.bean.ManagedBean;

import model.StringDouble;
import model.StringUtil;
import model.Util;
import model.plant.Plant;
import ontology.Config;
import ontology.Consts;
import ontology.Ontology;
import ontology.OntologyInstance;
import ontology.OntologyUtil;

import com.hp.hpl.jena.ontology.OntResource;
import com.hp.hpl.jena.rdf.model.StmtIterator;

@ManagedBean
public class PhytochemicalBean {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private String currentResource;
    private List<StringDouble> dataList;
    private List<String> hierarchy;

    public List<Plant> getPlants() {
        List<Plant> plants = new ArrayList<Plant>();

        StmtIterator si =
onto.searchNotLiteral(Consts.PHYTOCHEMICAL_FOUND,
currentResource);
        while (si.hasNext()) {
            OntResource r =
onto.getOntResource(si.next().getSubject().getLocalName
());

            Plant plant = new Plant(r.getLocalName(),
                    onto.getPropertyValue(r,
Consts.SCIENTIFIC_NAME));

            List<String> localName = onto.listPropertyValues(r,
Consts.LOCAL_NAME);

    plant.setLocalName(StringUtil.convertToStringSingleLis
t(localName));

            List<String> image = onto.listPropertyValues(r,
Consts.IMAGE);
            List<String> imageSource =
onto.listPropertyValues(r, Consts.IMAGE_SOURCE);

    plant.setImage(StringUtil.convertToStringDoubleList(im
age, imageSource));

            plants.add(plant);
        }

        return plants;
    }

    public String getCurrentResource() {
        return currentResource;
    }

    public List<StringDouble> getDataList() {
        if (dataList == null) {
            loadDataList();
        }

        return dataList;
    }
```

```
    public List<String> getHierarchy() {
        if (hierarchy == null) {
            loadHierarchy();
        }

        return hierarchy;
    }

   public void setCurrentResource(String currentResource)
{
        this.currentResource = currentResource;
    }

    private void loadDataList() {
        OntResource ontResource =
onto.getOntResource(currentResource);

        if (ontResource == null)
            return;

        String[] fields =
OntologyUtil.getFields("phytochemical");

        dataList = new ArrayList<StringDouble>();

        for (String field : fields) {
            List<String> values =
onto.listPropertyValues(ontResource, field);
            String valueString = "";

            if (values.size() == 1) {

                valueString = values.get(0);

                if (field.equals(Consts.KEGG)) {
                    if (!valueString.equals(""))
                        valueString = "<a href=\"" +
Config.getInstance().getValue("kegg.url")
                            + valueString + "\" target=\"_blank\">" +
valueString + "</a>";
                }

                if (field.equals(Consts.CHEBI)) {
                    if (!valueString.equals(""))
                        valueString = "<a href=\"" +
Config.getInstance().getValue("chebi.url")
                            + valueString + "\"
target=\"_blank\">CHEBI:" + valueString + "</a>";
                }
            } else {
                valueString = "<ul>";

                for (String value : values) {
                    valueString += "<li>" + value + "</li>";
                }

                valueString += "</ul>";
            }

            dataList.add(new StringDouble(field.replace("_", "
"), valueString));
        }
    }

    private void loadHierarchy() {
        hierarchy =
Util.reverseOrder(onto.getHierarchy(currentResource,
"Phytochemical"));
    }

}
```

PhytochemicalComparator.java

```
package model.phytochemical;

import java.util.Comparator;

public class PhytochemicalComparator implements
Comparator<Phytochemical> {

    @Override
   public int compare(Phytochemical p1, Phytochemical p2)
{
        return
p1.getName().compareToIgnoreCase(p2.getName());
    }

}
```

PhytochemicalForm.java

```
package model.phytochemical;

import java.util.List;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.event.ActionEvent;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;

import model.AbstractForm;
import model.StringSingle;
import model.StringUtil;
import ontology.Consts;
import ontology.Ontology;
import ontology.OntologyInstance;

import com.hp.hpl.jena.ontology.Individual;
import com.hp.hpl.jena.ontology.OntResource;

@ManagedBean
@ViewScoped
public class PhytochemicalForm extends AbstractForm {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private String currentResource;
    private String parentClass;
    private Phytochemical phytochemical = new
Phytochemical();

    private transient DataModel<StringSingle>
synonymModel;

    public PhytochemicalForm() {
        loadPhytochemical();
    }

    public void loadPhytochemical() {
        if (currentResource != null
&& !currentResource.equals("")) {
            OntResource ontResource =
onto.getOntResource(currentResource);
            phytochemical = new
Phytochemical(currentResource,
                    onto.getPropertyValue(ontResource,
Consts.PHYTOCHEMICAL_NAME));

            String kegg = onto.getPropertyValue(ontResource,
Consts.KEGG);
            phytochemical.setKegg(kegg);

           String chebi = onto.getPropertyValue(ontResource,
Consts.CHEBI);
            phytochemical.setChebi(chebi);
```

```
            String formula =
onto.getPropertyValue(ontResource, Consts.FORMULA);
            phytochemical.setFormula(formula);

            String molecularWeight =
onto.getPropertyValue(ontResource,
Consts.MOLECULAR_WEIGHT);

    phytochemical.setMolecularWeight(molecularWeight);

            String iupacName =
onto.getPropertyValue(ontResource,
Consts.IUPAC_NAME);
            phytochemical.setIupacName(iupacName);

            List<String> synonym =
onto.listPropertyValues(ontResource, Consts.SYNONYM);

    phytochemical.setSynonym(StringUtil.convertToStringSi
ngleList(synonym));
        }
    }

    public String actionAdd() {
        Individual individual =
onto.createIndividual(phytochemical.getName(),
parentClass);

    individual.addLiteral(onto.getProperty(Consts.PHYTOC
HEMICAL_NAME),
                phytochemical.getName());
        individual.addLiteral(onto.getProperty(Consts.KEGG),
phytochemical.getKegg());
        individual.addLiteral(onto.getProperty(Consts.CHEBI),
phytochemical.getChebi());

    individual.addLiteral(onto.getProperty(Consts.FORMUL
A), phytochemical.getFormula());

    individual.addLiteral(onto.getProperty(Consts.MOLECU
LAR_WEIGHT),
                phytochemical.getMolecularWeight());

    individual.addLiteral(onto.getProperty(Consts.IUPAC_N
AME), phytochemical.getIupacName());

        onto.commit();

        return "show?faces-redirect=true&name=" +
individual.getLocalName();
    }

    public String actionEdit() {
        OntResource individual =
onto.getOntResource(phytochemical.getOntologyLocalNa
me());


    individual.removeAll(onto.getProperty(Consts.KEGG));
        individual.addLiteral(onto.getProperty(Consts.KEGG),
phytochemical.getKegg().trim());


    individual.removeAll(onto.getProperty(Consts.CHEBI));
        individual.addLiteral(onto.getProperty(Consts.CHEBI),
phytochemical.getChebi().trim());

    individual.removeAll(onto.getProperty(Consts.FORMUL
A));
    individual.addLiteral(onto.getProperty(Consts.FORMUL
A), phytochemical.getFormula().trim());

    individual.removeAll(onto.getProperty(Consts.MOLECU
LAR_WEIGHT));
    individual.addLiteral(onto.getProperty(Consts.MOLECU
LAR_WEIGHT),
                phytochemical.getMolecularWeight().trim());

    individual.removeAll(onto.getProperty(Consts.IUPAC_N
AME));
    individual.addLiteral(onto.getProperty(Consts.IUPAC_N
AME), phytochemical.getIupacName());

    individual.removeAll(onto.getProperty(Consts.SYNONY
M));
        for (StringSingle ss : phytochemical.getSynonym()) {
    individual.addLiteral(onto.getProperty(Consts.SYNONY
M), ss.getString().trim());
        }

        onto.commit();

        return "show?faces-redirect=true&name=" +
phytochemical.getOntologyLocalName();
    }

    public String actionDelete() {
        onto.removeIndividual(currentResource);
        onto.commit();

        return "list?faces-redirect=true";
    }

    public void actionGetParent(ActionEvent event) {
        parentClass = (String)
event.getComponent().getAttributes().get("parentClass");
    }

    public void addNewSynonym() {
        phytochemical.getSynonym().add(new StringSingle());
    }

    public void deleteSynonym() {

    phytochemical.getSynonym().remove(synonymModel.g
etRowData());
    }

    public String getCurrentResource() {
        return currentResource;
    }

    public Phytochemical getPhytochemical() {
        if (phytochemical == null ||
phytochemical.getOntologyLocalName() == null

    || !phytochemical.getOntologyLocalName().equals(curre
ntResource)) {
            loadPhytochemical();
        }

        return phytochemical;
    }

    public DataModel<StringSingle> getSynonymModel() {
        if (synonymModel == null) {
            synonymModel = new
ListDataModel<StringSingle>(phytochemical.getSynonym(
));
        }

        return synonymModel;
    }

  public void setCurrentResource(String currentResource)
{
        this.currentResource = currentResource;
    }
```

```
}
```

PhytochemicalRenameForm.java

```
package model.phytochemical;

import javax.faces.bean.ManagedBean;

import model.TaxonomyRenameForm;
import ontology.Consts;
import ontology.Ontology;

import com.hp.hpl.jena.ontology.OntResource;

@ManagedBean
public class PhytochemicalRenameForm extends
TaxonomyRenameForm {

   @Override
   public String actionEdit() {
      onto.renameResource(oldName, newName);

      OntResource individual =
onto.getOntResource(Ontology.getValidLocalName(newN
ame));

   individual.removeAll(onto.getProperty(Consts.PHYTOC
HEMICAL_NAME));

   individual.addLiteral(onto.getProperty(Consts.PHYTOC
HEMICAL_NAME), newName);

      onto.commit();

      return "show?faces-redirect=true&name=" +
individual.getLocalName();
   }

   @Override
   public void setOldName(String oldName) {
      OntResource individual =
onto.getOntResource(oldName);
      this.oldName = oldName;
      this.newName = onto.getPropertyValue(individual,
Consts.PHYTOCHEMICAL_NAME);
   }

}
```

PhytochemicalSearchForm.java

```
package model.phytochemical;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import javax.faces.bean.ManagedBean;

import model.StringUtil;
import ontology.Consts;
import ontology.Ontology;
import ontology.OntologyInstance;

import com.hp.hpl.jena.ontology.OntResource;
import com.hp.hpl.jena.rdf.model.StmtIterator;

@ManagedBean
public class PhytochemicalSearchForm {

   private Ontology onto =
OntologyInstance.getInstance().getOntology();

   private String query;
   private List<Phytochemical> results;

   public void search() {
      results = new ArrayList<Phytochemical>();

      StmtIterator si =
onto.search(Consts.PHYTOCHEMICAL_NAME, query);
      while (si.hasNext()) {
         Phytochemical phytochemical =
convertToPhytochemical(
               si.next().getSubject().getLocalName());
         if (!results.contains(phytochemical))
            results.add(phytochemical);
      }

      si = onto.search(Consts.SYNONYM, query);
      while (si.hasNext()) {
         Phytochemical phytochemical =
convertToPhytochemical(
               si.next().getSubject().getLocalName());
         if (!results.contains(phytochemical))
            results.add(phytochemical);
      }

      si =
onto.searchResource(Consts.PHYTOCHEMICAL_NAME,
query);
      while (si.hasNext()) {
         Phytochemical phytochemical =
convertToPhytochemical(
               si.next().getSubject().getLocalName());
         if (!results.contains(phytochemical))
            results.add(phytochemical);
      }

      Collections.sort(results, new
PhytochemicalComparator());
   }

   public String getQuery() {
      return query;
   }

   public List<Phytochemical> getResults() {
      search();
      return results;
   }

   public void setQuery(String query) {
      this.query = query.trim().toLowerCase();
   }

   private Phytochemical convertToPhytochemical(String
localName) {
      OntResource ontResource =
onto.getOntResource(localName);

      Phytochemical phytochemical = new
Phytochemical(ontResource.getLocalName(),
            onto.getPropertyValue(ontResource,
Consts.PHYTOCHEMICAL_NAME));

   phytochemical.setSynonym(StringUtil.convertToStringSi
ngleList(
            onto.listPropertyValues(ontResource,
Consts.SYNONYM)));

      return phytochemical;
   }

}
```

PhytochemicalTaxonomyForm.java

```
package model.phytochemical;
```

```
import javax.faces.bean.ManagedBean;

import ontology.Ontology;
import ontology.OntologyInstance;

@ManagedBean
public class PhytochemicalTaxonomyForm {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private String name;
    private String parentClass = "Phytochemical";

    public String actionAdd() {
        onto.createClass(name, parentClass);

        return "add-success";
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

PhytochemicalTaxonomySelectBean.java

```
package model.phytochemical;

import java.util.ArrayList;
import java.util.List;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.model.SelectItem;

import ontology.Ontology;
import ontology.OntologyInstance;

@ManagedBean
@ViewScoped
public class PhytochemicalTaxonomySelectBean {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private String category;

    public List<SelectItem> getCategories() {
        List<SelectItem> groupList = new
ArrayList<SelectItem>();

        List<String> groups =
onto.listSubClasses("Phytochemical");
        for (String group : groups) {
            groupList.add(new SelectItem(group));
        }

        return groupList;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

}
```

PhytochemicalTaxonomyTreeBean.java

```
package model.phytochemical;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import javax.faces.bean.ManagedBean;

import model.StringUtil;
import model.TaxonomyUtil;
import ontology.Consts;
import ontology.Ontology;
import ontology.OntologyInstance;

import org.primefaces.model.DefaultTreeNode;
import org.primefaces.model.TreeNode;

import com.hp.hpl.jena.ontology.Individual;

@ManagedBean
public class PhytochemicalTaxonomyTreeBean {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private TreeNode root;
    private TreeNode selectedNode;
    private String selectedNodeString;
    private List<Phytochemical> phytochemicals;

    public PhytochemicalTaxonomyTreeBean() {
        selectedNodeString = "Phytochemical";

        root = new DefaultTreeNode(selectedNodeString,
null);
        TaxonomyUtil.getSubClassesOf(onto,
selectedNodeString, root);

        loadPhytochemicals();
    }

    public TreeNode getRoot() {
        return root;
    }

    public TreeNode getSelectedNode() {
        return selectedNode;
    }

    public String getSelectedNodeString() {
        return selectedNodeString;
    }

    public List<Phytochemical> getPhytochemicals() {
        return phytochemicals;
    }

    public void setSelectedNode(TreeNode selectedNode) {
        if (selectedNode != null) {
            this.selectedNode = selectedNode;
            selectedNodeString =
selectedNode.getData().toString().replace(" ", "_");
            loadPhytochemicals();
        }
    }

    private void loadPhytochemicals() {
        phytochemicals = new ArrayList<Phytochemical>();
        List<Individual> individuals =
onto.listInstances(selectedNodeString, false);

        for (Individual individual : individuals) {
```

```
            Phytochemical phytochemical = new
Phytochemical(individual.getLocalName(),
                    onto.getPropertyValue(individual,
Consts.PHYTOCHEMICAL_NAME));
            List<String> synonym =
onto.listPropertyValues(individual, Consts.SYNONYM);

    phytochemical.setSynonym(StringUtil.convertToStringSi
ngleList(synonym));
            phytochemicals.add(phytochemical);
        }

        Collections.sort(phytochemicals, new
PhytochemicalComparator());
    }

}
```

Plant.java

```
package model.plant;

import java.util.List;

import model.StringDouble;
import model.StringSingle;
import model.StringTriple;

public class Plant {

    private String ontologyLocalName;
    private String scientificName;
    private List<StringSingle> speciesSynonym;
    private List<StringSingle> localName;
    private List<StringTriple> phytochemical;
    private List<StringDouble> usablePart;
    private List<StringDouble> image;

    public Plant() {

    }

    public Plant(String ontologyLocalName, String
scientificName) {
        this.ontologyLocalName = ontologyLocalName;
        this.scientificName = scientificName;
    }

    @Override
    public boolean equals(Object other) {
        if (this == other)
            return true;

        if (!(other instanceof Plant))
            return false;

        Plant otherPlant = (Plant) other;
        if
(this.getOntologyLocalName().equals(otherPlant.getOntol
ogyLocalName()))
            return true;
        else
            return false;
    }

    public String getOntologyLocalName() {
        return ontologyLocalName;
    }

    public String getScientificName() {
        return scientificName;
    }

    public List<StringSingle> getSpeciesSynonym() {
        return speciesSynonym;
    }

    public List<StringSingle> getLocalName() {
        return localName;
    }

    public List<StringTriple> getPhytochemical() {
        return phytochemical;
    }

    public List<StringDouble> getUsablePart() {
        return usablePart;
    }

    public List<StringDouble> getImage() {
        return image;
    }

    public void setOntologyLocalName(String
ontologyLocalName) {
        this.ontologyLocalName = ontologyLocalName;
    }

    public void setScientificName(String scientificName) {
        this.scientificName = scientificName;
    }

    public void setSpeciesSynonym(List<StringSingle>
speciesSynonym) {
        this.speciesSynonym = speciesSynonym;
    }

    public void setLocalName(List<StringSingle>
localName) {
        this.localName = localName;
    }

    public void setPhytochemical(List<StringTriple>
phytochemical) {
        this.phytochemical = phytochemical;
    }

    public void setUsablePart(List<StringDouble>
usablePart) {
        this.usablePart = usablePart;
    }

    public void setImage(List<StringDouble> image) {
        this.image = image;
    }

}
```

PlantBean.java

```
package model.plant;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;

import model.StringDouble;
import model.StringUtil;
import model.Util;
import model.image.ImageUtil;
import model.location.Location;
import ontology.Consts;
import ontology.Ontology;
import ontology.OntologyInstance;
import ontology.OntologyUtil;
```

```
import com.hp.hpl.jena.ontology.OntResource;
import com.hp.hpl.jena.rdf.model.Property;

@ManagedBean
@ViewScoped
public class PlantBean {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private String currentResource;
    private OntResource ontResource;

    private List<StringDouble> dataList;
    private List<String> hierarchy;

    private List<Location> locations;
    private transient DataModel<Location> locationModel;
    private Location location = new Location();
    private String oldCoordinates;
    private boolean isEditLocation;

    private List<StringDouble> images;
    private transient DataModel<StringDouble>
imageModel;
    private StringDouble image = new StringDouble();
    private boolean isEditImage;

    public void actionEditLocation() {
        location = locationModel.getRowData();
        oldCoordinates = location.getCoordinates();
        isEditLocation = true;
    }

    public void actionSaveLocation() {
        if (ontResource == null) {
            ontResource =
onto.getOntResource(currentResource);
        }

        onto.removePairedProperty(currentResource,
Consts.LOCATION_COORDINATES,
                Consts.LOCATION_STRING, oldCoordinates);

        Property p =
onto.getProperty(Consts.LOCATION_COORDINATES);
        ontResource.addLiteral(p, location.getCoordinates());
        p = onto.getProperty(Consts.LOCATION_STRING);
        ontResource.addLiteral(p, location.getCoordinates()
+ location.getDescription());

        onto.commit();

        location = new Location();
        isEditLocation = false;
    }

    public void actionDeleteLocation() {
        Location rowData = locationModel.getRowData();

        String coordinates = rowData.getCoordinates();

        onto.removePairedProperty(currentResource,
Consts.LOCATION_COORDINATES,
                Consts.LOCATION_STRING, coordinates);
        onto.commit();

        locations.remove(rowData);
    }

    public void actionEditImage() {
        image = imageModel.getRowData();
        isEditImage = true;
    }

    public void actionSaveImage() {
        if (ontResource == null) {
            ontResource =
onto.getOntResource(currentResource);
        }

        onto.removePairedProperty(currentResource,
Consts.IMAGE, Consts.IMAGE_SOURCE,
                image.getString1());

        Property p = onto.getProperty(Consts.IMAGE);
        ontResource.addLiteral(p, image.getString1());
        p = onto.getProperty(Consts.IMAGE_SOURCE);
        ontResource.addLiteral(p, image.getString1() +
image.getString2());

        onto.commit();

        image = new StringDouble();
        isEditImage = false;
    }

    public void actionDeleteImage() {
        StringDouble rowData = imageModel.getRowData();

        onto.removePairedProperty(currentResource,
Consts.IMAGE, Consts.IMAGE_SOURCE,
                rowData.getString1());
        onto.commit();

        ImageUtil.deleteImage(rowData.getString1());

        images.remove(rowData);
    }

    public String getCurrentResource() {
        return currentResource;
    }

    public List<StringDouble> getDataList() {
        if (dataList == null) {
            loadDataList();
        }

        return dataList;
    }

    public List<String> getHierarchy() {
        if (hierarchy == null) {
            loadHierarchy();
        }

        return hierarchy;
    }

    public DataModel<Location> getLocationModel() {
        if (locationModel == null) {
            if (ontResource == null) {
                return null;
            }

            List<String> coordinates =
onto.listPropertyValues(ontResource,
                    Consts.LOCATION_COORDINATES);
            locations = new ArrayList<Location>();

            for (String coordinate : coordinates) {
                locations.add(new Location(coordinate,
onto.getPairedProperty(ontResource,
                        Consts.LOCATION_STRING, coordinate)));
            }

            locationModel = new
ListDataModel<Location>(locations);
```

```
        }

        return locationModel;
    }

    public Location getLocation() {
        return location;
    }

    public boolean isEditLocation() {
        return isEditLocation;
    }

    public DataModel<StringDouble> getImageModel() {
        if (imageModel == null) {
            if (ontResource == null) {
                return null;
            }

            List<String> image =
onto.listPropertyValues(ontResource, Consts.IMAGE);
            List<String> imageSource = new
ArrayList<String>();

            for (String currentImage : image) {

    imageSource.add(onto.getPairedProperty(ontResource,
Consts.IMAGE_SOURCE,
                        currentImage));
            }

            images =
StringUtil.convertToStringDoubleList(image, imageSource);
            imageModel = new
ListDataModel<StringDouble>(images);
        }

        return imageModel;
    }

    public StringDouble getImage() {
        return image;
    }

    public boolean isEditImage() {
        return isEditImage;
    }

   public void setCurrentResource(String currentResource)
{
        this.currentResource = currentResource;
    }

    private void loadDataList() {
        ontResource =
onto.getOntResource(currentResource);

        if (ontResource == null)
            return;

        String[] fields = OntologyUtil.getFields("plant");
        String[] prop1 = OntologyUtil.getFields("plant.prop1");
        String[] prop2 = OntologyUtil.getFields("plant.prop2");

        dataList = new ArrayList<StringDouble>();

        for (String field : fields) {
            List<String> values =
onto.listPropertyValues(ontResource, field);
            String valueString = "";

            int pos = Arrays.asList(prop1).indexOf(field);

            if (values.size() == 1) {
                valueString = values.get(0);

                if
(field.equals(Consts.PHYTOCHEMICAL_FOUND)) {

                    String phytochemicalSource =
onto.getPairedProperty(ontResource,

    Consts.PHYTOCHEMICAL_FOUND_REFERENCE,
valueString);
                    String plantPart =
onto.getPairedProperty(ontResource,

    Consts.PHYTOCHEMICAL_FOUND_PLANT_PART,
valueString);

                    valueString = plantPart + ": " +
getPhytochemicalLink(valueString,
                            phytochemicalSource);
                } else if (pos >= 0) {
                    String pairedProp =
onto.getPairedProperty(ontResource, prop2[pos],
                            valueString);

                    valueString = valueString + " (" + pairedProp
+ ")";
                }
            } else {
                valueString = "<ul>";

                if
(field.equals(Consts.PHYTOCHEMICAL_FOUND)) {
                    for (String currentPhytochemical : values) {
                        String currentSource =
onto.getPairedProperty(ontResource,

    Consts.PHYTOCHEMICAL_FOUND_REFERENCE,
currentPhytochemical);
                        String currentPart =
onto.getPairedProperty(ontResource,

    Consts.PHYTOCHEMICAL_FOUND_PLANT_PART,
currentPhytochemical);

                        valueString += "<li>"
                                + currentPart + ": "
                                +
getPhytochemicalLink(currentPhytochemical,
currentSource)
                                + "</li>";
                    }
                } else if (pos >= 0) {
                    for (String currentValue : values) {
                        String currentPairedProp =
onto.getPairedProperty(ontResource,
                                prop2[pos], currentValue);

                        valueString += "<li>"
                                + currentValue + " (" +
currentPairedProp + ")"
                                + "</li>";
                    }
                } else {
                    for (String value : values) {
                        valueString += "<li>" + value + "</li>";
                    }
                }

                valueString += "</ul>";
            }

            dataList.add(new StringDouble(field.replace("_", "
"), valueString));
        }
    }
```

```
    private void loadHierarchy() {
        hierarchy =
Util.reverseOrder(onto.getHierarchy(currentResource,
"Plant"));
    }

    private String getPhytochemicalLink(String
phytochemicalName, String phytochemicalSource) {
        OntResource phytochemical =
onto.getOntResource(phytochemicalName);
        String properName =
onto.getPropertyValue(phytochemical,
Consts.PHYTOCHEMICAL_NAME);

        return "<a href=\"../phytochemical/show.jsf?name=" +
phytochemicalName + "\">" + properName
                + "</a> (" + phytochemicalSource + ")";
    }

}
```

PlantComparator.java

```
package model.plant;

import java.util.Comparator;

public class PlantComparator implements
Comparator<Plant> {

    @Override
    public int compare(Plant plant1, Plant plant2) {
        return
plant1.getScientificName().compareToIgnoreCase(plant2.
getScientificName());
    }

}
```

PlantForm.java

```
package model.plant;

import java.util.ArrayList;
import java.util.List;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.event.ActionEvent;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;

import model.AbstractForm;
import model.StringDouble;
import model.StringSingle;
import model.StringTriple;
import model.StringUtil;
import ontology.Consts;
import ontology.Ontology;
import ontology.OntologyInstance;

import com.hp.hpl.jena.ontology.Individual;
import com.hp.hpl.jena.ontology.OntResource;
import com.hp.hpl.jena.rdf.model.StmtIterator;

@ManagedBean
@ViewScoped
public class PlantForm extends AbstractForm {
    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private String currentResource;
    private String parentClass;
    private Plant plant = new Plant();
    private transient DataModel<StringSingle>
speciesSynonymModel;
    private transient DataModel<StringSingle>
localNameModel;
    private transient DataModel<StringTriple>
phytochemicalModel;
    private transient DataModel<StringDouble>
usablePartModel;

    public PlantForm() {
        loadPlant();
    }

    public String actionAdd() {
        Individual individual =
onto.createIndividual(plant.getScientificName(),
parentClass);
    individual.addLiteral(onto.getProperty(Consts.SCIENTI
FIC_NAME),
                plant.getScientificName().trim());

        onto.commit();

        return "show?faces-redirect=true&name=" +
individual.getLocalName();
    }

    public String actionEdit() {
        OntResource individual =
onto.getOntResource(plant.getOntologyLocalName());

    individual.removeAll(onto.getProperty(Consts.SPECIES
_SYNONYM));
        for (StringSingle ss : plant.getSpeciesSynonym()) {
    individual.addLiteral(onto.getProperty(Consts.SPECIES
_SYNONYM), ss.getString().trim());
        }

    individual.removeAll(onto.getProperty(Consts.LOCAL_
NAME));
        for (StringSingle ss : plant.getLocalName()) {
    individual.addLiteral(onto.getProperty(Consts.LOCAL_
NAME), ss.getString().trim());
        }

    individual.removeAll(onto.getProperty(Consts.PHYTOC
HEMICAL_FOUND));
    individual.removeAll(onto.getProperty(Consts.PHYTOC
HEMICAL_FOUND_REFERENCE));
    individual.removeAll(onto.getProperty(Consts.PHYTOC
HEMICAL_FOUND_PLANT_PART));
        for (StringTriple sd : plant.getPhytochemical()) {
            String string1 = sd.getString1().trim().replace(" ",
"_");
    individual.addProperty(onto.getProperty(Consts.PHYTO
CHEMICAL_FOUND),
                    onto.getOntResource(string1));
    individual.addLiteral(onto.getProperty(Consts.PHYTOC
HEMICAL_FOUND_REFERENCE),
                    string1 + sd.getString2().trim());
    individual.addLiteral(onto.getProperty(Consts.PHYTOC
HEMICAL_FOUND_PLANT_PART),
                    string1 + sd.getString3().trim());
        }

    individual.removeAll(onto.getProperty(Consts.TRADITI
ONAL_KNOWLEDGE));
    individual.removeAll(onto.getProperty(Consts.TRADITI
ONAL_KNOWLEDGE_USE));
        for (StringDouble sd : plant.getUsablePart()) {
            String string1 = sd.getString1().trim().replace(" ",
"_");
    individual.addProperty(onto.getProperty(Consts.TRADI
TIONAL_KNOWLEDGE),
                    onto.getOntResource(string1));
```

```
    individual.addLiteral(onto.getProperty(Consts.TRADITI
ONAL_KNOWLEDGE_USE),
                string1 + sd.getString2().trim());
        }

        onto.commit();

        return "show?faces-redirect=true&name=" +
plant.getOntologyLocalName();
    }

    public String actionDelete() {
        onto.removeIndividual(currentResource);
        onto.commit();

        return "list?faces-redirect=true";
    }

    public void actionGetParent(ActionEvent event) {
        parentClass = (String)
event.getComponent().getAttributes().get("parentClass");
    }

    public void addNewSpeciesSynonym() {
        plant.getSpeciesSynonym().add(new StringSingle());
    }

    public void addNewLocalName() {
        plant.getLocalName().add(new StringSingle());
    }

    public void addNewPhytochemical() {
        plant.getPhytochemical().add(new StringTriple());
    }

    public void addNewUsablePart() {
        plant.getUsablePart().add(new StringDouble());
    }

    public void deleteSpeciesSynonym() {

    plant.getSpeciesSynonym().remove(speciesSynonymM
odel.getRowData());
    }

    public void deleteLocalName() {

    plant.getLocalName().remove(localNameModel.getRow
Data());
    }

    public void deletePhytochemical() {

    plant.getPhytochemical().remove(phytochemicalModel.
getRowData());
    }

    public void deleteUsablePart() {

    plant.getUsablePart().remove(usablePartModel.getRow
Data());
    }

    public String getCurrentResource() {
        return currentResource;
    }

    public Plant getPlant() {
        if (plant == null || plant.getOntologyLocalName() ==
null

    || !plant.getOntologyLocalName().equals(currentResour
ce)) {
            loadPlant();
        }
        return plant;
    }

    public DataModel<StringSingle>
getSpeciesSynonymModel() {
        if (speciesSynonymModel == null) {
            speciesSynonymModel = new
ListDataModel<StringSingle>(plant.getSpeciesSynonym());
        }

        return speciesSynonymModel;
    }

    public DataModel<StringSingle> getLocalNameModel()
{
        if (localNameModel == null) {
            localNameModel = new
ListDataModel<StringSingle>(plant.getLocalName());
        }

        return localNameModel;
    }

    public DataModel<StringTriple>
getPhytochemicalModel() {
        if (phytochemicalModel == null) {
            phytochemicalModel = new
ListDataModel<StringTriple>(plant.getPhytochemical());
        }

        return phytochemicalModel;
    }

    public DataModel<StringDouble> getUsablePartModel()
{
        if (usablePartModel == null) {
            usablePartModel = new
ListDataModel<StringDouble>(plant.getUsablePart());
        }

        return usablePartModel;
    }

   public void setCurrentResource(String currentResource)
{
        this.currentResource = currentResource;
    }

    private void loadPlant() {
        if (currentResource != null
&& !currentResource.equals("")) {
            OntResource ontResource =
onto.getOntResource(currentResource);
            plant = new Plant(currentResource,
                    onto.getPropertyValue(ontResource,
Consts.SCIENTIFIC_NAME));

            List<String> speciesSynonym =
onto.listPropertyValues(ontResource,
                    Consts.SPECIES_SYNONYM);

    plant.setSpeciesSynonym(StringUtil.convertToStringSin
gleList(speciesSynonym));

            List<String> localName =
onto.listPropertyValues(ontResource,
Consts.LOCAL_NAME);

    plant.setLocalName(StringUtil.convertToStringSingleLis
t(localName));

            List<String> phytochemicalFound =
onto.listPropertyValues(ontResource,
                    Consts.PHYTOCHEMICAL_FOUND);
```

```
        List<String> phytochemicalSource = new
ArrayList<String>();
        List<String> phytochemicalPart = new
ArrayList<String>();
        for (int i = 0; i < phytochemicalFound.size(); i++) {
            String currentPhytochemicalFound =
phytochemicalFound.get(i);
            phytochemicalFound.set(i,
currentPhytochemicalFound.replace("_", " "));

    phytochemicalSource.add(onto.getPairedProperty(ontR
esource,

    Consts.PHYTOCHEMICAL_FOUND_REFERENCE,
currentPhytochemicalFound));

    phytochemicalPart.add(onto.getPairedProperty(ontRes
ource,

    Consts.PHYTOCHEMICAL_FOUND_PLANT_PART,
currentPhytochemicalFound));
        }

    plant.setPhytochemical(StringUtil.convertToStringTriple
List(phytochemicalFound,
                phytochemicalSource, phytochemicalPart));

        List<String> usablePart =
onto.listPropertyValues(ontResource,
Consts.TRADITIONAL_KNOWLEDGE);
        List<String> usablePartUse = new
ArrayList<String>();
        for (int i = 0; i < usablePart.size(); i++) {
            String currentUsablePart = usablePart.get(i);
            usablePart.set(i, currentUsablePart.replace("_",
" "));

    usablePartUse.add(onto.getPairedProperty(ontResourc
e,

    Consts.TRADITIONAL_KNOWLEDGE_USE,
currentUsablePart));
        }

    plant.setUsablePart(StringUtil.convertToStringDoubleLi
st(usablePart, usablePartUse));
      }
    }

    public List<String> specieComplete(String query) {
      List<String> results = new ArrayList<String>();

      StmtIterator si =
onto.search(Consts.SCIENTIFIC_NAME, query);
      while (si.hasNext()) {

    results.add(si.next().getSubject().getLocalName().repla
ce("_", " "));
      }

      return results;
    }

    public List<String> phytochemicalComplete(String
query) {
      List<String> results = new ArrayList<String>();

      StmtIterator si =
onto.search(Consts.PHYTOCHEMICAL_NAME, query);
      while (si.hasNext()) {

    results.add(si.next().getSubject().getLocalName().repla
ce("_", " "));
      }

      return results;
    }

    public List<String> partComplete(String query) {
      List<String> results = new ArrayList<String>();

      StmtIterator si = onto.search(Consts.PART_NAME,
query);
      while (si.hasNext()) {

    results.add(si.next().getSubject().getLocalName().repla
ce("_", " "));
      }

      return results;
    }

}
```

PlantRenameForm.java

```
package model.plant;

import javax.faces.bean.ManagedBean;

import model.TaxonomyRenameForm;
import ontology.Consts;
import ontology.Ontology;

import com.hp.hpl.jena.ontology.OntResource;

@ManagedBean
public class PlantRenameForm extends
TaxonomyRenameForm {

    @Override
    public String actionEdit() {
      onto.renameResource(oldName, newName);

      OntResource individual =
onto.getOntResource(Ontology.getValidLocalName(newN
ame));

    individual.removeAll(onto.getProperty(Consts.SCIENTI
FIC_NAME));

    individual.addLiteral(onto.getProperty(Consts.SCIENTI
FIC_NAME), newName);

      onto.commit();

      return "show?faces-redirect=true&name=" +
individual.getLocalName();
    }

    @Override
    public void setOldName(String oldName) {
      OntResource individual =
onto.getOntResource(oldName);
      this.oldName = oldName;
      this.newName = onto.getPropertyValue(individual,
Consts.SCIENTIFIC_NAME);
    }

}
```

PlantSearchForm.java

```
package model.plant;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import javax.faces.bean.ManagedBean;
```

```
import model.StringUtil;
import ontology.Consts;
import ontology.Ontology;
import ontology.OntologyInstance;

import com.hp.hpl.jena.ontology.OntResource;
import com.hp.hpl.jena.rdf.model.StmtIterator;

@ManagedBean
public class PlantSearchForm {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private String query;
    private List<Plant> results;

    public void search() {
        results = new ArrayList<Plant>();

        StmtIterator si =
onto.search(Consts.SCIENTIFIC_NAME, query);
        while (si.hasNext()) {
            Plant plant =
convertToPlant(si.next().getSubject().getLocalName());
            if (plant != null)
                if (!results.contains(plant))
                    results.add(plant);
        }

        si = onto.search(Consts.SPECIES_SYNONYM,
query);
        while (si.hasNext()) {
            Plant plant =
convertToPlant(si.next().getSubject().getLocalName());
            if (plant != null)
                if (!results.contains(plant))
                    results.add(plant);
        }

        si = onto.search(Consts.LOCAL_NAME, query);
        while (si.hasNext()) {
            Plant plant =
convertToPlant(si.next().getSubject().getLocalName());
            if (plant != null)
                if (!results.contains(plant))
                    results.add(plant);
        }

        si =
onto.searchResource(Consts.SCIENTIFIC_NAME, query);
        while (si.hasNext()) {
            Plant plant =
convertToPlant(si.next().getSubject().getLocalName());
            if (plant != null)
                if (!results.contains(plant))
                    results.add(plant);
        }

        Collections.sort(results, new PlantComparator());
    }

    public String getQuery() {
        return query;
    }

    public List<Plant> getResults() {
        search();
        return results;
    }

    public void setQuery(String query) {
        this.query = query.trim().toLowerCase();
    }

    private Plant convertToPlant(String localName) {
        OntResource ontResource =
onto.getOntResource(localName);
        if (ontResource == null)
            return null;

       Plant plant = new Plant(ontResource.getLocalName(),
                onto.getPropertyValue(ontResource,
Consts.SCIENTIFIC_NAME));

    plant.setLocalName(StringUtil.convertToStringSingleLis
t(
                onto.listPropertyValues(ontResource,
Consts.LOCAL_NAME)));

        return plant;
    }

}
```

PlantTaxonomyForm.java

```
package model.plant;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.event.ActionEvent;

import ontology.Ontology;
import ontology.OntologyInstance;

import model.AbstractForm;

@ManagedBean
@ViewScoped
public class PlantTaxonomyForm extends AbstractForm {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private String currentType = "group";
    private String name;
    private String parentClass = "Plant";

    public String actionAdd() {
        onto.createClass(name, parentClass);

        return "add-success";
    }

    public void actionGetParent(ActionEvent event) {
        if (currentType.equals("family")) {
            parentClass = (String)
event.getComponent().getAttributes().get("groupClass");
        } else if (currentType.equals("genus")) {
            parentClass = (String)
event.getComponent().getAttributes().get("familyClass");
        }
    }

    public String getCurrentType() {
        return currentType;
    }

    public String getName() {
        return name;
    }

    public void setCurrentType(String currentType) {
        this.currentType = currentType;
    }

    public void setName(String name) {
        this.name = name;
```

```
    }

}
```

PlantTaxonomySelectBean.java

```
package model.plant;

import java.util.ArrayList;
import java.util.List;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.model.SelectItem;

import ontology.Ontology;
import ontology.OntologyInstance;

@ManagedBean
@ViewScoped
public class PlantTaxonomySelectBean {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private String group;
    private String family;
    private String genus;

    public List<SelectItem> getGroups() {
        List<SelectItem> groupList = new
ArrayList<SelectItem>();

        List<String> groups = onto.listSubClasses("Plant");
        for (String group : groups) {
            groupList.add(new SelectItem(group));
        }

        genus = "";

        return groupList;
    }

    public List<SelectItem> getFamilies() {
        List<SelectItem> familyList = new
ArrayList<SelectItem>();

        if (group != null && !group.isEmpty()) {
            List<String> families = onto.listSubClasses(group);
            for (String family : families) {
                familyList.add(new SelectItem(family));
            }

        }

        return familyList;
    }

    public List<SelectItem> getGenera() {
        List<SelectItem> genusList = new
ArrayList<SelectItem>();

        if (family != null && !family.isEmpty()) {
            List<String> genera = onto.listSubClasses(family);
            for (String genus : genera) {
                genusList.add(new SelectItem(genus));
            }

        }

        return genusList;
    }

    public String getGroup() {
        return group;
    }

    public String getFamily() {
        return family;
    }

    public String getGenus() {
        return genus;
    }

    public void setGroup(String group) {
        this.group = group;
    }

    public void setFamily(String family) {
        this.family = family;
    }

    public void setGenus(String genus) {
        this.genus = genus;
    }

}
```

PlantTaxonomyTreeBean.java

```
package model.plant;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;

import model.StringUtil;
import model.TaxonomyUtil;
import ontology.Consts;
import ontology.Ontology;
import ontology.OntologyInstance;

import org.primefaces.model.DefaultTreeNode;
import org.primefaces.model.TreeNode;

import com.hp.hpl.jena.ontology.Individual;

@ManagedBean
@ViewScoped
public class PlantTaxonomyTreeBean {

    private Ontology onto =
OntologyInstance.getInstance().getOntology();

    private TreeNode root;
    private TreeNode selectedNode;
    private String selectedNodeString;
    private List<Plant> plants;

    public PlantTaxonomyTreeBean() {
        selectedNodeString = "Plant";

        root = new DefaultTreeNode(selectedNodeString,
null);
        TaxonomyUtil.getSubClassesOf(onto,
selectedNodeString, root);

        loadPlants();
    }

    public TreeNode getRoot() {
        return root;
    }

    public TreeNode getSelectedNode() {
        return selectedNode;
```

```
    }

    public String getSelectedNodeString() {
        return selectedNodeString;
    }

    public List<Plant> getPlants() {
        return plants;
    }

    public void setSelectedNode(TreeNode selectedNode) {
        if (selectedNode != null) {
            this.selectedNode = selectedNode;
            selectedNodeString =
selectedNode.getData().toString().replace(" ", "_");
            loadPlants();
        }
    }

    private void loadPlants() {
        plants = new ArrayList<Plant>();
        List<Individual> individuals =
onto.listInstances(selectedNodeString, false);

        for (Individual individual : individuals) {
            Plant plant = new Plant(individual.getLocalName(),
                    onto.getPropertyValue(individual,
Consts.SCIENTIFIC_NAME));

            List<String> localName =
onto.listPropertyValues(individual,  Consts.LOCAL_NAME);

    plant.setLocalName(StringUtil.convertToStringSingleLis
t(localName));

            List<String> image =
onto.listPropertyValues(individual, Consts.IMAGE);
            List<String> imageSource =
onto.listPropertyValues(individual,
Consts.IMAGE_SOURCE);

    plant.setImage(StringUtil.convertToStringDoubleList(im
age, imageSource));

            plants.add(plant);
        }

        Collections.sort(plants, new PlantComparator());
    }

}
```

AbstractUserForm.java

```
package model.user;

import javax.faces.application.FacesMessage;
import javax.faces.component.UIComponent;
import javax.faces.component.UIInput;
import javax.faces.context.FacesContext;
import javax.faces.validator.ValidatorException;

import model.AbstractForm;
import webapp.Config;
import dao.DAOException;
import dao.UserDAO;

public abstract class AbstractUserForm extends
AbstractForm {

    public static final String REGISTERED_MESSAGE_ID
= "model.AbstractUserForm.REGISTERED";

    protected final UserDAO userDAO =
Config.getInstance().getDAOFactory().getUserDAO();

    public void validateUsername(FacesContext
facesContext, UIComponent component, Object value)
        throws ValidatorException
    {
        String username = (String) value;
        try {
            if (userDAO.existUsername(username)) {
                throw new ValidatorException(
                    new
FacesMessage(getMessage(EXIST_MESSAGE_ID,
getLabel(component))));
            }
        } catch (DAOException e) {
            setErrorMessage(e);
        }
    }

    public void validatePassword(FacesContext
facesContext, UIComponent component, Object value)
        throws ValidatorException
    {
        String passwordId = (String)
component.getAttributes().get("passwordId");
        UIInput passwordInput = (UIInput)
facesContext.getViewRoot().findComponent(passwordId);
        UIInput confirmInput = (UIInput) component;
        String password = (String) passwordInput.getValue();
        String confirm = (String) value;

        if (!confirm.equals(password)) {

            passwordInput.resetValue();
            confirmInput.resetValue();

            setErrorMessage(passwordInput,
                getMessage(INEQUAL_MESSAGE_ID,
getLabel(passwordInput), getLabel(confirmInput)));

            throw new ValidatorException(new
FacesMessage(""));
        }
    }

    public void validateEmail(FacesContext facesContext,
UIComponent component, Object value)
        throws ValidatorException
    {
        String email = (String) value;
        if
(!email.matches("([^.@]+)(\\.[^.@]+)*@([^.@]+\\.)+([^.@]+)
")) {
            throw new ValidatorException(
                new
FacesMessage(getMessage(INVALID_MESSAGE_ID,
getLabel(component))));
        } else {
            try {
                if (userDAO.existEmail(email)) {
                    throw new ValidatorException(
                        new
FacesMessage(getMessage(EXIST_MESSAGE_ID,
getLabel(component))));
                }
            } catch (DAOException e) {
                setErrorMessage(e);
            }
        }
    }

}
```

RegisterUserForm.java

```
package model.user;
```

```
import java.util.List;

import javax.faces.bean.ManagedBean;
import javax.faces.model.SelectItem;

import dao.DAOException;

@ManagedBean
public class RegisterUserForm extends AbstractUserForm
{

    private User user = new User();
    private List<SelectItem> userRoles;

    public String registerUser() {
        try {
            userDAO.create(user);
        } catch (DAOException e) {
            setErrorMessage(e);
        }

        return "show-users";
    }

    public User getUser() {
        return user;
    }

    public List<SelectItem> getUserRoles() {
    if (userRoles == null) {
        userRoles = UserUtil.getUserRoles();
    }

    return userRoles;
    }

}
```

User.java

```
package model.user;

public class User {

    private Long id;
    private String username;
    private String password;
    private UserRole userRole = new UserRole();
    private String lastName;
    private String firstName;

    public User() {

    }

    public User(Long id, String username, String password,
UserRole userRole,
            String lastName, String firstName) {
        this.id = id;
        this.username = username;
        this.password = password;
        this.userRole = userRole;
        this.lastName = lastName;
        this.firstName = firstName;
    }

    public Long getId() {
        return id;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }

    public UserRole getUserRole() {
        return userRole;
    }

    public String getLastName() {
        return lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public void setUserRole(UserRole userRole) {
        this.userRole = userRole;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public boolean equals(Object other) {
        return (other instanceof User) && (id != null)
            ? id.equals(((User) other).id) : (other == this);
    }

    public int hashCode() {
        return (id != null) ? (this.getClass().hashCode() +
id.hashCode()) : super.hashCode();
    }

}
```

UserForm.java

```
package model.user;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ManagedProperty;

import dao.DAOException;

@ManagedBean
public class UserForm extends AbstractUserForm {

    @ManagedProperty(value="#{userSession}")
    private UserSession userSession;
    private String username;
    private String password;

    public String login() {
        try {
            User user = userDAO.find(username, password);

            if (user != null) {
                userSession.setUser(user);
```

```
            } else {
                return "login-failed";
            }
        } catch (DAOException e) {

        }

        return "logged-in";
    }

    public String logout() {
        userSession.setUser(null);

        return "logged-out";
    }

    public String saveUser() {
    try {
            if (password != null && !password.equals(""))
                userSession.getUser().setPassword(password);
            userDAO.save(userSession.getUser());
        } catch (DAOException e) {
            e.printStackTrace();
        } finally {
            password = "";
        }

    return "show";
    }

    public UserSession getUserSession() {
        return userSession;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }

    public void setUserSession(UserSession userSession) {
        this.userSession = userSession;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public void setPassword(String password) {
        this.password = password;
    }

}
```

UserListBean.java

```
package model.user;

import java.util.List;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.ViewScoped;
import javax.faces.model.DataModel;
import javax.faces.model.ListDataModel;
import javax.faces.model.SelectItem;

import webapp.Config;
import dao.DAOException;
import dao.UserDAO;

@ManagedBean
@ViewScoped
public class UserListBean {

    private List<User> users;
    private transient DataModel<User> userModel;
    private User user = new User();
    private boolean isEditUser;
    private String password;

    private List<SelectItem> userRoles;

    protected final UserDAO userDAO =
Config.getInstance().getDAOFactory().getUserDAO();

    public void actionEditUser() {
        user = userModel.getRowData();
        isEditUser = true;
    }

    public void actionSaveUser() {
        try {
            if (password != null && !password.equals(""))
                user.setPassword(password);
            userDAO.save(user);
        } catch (DAOException e) {
            e.printStackTrace();
        } finally {
            user = new User();
            isEditUser = false;
            password = "";
        }
    }

    public void actionDeleteUser() {
        User rowData = userModel.getRowData();

        try {
            userDAO.delete(rowData);
        } catch (DAOException e) {
            e.printStackTrace();
        } finally {
            users.remove(rowData);
        }
    }

    public DataModel<User> getUserModel() {
        if (userModel == null) {
            try {
                users = userDAO.list();
                userModel = new ListDataModel<User>(users);
            } catch (DAOException e) {
                e.printStackTrace();
            }
        }

        return userModel;
    }

    public User getUser() {
        return user;
    }

    public boolean isEditUser() {
        return isEditUser;
    }

    public String getPassword() {
        return password;
    }

    public List<SelectItem> getUserRoles() {
    if (userRoles == null) {
        userRoles = UserUtil.getUserRoles();
    }

    return userRoles;
    }
```

```
    public void setPassword(String password) {
        this.password = password;
    }

}
```

UserRole.java

```
package model.user;

public class UserRole {

    private Long id;
    private String name;

    public UserRole() {

    }

    public UserRole(Long id, String name) {
        this.id = id;
        this.name = name;
    }

    public Long getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public void setName(String name) {
        this.name = name;
    }

    public boolean equals(Object other) {
        return other instanceof UserRole && (id != null) ?
id.equals(((UserRole) other).id) : (other == this);
    }

    public int hashCode() {
        return id != null ? this.getClass().hashCode() +
id.hashCode() : super.hashCode();
    }

}
```

UserRoleConverter.java

```
package model.user;

import javax.faces.component.UIComponent;
import javax.faces.context.FacesContext;
import javax.faces.convert.Converter;

import dao.DAOException;
import dao.UserRoleDAO;

import webapp.Config;

public class UserRoleConverter implements Converter {

    private static UserRoleDAO userRoleDAO =

    Config.getInstance().getDAOFactory().getUserRoleDA
O();

    public Object getAsObject(FacesContext context,
UIComponent component, String value) {
        UserRole userRole = null;

        try {
            userRole =
userRoleDAO.find(Long.parseLong(value));
        } catch (NumberFormatException e) {
            e.printStackTrace();
        } catch (DAOException e) {
            e.printStackTrace();
        }

        return userRole;
    }

    public String getAsString(FacesContext context,
UIComponent component, Object value) {
        return ((UserRole) value).getId().toString();
    }

}
```

UserSession.java

```
package model.user;

import java.io.Serializable;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean
@SessionScoped
public class UserSession implements Serializable {

    private static final long serialVersionUID = 1L;

    private User user;

    public UserSession() {
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public boolean isLoggedIn() {
        return user != null;
    }

}
```

UserUtil.java

```
package model.user;

import java.util.ArrayList;
import java.util.List;

import javax.faces.model.SelectItem;

import webapp.Config;

import dao.DAOException;
import dao.UserRoleDAO;

public class UserUtil {

    static List<SelectItem> getUserRoles() {
        UserRoleDAO userRoleDAO =
Config.getInstance().getDAOFactory().getUserRoleDAO();
        List<SelectItem> userRoles = null;
```

```
        try {
        userRoles = new ArrayList<SelectItem>();
        for (UserRole userRole : userRoleDAO.list()) {
            userRoles.add(new SelectItem(userRole,
userRole.getName()));
        }
        } catch (DAOException e) {
            e.printStackTrace();
        }

        return userRoles;
    }

}
```

Config.java

```
package ontology;

import java.util.ResourceBundle;

public class Config {
    private static Config config;
    private ResourceBundle rb;
    private static final String SOURCE = "ontology/config";

    private Config() {
    }

    public static Config getInstance() {
        if (config == null) {
            config = new Config();
            config.rb = ResourceBundle.getBundle(SOURCE);
        }
        return config;
    }

    public String getValue(String key) {
        return rb.getString(key);
    }

    public void setTemplate(String template) {

    }
}
```

Consts.java

```
package ontology;

public final class Consts {

    public static final String SCIENTIFIC_NAME =
"Scientific_Name";
    public static final String SPECIES_SYNONYM =
"Species_Synonym";
   public static final String LOCAL_NAME = "Local_Name";
    public static final String PHYTOCHEMICAL_FOUND =
"Phytochemical_Found";
    public static final String
PHYTOCHEMICAL_FOUND_PLANT_PART =
"Phytochemical_Found_Plant_Part";
    public static final String
PHYTOCHEMICAL_FOUND_REFERENCE =
"Phytochemical_Found_Reference";
    public static final String TRADITIONAL_KNOWLEDGE
= "Traditional_Knowledge";
    public static final String
TRADITIONAL_KNOWLEDGE_USE =
"Traditional_Knowledge_Use";
    public static final String LOCATION_COORDINATES =
"Location_Coordinates";
    public static final String LOCATION_STRING =
"Location_String";
    public static final String IMAGE = "Image";
    public static final String IMAGE_SOURCE =
"Image_Source";

    public static final String PHYTOCHEMICAL_NAME =
"Name";
    public static final String KEGG = "KEGG";
    public static final String CHEBI = "ChEBI";
    public static final String FORMULA = "Formula";
    public static final String MOLECULAR_WEIGHT =
"Molecular_Weight";
    public static final String IUPAC_NAME =
"IUPAC_Name";
    public static final String SYNONYM = "Synonym";

    public static final String PART_NAME = "Part_Name";

}
```

Ontology.java

```
package ontology;

import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;

import com.hp.hpl.jena.ontology.Individual;
import com.hp.hpl.jena.ontology.OntClass;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntResource;
import com.hp.hpl.jena.rdf.model.NodeIterator;
import com.hp.hpl.jena.rdf.model.Property;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.SimpleSelector;
import com.hp.hpl.jena.rdf.model.Statement;
import com.hp.hpl.jena.rdf.model.StmtIterator;
import com.hp.hpl.jena.util.ResourceUtils;

public class Ontology {

    private String ontology;
    private OntModel model = null;
    private String prefix;

    public Ontology(String ontology) {
        this.ontology = ontology;
        model =
OntologyConnect.loadOntology(this.ontology, false, null);
        prefix = Config.getInstance().getValue(ontology +
".prefix");
    }

    public OntResource getOntResource(String localName)
{
        return getOntResource(localName, false);
    }

    public OntResource getOntResource(String name,
boolean isProperName) {
        if (isProperName) {
            name = getValidLocalName(name);
        }
        return model.getOntResource(prefix + name);
    }

    public Property getProperty(String propertyString) {
        return model.getProperty(prefix + propertyString);
    }

    public String getPropertyValue(OntResource
ontResource, String propertyString) {
```

```
        Property property = getProperty(propertyString);
        RDFNode value =
ontResource.getPropertyValue(property);

        String valueString = getPlainValueString(value);

        return valueString;
    }

    public List<String> listPropertyValues(OntResource
ontResource, String propertyString) {
        Property property = getProperty(propertyString);
        NodeIterator ni =
ontResource.listPropertyValues(property);

        List<String> values = new ArrayList<String>();

        while (ni.hasNext()) {
            RDFNode value = ni.next();
            String valueString = getPlainValueString(value);

            values.add(valueString);
        }

        return values;
    }

    public String getPairedProperty(OntResource
ontResource, String propertyString2,
            String valueString1) {

        Property property = getProperty(propertyString2);
        NodeIterator ni =
ontResource.listPropertyValues(property);

        while (ni.hasNext()) {
            RDFNode value = ni.next();
            String valueString = getPlainValueString(value);

            if (valueString.startsWith(valueString1)) {
                return valueString.replaceFirst(valueString1, "");
            }
        }

        return null;
    }

    public List<String> listSubClasses(String className) {
        OntClass mainClass;
        Iterator<OntClass> subClasses;

        try {
            mainClass = model.getOntClass(prefix +
className);

            subClasses = mainClass.listSubClasses(true);
        } catch (java.lang.NullPointerException e) {
            return null;
        }

        ArrayList<String> subClassStrings = new
ArrayList<String>();

        while (subClasses.hasNext()) {
            OntClass subClass = subClasses.next();
            subClassStrings.add(subClass.getLocalName());
        }

        Collections.sort(subClassStrings,
String.CASE_INSENSITIVE_ORDER);

        return subClassStrings;
    }

    public List<Individual> listInstances(String className,
boolean directOnly) {
        OntClass mainClass = model.getOntClass(prefix +
className);
        @SuppressWarnings("rawtypes")
        Iterator instances;

        try {
            instances = mainClass.listInstances(true);
        } catch (NullPointerException e) {
            return null;
        }

        List<Individual> individualList = new
ArrayList<Individual>();

        while (instances.hasNext()) {
            individualList.add((Individual) instances.next());
        }

        if (!directOnly) {
            Iterator<OntClass> subClasses =
mainClass.listSubClasses(true);
            while (subClasses.hasNext()) {

    individualList.addAll(listInstances(subClasses.next().get
LocalName(), false));
            }
        }

        return individualList;
    }

    public List<String> getHierarchy(String localName,
String root) {
        List<String> hierarchy = new ArrayList<String>();
        return getHierarchy(prefix + localName, prefix + root,
hierarchy);
    }

    private List<String> getHierarchy(String localName,
String root, List<String> hierarchy) {
       Resource resource = model.getResource(localName);

        StmtIterator si =
resource.listProperties(model.getProperty(
                "http://www.w3.org/1999/02/22-rdf-syntax-
ns#type"));

        while (si.hasNext()) {
            Statement parent = si.next();

            if
(parent.getObject().toString().equals("http://www.w3.org/2
002/07/owl#Thing"))
                continue;

            if
(parent.getObject().asResource().getLocalName().equals(
"Class")) {
                parent =
resource.getProperty(model.getProperty(
                        "http://www.w3.org/2000/01/rdf-
schema#subClassOf"));
                if (parent == null) {
                    continue;
                }
            }

            if (root.equals(parent.getObject().toString())) {
                return hierarchy;
            }
```

```
    hierarchy.add(getPlainValueString(parent.getObject()).r
eplace("_", " "));

          return getHierarchy(parent.getObject().toString(),
root, hierarchy);
       }

       return hierarchy;
    }

    public StmtIterator search(String propertyString, final
String searchTerm) {
       Property property = getProperty(propertyString);

       StmtIterator iter = model.listStatements(
             new SimpleSelector(null, property, (RDFNode)
null) {
                @Override
                public boolean selects(Statement s) {
                   return
s.getString().toLowerCase().contains(searchTerm);
                }
             }
       );

       return iter;
    }

    public StmtIterator searchNotLiteral(String
propertyString, final String searchTerm) {
       Property property = getProperty(propertyString);

       StmtIterator iter = model.listStatements(
             new SimpleSelector(null, property, (RDFNode)
null) {
                @Override
                public boolean selects(Statement s) {
                   return s.toString().contains(searchTerm);
                }
             }
       );

       return iter;
    }

    public StmtIterator searchResource(String
propertyString, final String objectString) {
       Property property = getProperty(propertyString);

       StmtIterator iter = model.listStatements(
             new SimpleSelector(null, property, (RDFNode)
null) {
                @Override
                public boolean selects(Statement s) {
                   Resource resource = s.getSubject();
                   return hasParent(resource, objectString);
                }

                private boolean hasParent(Resource
resource, String queryString) {
                   StmtIterator si =
resource.listProperties(model.getProperty(
                         "http://www.w3.org/1999/02/22-rdf-
syntax-ns#type"));

                   while (si.hasNext()) {
                      Statement parent = si.next();

                      if (parent.getObject().toString().equals(

    "http://www.w3.org/2002/07/owl#Thing"))
                         continue;

                      if
(parent.getObject().asResource().getLocalName().equals(
"Class")) {
                         parent =
resource.getProperty(model.getProperty(
                               "http://www.w3.org/2000/01/rdf-
schema#subClassOf"));
                         if (parent == null) {
                            continue;
                         }
                      }

                      return
parent.getObject().toString().toLowerCase().contains(quer
yString)
                            ||
hasParent(parent.getObject().asResource(), queryString);
                   }

                   return false;
                }
             }
       );

       return iter;
    }

    public void createClass(String properName, String
parentClassName) {
       OntClass parentClass = model.getOntClass(prefix +
parentClassName);
       OntClass newClass = model.createClass(prefix +
getValidLocalName(properName));

       newClass.setSuperClass(parentClass);
    }

    public Individual createIndividual(String properName,
String parentClassName) {
       OntClass parentClass = model.getOntClass(prefix +
parentClassName);
       Individual newIndividual =
model.createIndividual(prefix +
getValidLocalName(properName),
             parentClass);

       return newIndividual;
    }

    public void renameResource(String oldLocalName,
String newProperName) {
       String newLocalName =
getValidLocalName(newProperName);

       if (!oldLocalName.equals(newLocalName)) {
          OntResource origResource =
getOntResource(oldLocalName);
          ResourceUtils.renameResource(origResource,
prefix + newLocalName);
       }
    }

    public void removeIndividual(String localName) {
       Individual individual = model.getIndividual(prefix +
localName);
       individual.remove();
    }

    public void removePairedProperty(String localName,
String propertyString1,
          String propertyString2, String valueString1) {

       OntResource ontResource =
getOntResource(localName);
```

```
        Property property = getProperty(propertyString1);
        NodeIterator ni =
ontResource.listPropertyValues(property);

        while (ni.hasNext()) {
            RDFNode value = ni.next();
            String valueString = getPlainValueString(value);

            if (valueString.equals(valueString1)) {
                ontResource.removeProperty(property, value);
                break;
            }
        }

        property = getProperty(propertyString2);
        ni = ontResource.listPropertyValues(property);

        while (ni.hasNext()) {
            RDFNode value = ni.next();
            String valueString = getPlainValueString(value);

            if (valueString.startsWith(valueString1)) {
                ontResource.removeProperty(property, value);
                break;
            }
        }
    }

    public void commit() {
        model.commit();
    }

    public void importOntology(String filename) {
        model =
OntologyConnect.loadOntology(this.ontology, true,
filename);
    }

    public void exportOntology(OutputStream out) {
        model.write(out, "RDF/XML-ABBREV");
    }

    public static String getValidLocalName(String
properName) {
        String localName = properName.trim();
        localName = localName.replaceAll(" ", "_");
        localName = localName.replaceAll("\\W", "");
        localName = localName.replaceAll("_+", "_");

        return localName;
    }

    public String getPlainValueString(RDFNode value) {
        if (value == null)
            return "";

        String valueString = value.toString();

        if (value instanceof Resource) {
            valueString =
valueString.substring(valueString.indexOf("#") + 1,
                    valueString.length());
        } else {
            valueString = valueString.substring(0,
valueString.indexOf("^^"));
        }

        return valueString;
    }

}
```

OntologyConnect.java

```
package ontology;

import com.hp.hpl.jena.ontology.OntDocumentManager;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.ModelMaker;

import dao.DAOProperties;

public class OntologyConnect {

    public static OntModel loadOntology(String ontology,
boolean cleanDB, String source) {
        String sourceURI =
Config.getInstance().getValue(ontology + ".url");

        String filename;
        if (cleanDB) {
            filename = "file:" + source;
        } else {
            filename = "file:" +
Config.getInstance().getValue(ontology + ".file");
        }


    OntDocumentManager.getInstance().addAltEntry(sourc
eURI, filename);

        PersistentOntology po = new PersistentOntology();

        DAOProperties properties = new
DAOProperties("hanapinsp2.jdbc");

        try {
            Class.forName(properties.getProperty("driver",
true));
        } catch (Exception e) {
            e.printStackTrace();
        }

        if (cleanDB) {
        ModelMaker maker = po.getRDBMaker("MySQL",
true);

        po.loadDB(maker, sourceURI);
        }

        ModelMaker maker = po.getRDBMaker("MySQL",
false);

        Model base =
maker.createModel(Config.getInstance().getValue(ontolog
y + ".url"), false);

        OntModel m =
ModelFactory.createOntologyModel(po.getModelSpec(ma
ker), base);

        return m;
    }

    public static void clearDB() {

    }

    public static void removeModel(String model) {

    }

}
```

OntologyInstance.java

```
package ontology;
```

```
public class OntologyInstance {

    private static OntologyInstance ontologyInstance;
    private Ontology ontology;

    private OntologyInstance() {

    }

    public static OntologyInstance getInstance() {
        if (ontologyInstance == null) {
            ontologyInstance = new OntologyInstance();
            ontologyInstance.ontology = new Ontology("npo");
        }
        return ontologyInstance;
    }

    public Ontology getOntology() {
        return ontology;
    }

}
```

OntologyUtil.java

```
package ontology;

public class OntologyUtil {

    public static String[] getFields(String type) {
        String fieldString =
Config.getInstance().getValue("npo." + type + ".fields");
        String[] fields = fieldString.split(",");
        return fields;
    }

}
```

PersistentOntology.java

```
/*************************************************************************
***
 * Source code information
 * -----------------------
 * Original author    Ian Dickinson, HP Labs Bristol
 * Author email       ian_dickinson@users.sourceforge.net
 * Package            Jena 2
 * Web                http://sourceforge.net/projects/jena/
 * Created            25-Jul-2003
 * Filename           $RCSfile: PersistentOntology.java,v $
 * Revision           $Revision: 1.3 $
 * Release status     $State: Exp $
 *
 * Last modified on   $Date: 2009/10/06 13:04:42 $
 *               by   $Author: ian_dickinson $
 *
 * (c) Copyright 2002, 2003, 2004, 2005 Hewlett-Packard
Development Company, LP
 * (see footer for full conditions)

*************************************************************************
**/

package ontology;

import com.hp.hpl.jena.db.DBConnection;
import com.hp.hpl.jena.db.IDBConnection;
import com.hp.hpl.jena.ontology.OntModel;
import com.hp.hpl.jena.ontology.OntModelSpec;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.ModelMaker;

import dao.ConnectionPool;

public class PersistentOntology {

    public void loadDB(ModelMaker maker, String source) {
        // use the model maker to get the base model as a
persistent model
        // strict=false, so we get an existing model by that
name if it exists
        // or create a new one
        Model base = maker.createModel(source, false);

        // now we plug that base model into an ontology
model that also uses
        // the given model maker to create storage for
imported models
        OntModel m =
ModelFactory.createOntologyModel(getModelSpec(maker
), base);

        // now load the source document, which will also load
any imports
        m.read(source);
    }

    public ModelMaker getRDBMaker(String dbType,
boolean cleanDB) {
        try {
            // Create database connection
            IDBConnection conn  = new
DBConnection(ConnectionPool.getInstance().getConnecti
on(),
                dbType);

            // do we need to clean the database?
            if (cleanDB) {
                conn.cleanDB();
            }

            // Create a model maker object
            return  ModelFactory.createModelRDBMaker(conn);
        }
        catch (Exception e) {
            e.printStackTrace();
            System.exit(1);
        }

        return null;
    }

    public OntModelSpec getModelSpec(ModelMaker
maker) {
        // create a spec for the new ont model that will use no
inference over models
        // made by the given maker (which is where we get
the persistent models from)
        OntModelSpec spec = new
OntModelSpec(OntModelSpec.OWL_MEM);
        spec.setImportModelMaker(maker);

        return spec;
    }

}

/*
    (c) Copyright 2002, 2003, 2004, 2005 Hewlett-Packard
Development Company, LP
    All rights reserved.

    Redistribution and use in source and binary forms, with
or without
    modification, are permitted provided that the following
conditions
    are met:
```

```
   1. Redistributions of source code must retain the above
copyright
      notice, this list of conditions and the following
disclaimer.

   2. Redistributions in binary form must reproduce the
above copyright
      notice, this list of conditions and the following
disclaimer in the
      documentation and/or other materials provided with
the distribution.

   3. The name of the author may not be used to endorse
or promote products
      derived from this software without specific prior written
permission.

   THIS SOFTWARE IS PROVIDED BY THE AUTHOR
``AS IS'' AND ANY EXPRESS OR
   IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES
   OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED.
   IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
ANY DIRECT, INDIRECT,
   INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT
   NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE,
   DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY
   THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT LIABILITY, OR TORT
   (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF
   THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
*/
```

config.properties

```
npo.url = http://purl.org/biotop/biotop.owl
npo.file = NPO.owl
npo.prefix = http://purl.org/biotop/biotop.owl#

npo.plant.fields =
Scientific_Name,Species_Synonym,Local_Name,Phytoch
emical_Found,Traditional_Knowledge
npo.plant.prop1.fields = Traditional_Knowledge
npo.plant.prop2.fields = Traditional_Knowledge_Use

npo.phytochemical.fields =
Name,KEGG,ChEBI,Formula,Molecular_Weight,IUPAC_N
ame,Synonym

kegg.url = http://www.kegg.jp/entry/
chebi.url =
https://www.ebi.ac.uk/chebi/searchId.do?chebiId=
```

Config.java

```
package webapp;

import javax.faces.bean.ApplicationScoped;
import javax.faces.bean.ManagedBean;
import javax.faces.context.FacesContext;

import dao.DAOFactory;

@ManagedBean(name="config")
@ApplicationScoped
public class Config {

   private static final String MANAGED_BEAN_NAME =
"config";

   private DAOFactory daoFactory;

   public Config() {
      String databaseName = "hanapinsp2.jdbc";
      this.daoFactory =
DAOFactory.getInstance(databaseName);
   }

   public static Config getInstance() {
      FacesContext facesContext =
FacesContext.getCurrentInstance();
      return (Config)
facesContext.getApplication().evaluateExpressionGet(
         facesContext, "#{" + MANAGED_BEAN_NAME +
"}", Config.class);
   }

   public DAOFactory getDAOFactory() {
      return daoFactory;
   }

}
```

Functions.java

```
package webapp;

public class Functions {

   public static String concat(String string1, String string2)
{
      return string1.concat(string2);
   }

}
```

dao.properties

```
hanapinsp2.jdbc.url = jdbc:mysql://localhost/HANAPIN-
SP-2
hanapinsp2.jdbc.driver = com.mysql.jdbc.Driver
hanapinsp2.jdbc.username = HANAPIN-SP-2
hanapinsp2.jdbc.password = ***************
```

messages.properties

```
javax.faces.component.UIInput.REQUIRED = Please
enter {0}.
javax.faces.converter.IntegerConverter.INTEGER =
Please enter valid {2} (numbers only).
javax.faces.validator.LengthValidator.MINIMUM = {1}
should be at least {0} characters long.

model.AbstractForm.EXIST = {0} already in use, please
choose another.
model.AbstractForm.INVALID = Please enter valid {0}.
model.AbstractForm.INEQUAL = {0} and {1} are not equal,
please retype both.

model.AbstractUserForm.REGISTERED = Registration
succeed! Your new user ID is {0}.

dao.DAOException = Request failed due to database error.
Please try again later. Detail message: {0}
```

admin/add-user.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../WEB-INF/templates/main.xhtml">
```

```
<ui:define name="title">Register New User</ui:define>

<ui:define name="body">

   <comp:auth role="1" />

   <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 1}">
      <h:form>
         <h:panelGrid columns="2">
            <h:outputText value="Username"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputText
value="#{registerUserForm.user.username}"
id="username"
                  label="username" required="true"
size="30">
                  <f:validateLength minimum="3" />
               </h:inputText>
               <br/>
               <h:message for="username"
styleClass="message" />
            </h:panelGroup>

            <h:outputText value="Password"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputText
value="#{registerUserForm.user.password}"
id="password"
                  label="password" required="true"
size="30">
                  <f:validateLength minimum="3" />
               </h:inputText>
               <br/>
               <h:message for="password"
styleClass="message" />
            </h:panelGroup>

            <h:outputText value="User Role"
styleClass="field_name" />
            <h:panelGroup>
               <h:selectOneMenu
value="#{registerUserForm.user.userRole}" id="userRole"
                  label="user role" required="true">
                  <f:selectItems
value="#{registerUserForm.userRoles}" />
               </h:selectOneMenu>
               <br/>
               <h:message for="userRole"
styleClass="message" />
            </h:panelGroup>

            <h:outputText value="Last Name"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputText
value="#{registerUserForm.user.lastName}"
id="lastName"
                  label="last name" required="true"
size="30">
               </h:inputText>
               <br/>
               <h:message for="lastName"
styleClass="message" />
            </h:panelGroup>

            <h:outputText value="First Name"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputText
value="#{registerUserForm.user.firstName}"
id="firstName"
                  label="first name" required="true"
size="30">
               </h:inputText>
               <br/>
               <h:message for="firstName"
styleClass="message" />
            </h:panelGroup>

            <h:outputText />
            <h:commandButton value="Register"
action="#{registerUserForm.registerUser}" />
         </h:panelGrid>
      </h:form>
   </h:panelGroup>

</ui:define>

</ui:composition>
```

admin/index.jsp

```
<jsp:scriptlet> response.sendRedirect("index.jsf");
</jsp:scriptlet>
```

admin/index.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">Administration</ui:define>

<ui:define name="body">

   <comp:auth role="1" />

   <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 1}">
      <ul>
      <li><h:outputLink value="show-users.jsf">Manage
Users</h:outputLink></li>
      <li><h:outputLink value="add-user.jsf">Register New
User</h:outputLink></li>
      </ul>
   </h:panelGroup>

</ui:define>

</ui:composition>
```

admin/show-users.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">Manage Users</ui:define>

<ui:define name="body">

   <comp:auth role="1" />

   <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 1}">
      <h:panelGroup
rendered="#{userListBean.editUser}">
         <h6>Editing User</h6>
         <h:form>
            <h:panelGrid columns="2">
```

```
                <h:outputText value="Username"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{userListBean.user.username}" id="username"
                        label="username" required="true"
size="30">
                        <f:validateLength minimum="3" />
                    </h:inputText>
                    <br/>
                    <h:message for="username"
styleClass="message" />
                </h:panelGroup>

                <h:outputText value="Password"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{userListBean.password}" id="password"
                        label="password" size="30">
                        <f:validateLength minimum="3" />
                    </h:inputText>
                    <br/>
                    <h:outputText value="(Leave blank to
retain old password.)"
                        styleClass="message_info" />
                    <h:message for="password"
styleClass="message" />
                </h:panelGroup>

                <h:outputText value="User Role"
styleClass="field_name" />
                <h:panelGroup>
                    <h:selectOneMenu
value="#{userListBean.user.userRole}" id="userRole"
                        label="user role" required="true">
                        <f:selectItems
value="#{userListBean.userRoles}" />
                    </h:selectOneMenu>
                    <br/>
                    <h:message for="userRole"
styleClass="message" />
                </h:panelGroup>

                <h:outputText value="Last Name"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{userListBean.user.lastName}" id="lastName"
                        label="last name" required="true"
size="30">
                    </h:inputText>
                    <br/>
                    <h:message for="lastName"
styleClass="message" />
                </h:panelGroup>

                <h:outputText value="First Name"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{userListBean.user.firstName}" id="firstName"
                        label="first name" required="true"
size="30">
                    </h:inputText>
                    <br/>
                    <h:message for="firstName"
styleClass="message" />
                </h:panelGroup>

                <h:outputText />
                <h:commandButton value="Save"
action="#{userListBean.actionSaveUser}" />
            </h:panelGrid>
        </h:form>
    </h:panelGroup>

    <h:form>
        <h:dataTable value="#{userListBean.userModel}"
var="dataItem">
            <h:column>
            <f:facet name="header"><h:outputText
value="Username" /></f:facet>
            <h:outputText value="#{dataItem.username}" />
            </h:column>

            <h:column>
            <f:facet name="header"><h:outputText
value="Role" /></f:facet>
            <h:outputText
value="#{dataItem.userRole.name}" />
            </h:column>

            <h:column>
            <f:facet name="header"><h:outputText
value="Last Name" /></f:facet>
            <h:outputText value="#{dataItem.lastName}" />
            </h:column>

            <h:column>
            <f:facet name="header"><h:outputText
value="First Name" /></f:facet>
            <h:outputText value="#{dataItem.firstName}" />
            </h:column>

            <h:column>
            <h:commandButton value="edit"
action="#{userListBean.actionEditUser}"
                rendered="#{userSession.user.id !=
dataItem.id}" />
            <h:commandButton value="delete"
action="#{userListBean.actionDeleteUser}"
                onclick="javascript:return confirm('Are you
sure you want to delete this user?')"
                rendered="#{userSession.user.id !=
dataItem.id}" />
        </h:column>
        </h:dataTable>
    </h:form>
</h:panelGroup>

</ui:define>

</ui:composition>
```

curation/export-ontology.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:comp="http://java.sun.com/jsf/composite/comp"
    template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">Export Ontology</ui:define>

<ui:define name="body">

    <comp:auth role="2" />

    <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
        <h:form>
            <h:commandLink value="Download OWL file."
action="#{ontologyBean.actionExport}" />
        </h:form>
    </h:panelGroup>

</ui:define>

</ui:composition>
```

curation/import-ontology.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:t="http://myfaces.apache.org/tomahawk"
xmlns:comp="http://java.sun.com/jsf/composite/comp"
template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">Import Ontology</ui:define>

<ui:define name="body">

   <comp:auth role="2"/>

   <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
      <h:form enctype="multipart/form-data">
         <h:panelGrid columns="2">
            <h:outputText value="OWL File"
styleClass="field_name"/>
            <h:panelGroup>
               <t:inputFileUpload
value="#{ontologyBean.uploadedFile}" id="filename"
label="OWL file" required="true">
                  <f:validator
validatorId="ontologyUploadValidator"/>
               </t:inputFileUpload>
               <br/>
               <h:message for="filename"
styleClass="message"/>
            </h:panelGroup>

            <h:outputText/>
            <h:commandButton value="Import"
action="#{ontologyBean.actionImport}"
onclick="javascript:return confirm('Are you sure you want
to import a new ontology and overwrite the existing
one?')"/>
         </h:panelGrid>
      </h:form>
   </h:panelGroup>

</ui:define>

</ui:composition>
```

curation/index.jsp

```
<jsp:scriptlet> response.sendRedirect("index.jsf");
</jsp:scriptlet>
```

curation/index.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:comp="http://java.sun.com/jsf/composite/comp"
template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">Curation</ui:define>

<ui:define name="body">

   <comp:auth role="2"/>

   <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
      <ul>
      <li><h:outputLink value="import-ontology.jsf">Import
Ontology</h:outputLink></li>
      <li><h:outputLink value="export-ontology.jsf">Export
Ontology</h:outputLink></li>
      </ul>
   </h:panelGroup>

</ui:define>

</ui:composition>
```

phytochemical/taxonomy/add.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../../WEB-INF/templates/main.xhtml">

<ui:define name="title">New Phytochemical Taxonomy
Category</ui:define>

<ui:define name="body">

   <comp:auth role="2" />

   <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
      <h:form>
         <h:panelGrid columns="2">
            <h:outputText value="Name"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputText
value="#{phytochemicalTaxonomyForm.name}" id="name"
                  label="Name" required="true"
size="50">
                  <f:validator validatorId="existsValidator"
/>
               </h:inputText>
               <br />
               <h:message for="name"
styleClass="message" />
            </h:panelGroup>

            <h:outputText />
            <h:commandButton value="Add"
action="#{phytochemicalTaxonomyForm.actionAdd}" />
         </h:panelGrid>
      </h:form>
   </h:panelGroup>

</ui:define>

</ui:composition>
```

phytochemical/taxonomy/edit.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../../WEB-INF/templates/main.xhtml">

<ui:define
name="title">#{taxonomyRenameForm.oldName}</ui:defi
ne>

<ui:define name="body">

   <f:metadata>
      <f:viewParam name="name"
value="#{taxonomyRenameForm.oldName}" />
   </f:metadata>

   <comp:auth role="2" />
```

```
    <comp:resourceRename />

</ui:define>

</ui:composition>
```

phytochemical/add.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:comp="http://java.sun.com/jsf/composite/comp"
    template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">New Phytochemical</ui:define>

<ui:define name="body">

    <comp:auth role="2" />

    <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
        <h:form>
            <h:panelGrid columns="2">
                <h:outputText value="Category"
styleClass="field_name" />
                <comp:phytochemicalTaxonomySelector />

                <h:outputText value="Name"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{phytochemicalForm.phytochemical.name}"
                        id="name" label="name" required="true"
size="50">
                        <f:validator validatorId="existsValidator" />
                    </h:inputText>
                    <br />
                    <h:message for="name"
styleClass="message" />
                </h:panelGroup>

                <h:outputText value="KEGG"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{phytochemicalForm.phytochemical.kegg}"
                        id="kegg" label="KEGG" size="50">
                    </h:inputText>
                    <br />
                    <h:message for="KEGG"
styleClass="message" />
                </h:panelGroup>

                <h:outputText value="ChEBI"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{phytochemicalForm.phytochemical.chebi}"
                        id="chebi" label="ChEBI" size="50">
                        <f:validateLongRange />
                    </h:inputText>
                    <br />
                    <h:message for="chebi"
styleClass="message" />
                </h:panelGroup>

                <h:outputText value="Formula"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{phytochemicalForm.phytochemical.formula}"
                        id="formula" label="formula"
required="true" size="50">
                    </h:inputText>
                    <br />
                    <h:message for="formula"
styleClass="message" />
                </h:panelGroup>

                <h:outputText value="Molecular Weight"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{phytochemicalForm.phytochemical.molecularWei
ght}"
                        id="molecularWeight" label="molecular
weight" required="true" size="50">
                        <f:validateDoubleRange />
                    </h:inputText>
                    <br />
                    <h:message for="molecularWeight"
styleClass="message" />
                </h:panelGroup>

                <h:outputText />
                <h:commandButton value="Add"
action="#{phytochemicalForm.actionAdd}"

    actionListener="#{phytochemicalForm.actionGetParent}
">
                    <f:attribute name="parentClass"

    value="#{phytochemicalTaxonomySelectBean.category
}" />
                </h:commandButton>
            </h:panelGrid>
        </h:form>
    </h:panelGroup>

</ui:define>

</ui:composition>
```

phytochemical/edit.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:f="http://java.sun.com/jsf/core"
xmlns:ui="http://java.sun.com/jsf/facelets"
xmlns:comp="http://java.sun.com/jsf/composite/comp"
template="../WEB-INF/templates/main.xhtml">

<ui:define
name="title">#{phytochemicalForm.phytochemical.name}<
/ui:define>

<ui:define name="body">

    <f:metadata>
        <f:viewParam name="name"
value="#{phytochemicalForm.currentResource}"/>
    </f:metadata>

    <comp:auth role="2"/>

    <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
        <h:form>
            <h:panelGrid columns="2">
                <h:outputText value="KEGG"
styleClass="field_name"/>
                <h:panelGroup>
                    <h:inputText
value="#{phytochemicalForm.phytochemical.kegg}"
id="kegg" label="KEGG" size="50">
                    </h:inputText>
                    <br/>
```

```
            <h:message for="kegg"
styleClass="message"/>
         </h:panelGroup>

         <h:outputText value="ChEBI"
styleClass="field_name"/>
         <h:panelGroup>
            <h:inputText
value="#{phytochemicalForm.phytochemical.chebi}"
id="chebi" label="ChEBI" size="50">
               <f:validateLongRange/>
            </h:inputText>
            <br/>
            <h:message for="chebi"
styleClass="message"/>
         </h:panelGroup>

         <h:outputText value="Formula"
styleClass="field_name"/>
         <h:panelGroup>
            <h:inputText
value="#{phytochemicalForm.phytochemical.formula}"
id="formula" label="formula" required="true" size="50">
            </h:inputText>
            <br/>
            <h:message for="formula"
styleClass="message"/>
         </h:panelGroup>

         <h:outputText value="Molecular Weight"
styleClass="field_name"/>
         <h:panelGroup>
            <h:inputText
value="#{phytochemicalForm.phytochemical.molecularWei
ght}" id="molecularWeight" label="molecular weight"
required="true" size="50">
               <f:validateDoubleRange/>
            </h:inputText>
            <br/>
            <h:message for="molecularWeight"
styleClass="message"/>
         </h:panelGroup>

         <h:outputText value="IUPAC Name"
styleClass="field_name"/>
         <h:panelGroup>
            <h:inputText
value="#{phytochemicalForm.phytochemical.iupacName}"
id="iupacName" label="IUPAC name" size="50">
            </h:inputText>
            <br/>
            <h:message for="iupacName"
styleClass="message"/>
         </h:panelGroup>

         <h:outputText value="Synonym"
styleClass="field_name"/>
         <h:panelGroup>
            <h:dataTable
value="#{phytochemicalForm.synonymModel}"
var="dataItem">
               <h:column>
                  <h:inputText value="#{dataItem.string}"
id="synonym#{dataItem.string}" label="synonym"
required="true" size="30"/>
                  <br/>
                  <h:message
for="synonym#{dataItem.string}" styleClass="message"/>
               </h:column>
               <h:column>
                  <h:commandButton value="remove"
immediate="true"
action="#{phytochemicalForm.deleteSynonym}"/>
               </h:column>
            </h:dataTable>
            <h:commandButton value="Add"
action="#{phytochemicalForm.addNewSynonym}"/>
         </h:panelGroup>

         <h:outputText/>
         <h:panelGroup>
            <h:commandButton value="Save"
action="#{phytochemicalForm.actionEdit}"/>
            <h:commandButton value="Delete"
action="#{phytochemicalForm.actionDelete}"
onclick="javascript:return confirm('Are you sure you want
to delete this phytochemical?')"/>
            <br/>
            <h:link
outcome="show?name=#{phytochemicalForm.currentRes
ource}" value="Cancel"/>
         </h:panelGroup>
      </h:panelGrid>
   </h:form>
</h:panelGroup>

</ui:define>

</ui:composition>
```

phytochemical/edit-name.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../WEB-INF/templates/main.xhtml">

<ui:define
name="title">#{phytochemicalRenameForm.newName}</u
i:define>

<ui:define name="body">

   <f:metadata>
      <f:viewParam name="name"
value="#{phytochemicalRenameForm.oldName}" />
   </f:metadata>

   <comp:auth role="2" />

   <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
      <h:form>
         <h:panelGrid columns="2">
            <h:outputText value="Name"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputText
value="#{phytochemicalRenameForm.newName}"
id="name" label="name"
                  required="true" size="50">
                  <f:validator validatorId="existsValidator" />
               </h:inputText>
               <br />
               <h:message for="name"
styleClass="message" />
            </h:panelGroup>

            <h:outputText />
            <h:panelGroup>
               <h:commandButton value="Edit"
action="#{phytochemicalRenameForm.actionEdit}" />
               <br />
               <h:link
outcome="show?name=#{phytochemicalRenameForm.old
Name}" value="Cancel" />
            </h:panelGroup>
         </h:panelGrid>
```

```
    </h:form>
  </h:panelGroup>

</ui:define>

</ui:composition>
```

phytochemical/index.jsp

```
<% response.sendRedirect("list.jsf"); %>
```

phytochemical/list.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:p="http://primefaces.prime.com.tr/ui"
  xmlns:comp="http://java.sun.com/jsf/composite/comp"
  template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">List of Phytochemicals</ui:define>

<ui:define name="body">

  <h:outputStylesheet name="treestyle.css" library="css"
/>

  <comp:phytochemicalSearch /><br />

  <h:form>

    <h:outputText value="Use the taxonomic tree to
narrow down the list of phytochemicals." />

    <h:panelGrid columns="2">

      <h:panelGroup>
        <h:panelGroup id="admin_options"
styleClass="admin_options"
          rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
          <h:outputText value="Options: " />
          <h:link value="Edit"

  outcome="taxonomy/edit?name=#{phytochemicalTaxon
omyTreeBean.selectedNodeString}"
            rendered="#{!empty
phytochemicalTaxonomyTreeBean.selectedNodeString
and
phytochemicalTaxonomyTreeBean.selectedNodeString !=
'Phytochemical'}" />
          <br />
          <br />
        </h:panelGroup>

        <p:tree
value="#{phytochemicalTaxonomyTreeBean.root}"
var="node" dynamic="true"
          selectionMode="single"
selection="#{phytochemicalTaxonomyTreeBean.selectedN
ode}"
          update="phytochemical_list admin_options">
          <p:treeNode>
            <h:outputText value="#{node}" />
          </p:treeNode>
        </p:tree>
      </h:panelGroup>

      <h:panelGroup>
        <h:panelGroup id="phytochemical_list">
          <h2>Phytochemicals</h2>

          <comp:phytochemicalList
  phytochemicals="#{phytochemicalTaxonomyTreeBean.
phytochemicals}" />
        </h:panelGroup>
      </h:panelGroup>

    </h:panelGrid>

  </h:form>

</ui:define>

</ui:composition>
```

phytochemical/search.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:comp="http://java.sun.com/jsf/composite/comp"
  template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">Search
Phytochemicals</ui:define>

<ui:define name="body">

  <h:outputStylesheet name="treestyle.css" library="css"
/>

  <comp:phytochemicalSearch /><br />

  <h:form>
    <ui:fragment rendered="#{!empty
phytochemicalSearchForm.query}">
      <h2>Results</h2>

      <comp:phytochemicalList
phytochemicals="#{phytochemicalSearchForm.results}" />
    </ui:fragment>
  </h:form>

</ui:define>

</ui:composition>
```

phytochemical/show.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:comp="http://java.sun.com/jsf/composite/comp"
  template="../WEB-INF/templates/main.xhtml">

<ui:define
name="title">#{phytochemicalBean.dataList[0].string2}</ui
:define>

<ui:define name="body">

  <f:metadata>
    <f:viewParam name="name"
value="#{phytochemicalBean.currentResource}" />
  </f:metadata>

  <h:outputStylesheet name="h1marginless.css"
library="css" />

  <h:panelGroup rendered="#{empty
phytochemicalBean.dataList}">
    <h:outputText value="Phytochemical does not exist."
/>
  </h:panelGroup>
```

```
   <h:panelGroup rendered="#{!empty
phytochemicalBean.dataList}">

      <h:panelGroup>
         <h:outputText value="&#8212; " />
         <ui:repeat var="class"
value="#{phytochemicalBean.hierarchy}"
varStatus="status">
            <h:outputText value="#{class} " />
            <h:outputText rendered="#{!status.last}"
value="&#8594; "/>
      </ui:repeat>
      </h:panelGroup>

      <br /><br />

      <h:panelGroup rendered="#{userSession.loggedIn
and userSession.user.userRole.id == 2}"
         styleClass="admin_options">
         <h:link outcome="edit-
name?name=#{phytochemicalBean.currentResource}"
value="Edit Name" />
         <h:outputText value=" | "/>
         <h:link
outcome="edit?name=#{phytochemicalBean.currentResou
rce}" value="Edit Details" />
      </h:panelGroup>

      <h:dataTable value="#{phytochemicalBean.dataList}"
var="dataItem" styleClass="details">
         <h:column>
            <h:outputText value="#{dataItem.string1}"
styleClass="field_name" />
         </h:column>

         <h:column>
            <h:outputText value="#{dataItem.string2}"
escape="false" />
         </h:column>
      </h:dataTable>

      <h6>Species Containing
#{phytochemicalBean.dataList[0].string2}</h6>

      <comp:plantList
plants="#{phytochemicalBean.plants}" />

   </h:panelGroup>

</ui:define>

</ui:composition>
```

plant/taxonomy/add.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../../WEB-INF/templates/main.xhtml">

<ui:define name="title">New Plant Taxonomy
Category</ui:define>

<ui:define name="body">

   <comp:auth role="2" />

   <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
      <h:form>
         <h:panelGrid columns="2">
            <h:outputText value="Type"
styleClass="field_name" />
            <h:panelGroup>
               <h:selectOneRadio
value="#{plantTaxonomyForm.currentType}" id="type"
                  label="type" required="true">
                  <f:selectItem itemLabel="Group"
itemValue="group" />
                  <f:selectItem itemLabel="Family"
itemValue="family" />
                  <f:selectItem itemLabel="Genus"
itemValue="genus" />
                  <f:ajax render="taxonomy_selector" />
               </h:selectOneRadio>
               <br />
               <h:message for="type" styleClass="message"
/>
               <h:panelGroup id="taxonomy_selector">
                  <comp:plantTaxonomySelector
type="#{plantTaxonomyForm.currentType}" />
               </h:panelGroup>
            </h:panelGroup>

            <h:outputText value="Name"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputText
value="#{plantTaxonomyForm.name}" id="name"
label="name"
                  required="true" size="50">
                  <f:validator validatorId="existsValidator" />
               </h:inputText>
               <br />
               <h:message for="name"
styleClass="message" />
            </h:panelGroup>

            <h:outputText />
            <h:commandButton value="Add"
action="#{plantTaxonomyForm.actionAdd}"

   actionListener="#{plantTaxonomyForm.actionGetParent
}">
               <f:attribute name="groupClass"
value="#{plantTaxonomySelectBean.group}" />
               <f:attribute name="familyClass"
value="#{plantTaxonomySelectBean.family}" />
            </h:commandButton>
         </h:panelGrid>
      </h:form>
   </h:panelGroup>

</ui:define>

</ui:composition>
```

plant/taxonomy/edit.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../../WEB-INF/templates/main.xhtml">

<ui:define
name="title">#{taxonomyRenameForm.oldName}</ui:defi
ne>

<ui:define name="body">

   <f:metadata>
      <f:viewParam name="name"
value="#{taxonomyRenameForm.oldName}" />
   </f:metadata>

   <comp:auth role="2" />
```

```
    <comp:resourceRename />

</ui:define>

</ui:composition>
```

plant/add.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:comp="http://java.sun.com/jsf/composite/comp"
    template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">New Species</ui:define>

<ui:define name="body">

    <comp:auth role="2" />

    <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
        <h:form>
            <h:panelGrid columns="2">
                <h:outputText value="Genus"
styleClass="field_name" />
                <h:panelGroup>
                    <comp:plantTaxonomySelector
type="species" />
                </h:panelGroup>

                <h:outputText value="Scientific Name"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{plantForm.plant.scientificName}" id="name"
                        label="scientific name" required="true"
size="50">
                        <f:validator validatorId="existsValidator" />
                    </h:inputText>
                    <br />
                    <h:message for="name"
styleClass="message" />
                </h:panelGroup>

                <h:outputText />
                <h:commandButton value="Add"
action="#{plantForm.actionAdd}"

    actionListener="#{plantForm.actionGetParent}">
                    <f:attribute name="parentClass"
value="#{plantTaxonomySelectBean.genus}" />
                </h:commandButton>
            </h:panelGrid>
        </h:form>
    </h:panelGroup>

</ui:define>

</ui:composition>
```

plant/add-location.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:comp="http://java.sun.com/jsf/composite/comp"
    template="../WEB-INF/templates/main.xhtml">

<ui:define
name="title">#{locationBean.plant.scientificName}</ui:defi
ne>

<ui:define name="body">

    <f:metadata>
        <f:viewParam name="name"
value="#{locationBean.currentResource}" />
    </f:metadata>

    <comp:auth role="2" />

    <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
        <h:form>
            <h:panelGrid columns="2">
                <h:outputText value="Latitude"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText value="#{locationBean.latitude}"
id="latitude"
                        label="latitude" required="true" size="30">
                        <f:validateDoubleRange />
                    </h:inputText>
                    <br />
                    <h:message for="latitude"
styleClass="message" />
                </h:panelGroup>

                <h:outputText value="Longitude"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{locationBean.longitude}" id="longitude"
                        label="longitude" required="true"
size="30">
                        <f:validateDoubleRange />
                    </h:inputText>
                    <br />
                    <h:message for="longitude"
styleClass="message" />
                </h:panelGroup>

                <h:outputText value="Description"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{locationBean.description}" id="description"
                        label="description" required="true"
size="30">
                    </h:inputText>
                    <br />
                    <h:message for="description"
styleClass="message" />
                </h:panelGroup>

                <h:outputText />
                <h:panelGroup>
                    <h:commandButton value="Add"
action="#{locationBean.actionAdd}" />
                    <br />
                    <h:link
outcome="show?name=#{locationBean.currentResource}"
value="Cancel" />
                </h:panelGroup>
            </h:panelGrid>
        </h:form>
    </h:panelGroup>

</ui:define>

</ui:composition>
```

plant/edit.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
```

```
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:p="http://primefaces.prime.com.tr/ui"
    xmlns:comp="http://java.sun.com/jsf/composite/comp"
    template="../WEB-INF/templates/main.xhtml">

<ui:define
name="title">#{plantForm.plant.scientificName}</ui:define
>

<ui:define name="body">

    <f:metadata>
        <f:viewParam name="name"
value="#{plantForm.currentResource}" />
    </f:metadata>

    <comp:auth role="2" />

    <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
        <h:form>
            <h:panelGrid columns="2">
                <h:outputText value="Species Synonym"
styleClass="field_name" />
                <h:panelGroup>
                    <h:dataTable
value="#{plantForm.speciesSynonymModel}"
var="dataItem">
                        <h:column>
                            <h:inputText value="#{dataItem.string}"
id="speciesSynonym#{dataItem.string}"
                                label="species synonym"
required="true" size="30" />
                            <br />
                            <h:message
for="speciesSynonym#{dataItem.string}"
styleClass="message" />
                        </h:column>
                        <h:column>
                            <h:commandButton value="remove"
immediate="true"

    action="#{plantForm.deleteSpeciesSynonym}" />
                        </h:column>
                    </h:dataTable>
                    <h:commandButton value="Add"
action="#{plantForm.addNewSpeciesSynonym}" />
                </h:panelGroup>

                <h:outputText value="Local Name"
styleClass="field_name" />
                <h:panelGroup>
                    <h:dataTable
value="#{plantForm.localNameModel}" var="dataItem">
                        <h:column>
                            <h:inputText value="#{dataItem.string}"
id="localName#{dataItem.string}"
                                label="local name" required="true"
size="30" />
                            <br />
                            <h:message
for="localName#{dataItem.string}" styleClass="message"
/>
                        </h:column>
                        <h:column>
                            <h:commandButton value="remove"
immediate="true"

    action="#{plantForm.deleteLocalName}" />
                        </h:column>
                    </h:dataTable>
                    <h:commandButton value="Add"
action="#{plantForm.addNewLocalName}" />
                </h:panelGroup>

                <h:outputText value="Phytochemical Found"
styleClass="field_name" />
                <h:panelGroup>
                    <h:dataTable
value="#{plantForm.phytochemicalModel}"
var="dataItem">
                        <h:column>
                            <f:facet name="header">
                                <h:outputText value="Plant Part" />
                            </f:facet>
                            <p:autoComplete
value="#{dataItem.string3}" forceSelection="true"

    completeMethod="#{plantForm.partComplete}"
size="10" />
                        </h:column>
                        <h:column>
                            <f:facet name="header">
                                <h:outputText value="Phytochemical
Name" />
                            </f:facet>
                            <p:autoComplete
value="#{dataItem.string1}" forceSelection="true"

    completeMethod="#{plantForm.phytochemicalComplete
}" size="15" />
                        </h:column>
                        <h:column>
                            <f:facet name="header">
                                <h:outputText value="Reference" />
                            </f:facet>
                            <h:inputText value="#{dataItem.string2}"
                                id="psource#{dataItem.string2}"
label="phytochemical source"
                                required="true" size="30" />
                            <br/>
                            <h:message
for="psource#{dataItem.string2}" styleClass="message" />
                        </h:column>
                        <h:column>
                            <h:commandButton value="remove"
immediate="true"

    action="#{plantForm.deletePhytochemical}" />
                        </h:column>
                    </h:dataTable>
                    <h:commandButton value="Add"
action="#{plantForm.addNewPhytochemical}" />
                </h:panelGroup>

                <h:outputText value="Traditional Knowledge"
styleClass="field_name" />
                <h:panelGroup>
                    <h:dataTable
value="#{plantForm.usablePartModel}" var="dataItem">
                        <h:column>
                            <f:facet name="header">
                                <h:outputText value="Part" />
                            </f:facet>
                            <p:autoComplete
value="#{dataItem.string1}" forceSelection="true"

    completeMethod="#{plantForm.partComplete}" />
                        </h:column>
                        <h:column>
                            <f:facet name="header">
                                <h:outputText value="Traditional Use
(separate different uses with comma)" />
                            </f:facet>
                            <h:inputTextarea
value="#{dataItem.string2}"
                                id="traduse#{dataItem.string2}"
label="traditional use"
                                required="true" rows="3" cols="30" />
```

```
                        <br/>
                        <h:message
for="traduse#{dataItem.string2}" styleClass="message" />
                    </h:column>
                    <h:column>
                        <h:commandButton value="remove"
immediate="true"

   action="#{plantForm.deleteUsablePart}" />
                    </h:column>
                </h:dataTable>
                <h:commandButton value="Add"
action="#{plantForm.addNewUsablePart}" />
            </h:panelGroup>

            <h:outputText />
            <h:panelGroup>
                <h:commandButton value="Save"
action="#{plantForm.actionEdit}" />
                <h:commandButton value="Delete"
action="#{plantForm.actionDelete}"
                    onclick="javascript:return confirm('Are you
sure you want to delete this species?')" />
                <br />
                <h:link
outcome="show?name=#{plantForm.currentResource}"
value="Cancel" />
            </h:panelGroup>
        </h:panelGrid>
    </h:form>
  </h:panelGroup>

</ui:define>

</ui:composition>
```

plant/edit-name.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../WEB-INF/templates/main.xhtml">

<ui:define
name="title">#{plantRenameForm.newName}</ui:define>

<ui:define name="body">

   <f:metadata>
      <f:viewParam name="name"
value="#{plantRenameForm.oldName}" />
   </f:metadata>

   <comp:auth role="2" />

   <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
      <h:form>
         <h:panelGrid columns="2">
            <h:outputText value="Scientific Name"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputText
value="#{plantRenameForm.newName}" id="name"
                  label="scientific name" required="true"
size="50">
                  <f:validator validatorId="existsValidator" />
               </h:inputText>
               <br />
               <h:message for="name"
styleClass="message" />
            </h:panelGroup>
            <h:outputText />
            <h:panelGroup>
               <h:commandButton value="Edit"
action="#{plantRenameForm.actionEdit}" />
               <br />
               <h:link
outcome="show?name=#{plantRenameForm.oldName}"
value="Cancel" />
            </h:panelGroup>
         </h:panelGrid>
      </h:form>
   </h:panelGroup>

</ui:define>

</ui:composition>
```

plant/index.jsp

```
<% response.sendRedirect("list.jsf"); %>
```

plant/list.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:p="http://primefaces.prime.com.tr/ui"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">List of Species</ui:define>

<ui:define name="body">

   <h:outputStylesheet name="treestyle.css" library="css"
/>

   <comp:plantSearch /><br />

   <h:form>

      <h:outputText value="Use the taxonomic tree to
narrow down the list of species." />

      <h:panelGrid columns="2">

         <h:panelGroup>
            <h:panelGroup id="admin_options"
styleClass="admin_options"
               rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
               <h:outputText value="Options: " />
               <h:link value="Edit"

   outcome="taxonomy/edit?name=#{plantTaxonomyTree
Bean.selectedNodeString}"
                  rendered="#{!empty
plantTaxonomyTreeBean.selectedNodeString and
plantTaxonomyTreeBean.selectedNodeString != 'Plant'}"
/>
               <br />
               <br />
            </h:panelGroup>

            <p:tree
value="#{plantTaxonomyTreeBean.root}" var="node"
dynamic="true"
               selectionMode="single"
selection="#{plantTaxonomyTreeBean.selectedNode}"
               update="plant_list admin_options">
               <p:treeNode>
                  <h:outputText value="#{node}" />
               </p:treeNode>
            </p:tree>
         </h:panelGroup>
```

```
        <h:panelGroup>
            <h:panelGroup id="plant_list">
                <h2>Plants</h2>

                <comp:plantList
plants="#{plantTaxonomyTreeBean.plants}" />
            </h:panelGroup>
        </h:panelGroup>

    </h:panelGrid>

</h:form>

</ui:define>

</ui:composition>
```

plant/map.html

```
<!DOCTYPE html>
<html>

<head>
    <meta name="viewport" content="initial-scale=1.0,
user-scalable=no" />
    <style type="text/css">
        html { height: 100% }
        body { height: 100%; margin: 0; padding: 0 }
        #map_canvas { height: 100% }
    </style>
    <script type="text/javascript"

    src="http://maps.googleapis.com/maps/api/js?sensor=fa
lse">
    </script>
    <script type="text/javascript">
    function initialize() {
        var coordinates =
window.location.href.split("=")[1].split("%2C");
        var latlng = new google.maps.LatLng(coordinates[0],
coordinates[1]);
        var options = {
                zoom: 10,
                center: latlng,
                mapTypeId:
google.maps.MapTypeId.ROADMAP
    };
        var map = new
google.maps.Map(document.getElementById("map_canva
s"), options);

        var marker = new google.maps.Marker({
            position: latlng,
            map: map,
            title: coordinates[0] + "," + coordinates[1]
        });
    }
    </script>
</head>

<body onload="initialize()">
    <div id="map_canvas" style="width:100%;
height:100%"></div>
</body>

</html>
```

plant/search.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:comp="http://java.sun.com/jsf/composite/comp"
    template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">Search Species</ui:define>

<ui:define name="body">

    <h:outputStylesheet name="treestyle.css" library="css"
/>

    <comp:plantSearch /><br />

    <h:form>
        <ui:fragment rendered="#{!empty
plantSearchForm.query}">
            <h2>Results</h2>

            <comp:plantList
plants="#{plantSearchForm.results}" />
        </ui:fragment>
    </h:form>

</ui:define>

</ui:composition>
```

plant/show.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:p="http://primefaces.prime.com.tr/ui"
    template="../WEB-INF/templates/main.xhtml">

<ui:define
name="title">#{plantBean.dataList[0].string2}</ui:define>

<ui:define name="body">

    <f:metadata>
        <f:viewParam name="name"
value="#{plantBean.currentResource}" />
    </f:metadata>

    <h:outputStylesheet name="h1marginless.css"
library="css" />

    <h:panelGroup rendered="#{empty
plantBean.dataList}">
        <h:outputText value="Plant does not exist." />
    </h:panelGroup>

    <h:panelGroup rendered="#{!empty
plantBean.dataList}">

        <h:panelGroup rendered="#{userSession.loggedIn
and userSession.user.userRole.id == 2 and
plantBean.editLocation}">
            <h6>Editing Location</h6>
            <h:form>
                <h:panelGrid columns="2">
                    <h:outputText value="Latitude"
styleClass="field_name" />
                    <h:panelGroup>
                        <h:inputText
value="#{plantBean.location.latitude}" id="latitude"
                            label="latitude" required="true"
size="30">
                            <f:validateDoubleRange />
                        </h:inputText>
                        <br/>
                        <h:message for="latitude"
styleClass="message" />
                    </h:panelGroup>
```

```
                <h:outputText value="Longitude"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{plantBean.location.longitude}" id="longitude"
                        label="longitude" required="true"
size="30">
                        <f:validateDoubleRange />
                    </h:inputText>
                    <br/>
                    <h:message for="longitude"
styleClass="message" />
                </h:panelGroup>

                <h:outputText value="Description"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{plantBean.location.description}" id="description"
                        label="description" required="true"
size="30">
                    </h:inputText>
                    <br/>
                    <h:message for="description"
styleClass="message" />
                </h:panelGroup>

                <h:outputText />
                <h:commandButton value="Save"
action="#{plantBean.actionSaveLocation}" />
            </h:panelGrid>
        </h:form>
    </h:panelGroup>

    <h:panelGroup rendered="#{userSession.loggedIn
and userSession.user.userRole.id == 2 and
plantBean.editImage}">
        <h6>Editing Image</h6>
        <h:form>
            <h:panelGrid columns="2">
                <h:outputText value="Image"
styleClass="field_name" />
                <h:graphicImage
value="/uploads/plants/thumbs/#{plantBean.image.string1}
" />
                <h:outputText value="Source"
styleClass="field_name" />
                <h:panelGroup>
                    <h:inputText
value="#{plantBean.image.string2}" id="imageSource"
                        label="image source" required="true"
size="30" />
                    <br />
                    <h:message for="imageSource"
styleClass="message" />
                </h:panelGroup>
                <h:outputText />
                <h:commandButton value="Save"
action="#{plantBean.actionSaveImage}" />
            </h:panelGrid>
        </h:form>
    </h:panelGroup>

    <h:panelGroup>
        <h:outputText value="&#8212; " />
        <ui:repeat var="class"
value="#{plantBean.hierarchy}" varStatus="status">
            <h:outputText value="#{class} " />
            <h:outputText rendered="#{!status.last}"
value="&#8594; "/>
    </ui:repeat>
    </h:panelGroup>

    <br /><br />

    <h:panelGroup rendered="#{userSession.loggedIn
and userSession.user.userRole.id == 2}"
        styleClass="admin_options">
        <h:link outcome="edit-
name?name=#{plantBean.currentResource}" value="Edit
Name" />
        <h:outputText value=" | "/>
        <h:link
outcome="edit?name=#{plantBean.currentResource}"
value="Edit Details" />
    </h:panelGroup>

    <h:dataTable value="#{plantBean.dataList}"
var="dataItem" styleClass="details">
        <h:column>
            <h:outputText value="#{dataItem.string1}"
styleClass="field_name" />
        </h:column>

        <h:column>
            <h:outputText value="#{dataItem.string2}"
escape="false" />
        </h:column>
    </h:dataTable>

    <h6>Locations</h6>

    <h:panelGroup rendered="#{userSession.loggedIn
and userSession.user.userRole.id == 2}">
        <span class="admin_options">
            <h:link outcome="add-
location?name=#{plantBean.currentResource}"
value="Add Location" />
        </span>
    </h:panelGroup>

    <h:form>
        <h:dataTable value="#{plantBean.locationModel}"
var="dataItem">
            <h:column>
            <p:lightBox iframe="true" width="80%"
height="80%">
                <h:outputLink title="#{dataItem.description}
(#{dataItem.coordinates})"

   value="map.html?coordinates=#{dataItem.coordinates}"
>
                    <h:outputText
value="#{dataItem.description}" />
                </h:outputLink>
            </p:lightBox>
            </h:column>

            <h:column rendered="#{userSession.loggedIn
and userSession.user.userRole.id == 2}">
            <h:commandButton value="edit"
action="#{plantBean.actionEditLocation}" />
            <h:commandButton value="remove"
action="#{plantBean.actionDeleteLocation}"
                onclick="javascript:return confirm('Are you
sure you want to delete this location?')" />
        </h:column>
        </h:dataTable>
    </h:form>

    <h6>Images</h6>

    <h:panelGroup rendered="#{userSession.loggedIn
and userSession.user.userRole.id == 2}">
        <span class="admin_options">
            <h:link outcome="upload-
image?name=#{plantBean.currentResource}" value="Add
Image" />
        </span>
    </h:panelGroup>
```

```
      <h:form>
         <p:lightBox id="images">
            <h:dataTable value="#{plantBean.imageModel}"
var="dataItem">
               <h:column>
               <h:outputLink
value="/uploads/plants/#{dataItem.string1}"
                  title="Source: #{dataItem.string2}">
                  <h:graphicImage
value="/uploads/plants/thumbs/#{dataItem.string1}" />
               </h:outputLink>
               </h:column>

               <h:column>
                  <h:outputText value="#{dataItem.string2}"
/>
               </h:column>

               <h:column
rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
               <h:commandButton value="edit source"
action="#{plantBean.actionEditImage}" />
               <h:commandButton value="remove"
action="#{plantBean.actionDeleteImage}"
                  onclick="javascript:return confirm('Are you
sure you want to delete this image?')" />
            </h:column>
            </h:dataTable>
         </p:lightBox>
      </h:form>

   </h:panelGroup>

</ui:define>

</ui:composition>
```

plant/upload-image.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:t="http://myfaces.apache.org/tomahawk"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../WEB-INF/templates/main.xhtml">

<ui:define
name="title">#{imageBean.plant.scientificName}</ui:defin
e>

<ui:define name="body">

   <f:metadata>
      <f:viewParam name="name"
value="#{imageBean.currentResource}" />
   </f:metadata>

   <comp:auth role="2" />

   <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
      <h:form enctype="multipart/form-data">
         <h:panelGrid columns="2">
            <h:outputText value="Image File"
styleClass="field_name" />
            <h:panelGroup>
               <t:inputFileUpload
value="#{imageBean.uploadedFile}" storage="file"
                  accept="image/*" id="filename"
label="image file" required="true">
                  <f:validator
validatorId="imageUploadValidator" />
               </t:inputFileUpload>
               <br />
               <h:message for="filename"
styleClass="message" />
            </h:panelGroup>

            <h:outputText value="Source"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputText value="#{imageBean.source}"
id="imageSource"
                  label="image source" required="true"
size="30" />
               <br />
               <h:message for="imageSource"
styleClass="message" />
            </h:panelGroup>

            <h:outputText />
            <h:panelGroup>
               <h:commandButton value="Upload"
action="#{imageBean.actionUpload}" />
               <br />
               <h:link
outcome="show?name=#{imageBean.currentResource}"
value="Cancel" />
            </h:panelGroup>
         </h:panelGrid>
      </h:form>
   </h:panelGroup>

</ui:define>

</ui:composition>
```

resources/comp/auth.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:composite="http://java.sun.com/jsf/composite"

xmlns:comp="http://java.sun.com/jsf/composite/comp">

<head>
   <title>(For validation only)</title>
</head>

<body>

<composite:interface>
   <composite:attribute name="role" />
</composite:interface>

<composite:implementation>
   <h:panelGroup rendered="#{!userSession.loggedIn
      || (!empty cc.attrs.role and
userSession.user.userRole.id != cc.attrs.role)}">
      <h:outputText value="You are not authorized to view
this page." />
   </h:panelGroup>
</composite:implementation>

</body>

</html>
```

resources/comp/phytochemicalList.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
```

```
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:composite="http://java.sun.com/jsf/composite"

xmlns:comp="http://java.sun.com/jsf/composite/comp">

<head>
    <title>(For validation only)</title>
</head>

<body>

<composite:interface>
    <composite:attribute name="phytochemicals"
required="true" />
</composite:interface>

<composite:implementation>
    <h:outputText value="No phytochemicals found."
rendered="#{empty cc.attrs.phytochemicals}" />

    <h:dataTable var="phytochemical"
value="#{cc.attrs.phytochemicals}"
        rendered="#{!empty cc.attrs.phytochemicals}">
        <h:column>
            <f:facet name="header">Name</f:facet>
            <h:link
outcome="/phytochemical/show?name=#{phytochemical.o
ntologyLocalName}"
                value="#{phytochemical.name}" />
        </h:column>

        <h:column>
            <f:facet name="header">Synonym</f:facet>
            <ui:repeat var="synonym"
value="#{phytochemical.synonym}" varStatus="status">
                <h:outputText value="#{synonym.string}" />
                <h:outputText rendered="#{!status.last}" value=",
" />
        </ui:repeat>
        </h:column>
    </h:dataTable>
</composite:implementation>

</body>

</html>
```

resources/comp/phytochemicalSearch.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:composite="http://java.sun.com/jsf/composite"

xmlns:comp="http://java.sun.com/jsf/composite/comp">

<head>
    <title>(For validation only)</title>
</head>

<body>

<composite:interface>
</composite:interface>

<composite:implementation>
    <h:form>
        <h:panelGrid columns="2">
            <h:panelGroup>
                <h:inputText
value="#{phytochemicalSearchForm.query}"
                    id="phytochemicalSearchQuery"
label="search term" size="30">
                    <f:validateLength minimum="3" />
                </h:inputText>
                <br />
                <h:message for="phytochemicalSearchQuery"
styleClass="message" />
            </h:panelGroup>

            <h:commandButton value="Search"
action="search">
                <f:attribute name="queryString"
value="#{phytochemicalSearchForm.query}" />
            </h:commandButton>
        </h:panelGrid>
    </h:form>
</composite:implementation>

</body>

</html>
```

resources/comp/phytochemicalTaxonomySelector.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:composite="http://java.sun.com/jsf/composite"

xmlns:comp="http://java.sun.com/jsf/composite/comp">

<head>
    <title>(For validation only)</title>
</head>

<body>

<composite:interface>
</composite:interface>

<composite:implementation>
    <h:selectOneListbox
value="#{phytochemicalTaxonomySelectBean.category}"
size="10"
        id="phytochemicalCategory" label="phytochemical
category" required="true">
        <f:selectItems
value="#{phytochemicalTaxonomySelectBean.categories}
" />
    </h:selectOneListbox>
    <br />
    <h:message for="phytochemicalCategory"
styleClass="message" />
</composite:implementation>

</body>

</html>
```

resources/comp/plantList.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:composite="http://java.sun.com/jsf/composite"

xmlns:comp="http://java.sun.com/jsf/composite/comp">

<head>
   <title>(For validation only)</title>
</head>

<body>

<composite:interface>
   <composite:attribute name="plants" required="true" />
</composite:interface>

<composite:implementation>
   <h:outputText value="No species found."
rendered="#{empty cc.attrs.plants}" />

   <h:dataTable var="plant" value="#{cc.attrs.plants}"
rendered="#{!empty cc.attrs.plants}">
      <h:column>
         <h:graphicImage
value="/uploads/plants/thumbs/#{plant.image[0].string1}"
            rendered="#{!empty plant.image[0].string1}" />
      </h:column>

      <h:column>
         <f:facet name="header">Scientific Name</f:facet>
         <h:link
outcome="/plant/show?name=#{plant.ontologyLocalName
}"
            value="#{plant.scientificName}" />
      </h:column>

      <h:column>
         <f:facet name="header">Local Name(s)</f:facet>
         <ui:repeat var="localName"
value="#{plant.localName}" varStatus="status">
            <h:outputText value="#{localName.string}" />
            <h:outputText rendered="#{!status.last}" value=",
"/>
      </ui:repeat>
      </h:column>
   </h:dataTable>
</composite:implementation>

</body>

</html>
```

resources/comp/plantSearch.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:composite="http://java.sun.com/jsf/composite"

xmlns:comp="http://java.sun.com/jsf/composite/comp">

<head>
   <title>(For validation only)</title>
</head>

<body>

<composite:interface>
</composite:interface>

<composite:implementation>
   <h:form>
      <h:panelGrid columns="2">
         <h:panelGroup>
            <h:inputText value="#{plantSearchForm.query}"
id="plantSearchQuery"
               label="search term" size="30">
               <f:validateLength minimum="3" />
            </h:inputText>
            <br />
            <h:message for="plantSearchQuery"
styleClass="message" />
         </h:panelGroup>

         <h:commandButton value="Search"
action="search">
            <f:attribute name="queryString"
value="#{plantSearchForm.query}" />
         </h:commandButton>
      </h:panelGrid>
   </h:form>
</composite:implementation>

</body>

</html>
```

resources/comp/plantTaxonomySelector.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:composite="http://java.sun.com/jsf/composite"

xmlns:comp="http://java.sun.com/jsf/composite/comp">

<head>
   <title>(For validation only)</title>
</head>

<body>

<composite:interface>
   <composite:attribute name="type" required="true" />
</composite:interface>

<composite:implementation>
   <h:panelGrid columns="3">
      <h:panelGroup rendered="#{cc.attrs.type !=
'group'}">
         <h:outputText value="Group" />
         <br />
         <h:selectOneListbox
value="#{plantTaxonomySelectBean.group}" size="10"
id="groupList"
            label="species group" required="true">
            <f:selectItems
value="#{plantTaxonomySelectBean.groups}" />
            <f:ajax render="familyList genusList" />
         </h:selectOneListbox>
      </h:panelGroup>

      <h:panelGroup rendered="#{cc.attrs.type == 'genus'
|| cc.attrs.type == 'species'}">
         <h:outputText value="Family" />
         <br />
         <h:selectOneListbox
value="#{plantTaxonomySelectBean.family}" size="10"
id="familyList"
            label="species family" required="true">
```

```
            <f:selectItems
value="#{plantTaxonomySelectBean.families}" />
            <f:ajax render="genusList" />
          </h:selectOneListbox>
      </h:panelGroup>

      <h:panelGroup rendered="#{cc.attrs.type ==
'species'}">
          <h:outputText value="Genus" />
          <br />
          <h:selectOneListbox
value="#{plantTaxonomySelectBean.genus}" size="10"
id="genusList"
            label="species genus" required="true">
            <f:selectItems
value="#{plantTaxonomySelectBean.genera}" />
          </h:selectOneListbox>
      </h:panelGroup>
   </h:panelGrid>

   <h:message for="groupList" styleClass="message"
/><h:outputText value=" " />
   <h:message for="familyList" styleClass="message"
/><h:outputText value=" " />
   <h:message for="genusList" styleClass="message" />
</composite:implementation>

</body>

</html>
```

resources/comp/resourceRename.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:composite="http://java.sun.com/jsf/composite"

xmlns:comp="http://java.sun.com/jsf/composite/comp">

<head>
   <title>(For validation only)</title>
</head>

<body>

<composite:interface>
</composite:interface>

<composite:implementation>
   <h:panelGroup rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}">
      <h:form>
         <h:panelGrid columns="2">
            <h:outputText value="Name"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputText
value="#{taxonomyRenameForm.newName}" id="name"
label="name"
                  required="true" size="50">
                  <f:validator validatorId="existsValidator" />
               </h:inputText>
               <br />
               <h:message for="name"
styleClass="message" />
            </h:panelGroup>

            <h:outputText />
            <h:commandButton value="Edit"
action="#{taxonomyRenameForm.actionEdit}" />
         </h:panelGrid>
      </h:form>
   </h:panelGroup>
</composite:implementation>

</body>

</html>
```

user/edit.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
   xmlns:h="http://java.sun.com/jsf/html"
   xmlns:f="http://java.sun.com/jsf/core"
   xmlns:ui="http://java.sun.com/jsf/facelets"
   xmlns:comp="http://java.sun.com/jsf/composite/comp"
   template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">Edit Account Details</ui:define>

<ui:define name="body">

   <comp:auth />

   <h:panelGroup rendered="#{userSession.loggedIn}">
      <h:form>
         <h:panelGrid columns="2">
            <h:outputText value="Password"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputSecret
value="#{userForm.password}" binding="#{password}"
                  id="password" label="password"
size="30">
                  <f:validateLength minimum="3" />
               </h:inputSecret>
               <br/>
               <h:outputText value="(Leave blank to retain
old password.)"
                  styleClass="message_info" />
               <h:message for="password"
styleClass="message" />
            </h:panelGroup>

            <h:outputText value="Retype Password"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputSecret id="password2" size="30">
                  <f:validator
validatorId="passwordValidator" />
                  <f:attribute name="password"
value="#{password.value}" />
               </h:inputSecret>
               <br/>
               <h:message for="password2"
styleClass="message" />
            </h:panelGroup>

            <h:outputText value="Last Name"
styleClass="field_name" />
            <h:panelGroup>
               <h:inputText
value="#{userSession.user.lastName}" id="lastName"
                  label="last name" required="true"
size="30">
               </h:inputText>
               <br/>
               <h:message for="lastName"
styleClass="message" />
            </h:panelGroup>

            <h:outputText value="First Name"
styleClass="field_name" />
            <h:panelGroup>
```

```
                <h:inputText
value="#{userSession.user.firstName}" id="firstName"
                    label="first name" required="true"
size="30">
                </h:inputText>
                <br/>
                <h:message for="firstName"
styleClass="message" />
            </h:panelGroup>

            <h:outputText />
            <h:commandButton value="Save"
action="#{userForm.saveUser}" />
        </h:panelGrid>
    </h:form>
  </h:panelGroup>

</ui:define>

</ui:composition>
```

user/index.jsp

```
<% response.sendRedirect("show.jsf"); %>
```

user/login.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">Log In</ui:define>

<ui:define name="body">

  <h:panelGroup rendered="#{!userSession.loggedIn}">
    <h:form>
      <h:panelGrid columns="2">
        <h:outputText value="Username"
styleClass="field_name" />
        <h:inputText value="#{userForm.username}"
size="30" />

        <h:outputText value="Password"
styleClass="field_name" />
        <h:inputSecret value="#{userForm.password}"
size="30" />

        <h:outputText />
        <h:commandButton value="Log in"
action="#{userForm.login}" />
      </h:panelGrid>
    </h:form>
  </h:panelGroup>

  <h:panelGroup rendered="#{userSession.loggedIn}">
    <h:outputText value="You are logged in as
#{userSession.user.firstName}
#{userSession.user.lastName}
(#{userSession.user.userRole.name})." />
    <h:form>
      <h:commandButton value="Log out"
action="#{userForm.logout}" />
    </h:form>
  </h:panelGroup>

</ui:define>

</ui:composition>
```

user/login-failed.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">Log In</ui:define>

<ui:define name="body">

  <h:panelGroup rendered="#{!userSession.loggedIn}">
    <h:outputText value="Wrong username/password."
styleClass="message" />
    <h:form>
      <h:panelGrid columns="2">
        <h:outputText value="Username"
styleClass="field_name" />
        <h:inputText value="#{userForm.username}"
size="30" />

        <h:outputText value="Password"
styleClass="field_name" />
        <h:inputSecret value="#{userForm.password}"
size="30" />

        <h:outputText />
        <h:commandButton value="Log in"
action="#{userForm.login}" />
      </h:panelGrid>
    </h:form>
  </h:panelGroup>

  <h:panelGroup rendered="#{userSession.loggedIn}">
    <h:outputText value="You are logged in as
#{userSession.user.firstName}
#{userSession.user.lastName}
(#{userSession.user.userRole.name})." />
    <h:form>
      <h:commandButton value="Log out"
action="#{userForm.logout}" />
    </h:form>
  </h:panelGroup>

</ui:define>

</ui:composition>
```

user/show.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:ui="http://java.sun.com/jsf/facelets"
  xmlns:comp="http://java.sun.com/jsf/composite/comp"
  template="../WEB-INF/templates/main.xhtml">

<ui:define name="title">Manage Account</ui:define>

<ui:define name="body">

  <comp:auth />

  <h:panelGroup rendered="#{userSession.loggedIn}">
    <h:panelGroup styleClass="admin_options">
      <h:link outcome="edit" value="Edit Account
Details" />
    </h:panelGroup>

    <h:panelGrid columns="2" styleClass="details">
      <h:outputText value="Username"
styleClass="field_name" />
      <h:outputText
value="#{userSession.user.username}" />

      <h:outputText value="User Role"
styleClass="field_name" />
      <h:outputText
value="#{userSession.user.userRole.name}" />
```

```
            <h:outputText value="Last Name"
styleClass="field_name" />
            <h:outputText
value="#{userSession.user.lastName}" />

            <h:outputText value="First Name"
styleClass="field_name" />
            <h:outputText
value="#{userSession.user.firstName}" />
        </h:panelGrid>
    </h:panelGroup>

</ui:define>

</ui:composition>
```

WEB-INF/templates/main.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:fn="http://java.sun.com/jsp/jstl/functions"
      xmlns:fnc="http://hanapin-sp-2.com/fnc"

xmlns:comp="http://java.sun.com/jsf/composite/comp">

<h:head>
    <title><ui:insert name="title">Home</ui:insert> |
HANAPIN-SP 2.0</title>
    <h:outputStylesheet name="960.css" library="css" />
    <h:outputStylesheet name="reset.css" library="css" />
    <h:outputStylesheet name="text.css" library="css" />
    <h:outputStylesheet name="style.css" library="css" />
</h:head>

<h:body>
    <div class="container_16" id="body">
        <div class="grid_16" id="header">
            <h:graphicImage library="images"
name="header.png" />
        </div>

        <div class="grid_16" id="menu">
            <ul>
                <li>
                    <h:outputLink
value="#{request.contextPath}/"

    styleClass="#{fn:endsWith(request.requestURI,
fnc:concat(request.contextPath, '/home.jsf')) ? 'active' :
'inactive'}">Home</h:outputLink>
                </li>
                <li>
                    <h:outputLink
value="#{request.contextPath}/plant/"

    styleClass="#{fn:startsWith(request.requestURI,
fnc:concat(request.contextPath, '/plant/')) ? 'active' :
'inactive'}">Species</h:outputLink>
                </li>
                <li>
                    <h:outputLink
value="#{request.contextPath}/phytochemical/"

    styleClass="#{fn:startsWith(request.requestURI,
fnc:concat(request.contextPath, '/phytochemical/')) ?
'active' : 'inactive'}">Phytochemicals</h:outputLink>
                </li>
                <li>
                    <h:outputLink
value="#{request.contextPath}/contact.jsf"

    styleClass="#{fn:endsWith(request.requestURI,
fnc:concat(request.contextPath, '/contact.jsf')) ? 'active' :
'inactive'}">Contact Us</h:outputLink>
                </li>
                <li>
                    <h:outputLink
value="#{request.contextPath}/user/login.jsf"
rendered="#{!userSession.loggedIn}"

    styleClass="#{fn:startsWith(request.requestURI,
fnc:concat(request.contextPath, '/user/')) ? 'active' :
'inactive'}">Log In</h:outputLink>
                </li>
                <li>
                    <h:outputLink
value="#{request.contextPath}/admin/"
rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 1}"

    styleClass="#{fn:startsWith(request.requestURI,
fnc:concat(request.contextPath, '/admin/')) ? 'active' :
'inactive'}">Administration</h:outputLink>
                </li>
                <li>
                    <h:outputLink
value="#{request.contextPath}/curation/"
rendered="#{userSession.loggedIn and
userSession.user.userRole.id == 2}"

    styleClass="#{fn:startsWith(request.requestURI,
fnc:concat(request.contextPath, '/curation/')) ? 'active' :
'inactive'}">Curation</h:outputLink>
                </li>
                <li>
                    <h:outputLink
value="#{request.contextPath}/user/"
rendered="#{userSession.loggedIn}"

    styleClass="#{fn:startsWith(request.requestURI,
fnc:concat(request.contextPath, '/user/')) ? 'active' :
'inactive'}">Manage Account</h:outputLink>
                </li>
            </ul>
        </div>

        <div class="grid_3" id="admin_panel">
             

            <h:panelGroup rendered="#{userSession.loggedIn
and userSession.user.userRole.id == 2}">
                <h:panelGroup
rendered="#{fn:startsWith(request.requestURI,
fnc:concat(request.contextPath, '/plant/'))}">
                    <ul>
                        <li><h:outputLink
value="#{request.contextPath}/plant/taxonomy/add.jsf">Ne
w Group/Family/Genus</h:outputLink></li>
                        <li><h:outputLink
value="#{request.contextPath}/plant/add.jsf">New
Species</h:outputLink></li>
                    </ul>
                </h:panelGroup>

                <h:panelGroup
rendered="#{fn:startsWith(request.requestURI,
fnc:concat(request.contextPath, '/phytochemical/'))}">
                    <ul>
                        <li><h:outputLink
value="#{request.contextPath}/phytochemical/taxonomy/a
dd.jsf">New Phytochemical Category</h:outputLink></li>
                        <li><h:outputLink
value="#{request.contextPath}/phytochemical/add.jsf">Ne
w Phytochemical</h:outputLink></li>
                    </ul>
```

```
            </h:panelGroup>
        </h:panelGroup>

        <h:panelGroup
rendered="#{userSession.loggedIn}">
            <h:outputText value="You are logged in as
#{userSession.user.firstName}
#{userSession.user.lastName}
(#{userSession.user.userRole.name})." />
            <h:form>
                <h:commandLink value="Log out"
action="#{userForm.logout}" />
            </h:form>
        </h:panelGroup>
    </div>

    <div class="grid_13" id="content">
        <h1><ui:insert name="title"></ui:insert></h1>
        <ui:insert name="body"></ui:insert>
    </div>

    <div class="grid_16" id="footer">
        Copyright &copy; 2011 University of the
Philippines, Manila. All Rights Reserved.
    </div>
  </div>
</h:body>

</html>
```

WEB-INF/faces-config.xml

```
<?xml version="1.0"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-
facesconfig_2_0.xsd"
  version="2.0">
  <application>
    <message-bundle>messages</message-bundle>
  </application>
  <converter>
    <description>userRoleConverter</description>
    <display-name>userRoleConverter</display-name>
    <converter-for-
class>model.user.UserRole</converter-for-class>
    <converter-
class>model.user.UserRoleConverter</converter-class>
  </converter>
  <navigation-rule>
  <from-view-id>*</from-view-id>
  <navigation-case>
      <from-outcome>logged-out</from-outcome>
      <to-view-id>/home.xhtml</to-view-id>
      <redirect />
  </navigation-case>
  </navigation-rule>
  <navigation-rule>
    <display-name>plant/taxonomy/add.xhtml</display-
name>
    <from-view-id>/plant/taxonomy/add.xhtml</from-
view-id>
    <navigation-case>
      <from-outcome>add-success</from-outcome>
      <to-view-id>/plant/list.xhtml</to-view-id>
    </navigation-case>
  </navigation-rule>
  <navigation-rule>
    <display-name>plant/taxonomy/edit.xhtml</display-
name>
    <from-view-id>/plant/taxonomy/edit.xhtml</from-
view-id>
    <navigation-case>
      <from-outcome>edit-success</from-outcome>
      <to-view-id>/plant/list.xhtml</to-view-id>
    </navigation-case>
  </navigation-rule>
  <navigation-rule>
    <display-
name>phytochemical/taxonomy/add.xhtml</display-
name>
    <from-view-
id>/phytochemical/taxonomy/add.xhtml</from-view-id>
    <navigation-case>
      <from-outcome>add-success</from-outcome>
      <to-view-id>/phytochemical/list.xhtml</to-view-id>
    </navigation-case>
  </navigation-rule>
  <navigation-rule>
    <display-
name>phytochemical/taxonomy/edit.xhtml</display-
name>
    <from-view-
id>/phytochemical/taxonomy/edit.xhtml</from-view-id>
    <navigation-case>
      <from-outcome>edit-success</from-outcome>
      <to-view-id>/phytochemical/list.xhtml</to-view-id>
    </navigation-case>
  </navigation-rule>
  <navigation-rule>
    <display-name>user/login.xhtml</display-name>
    <from-view-id>/user/login.xhtml</from-view-id>
    <navigation-case>
      <from-outcome>logged-in</from-outcome>
      <to-view-id>/home.xhtml</to-view-id>
      <redirect />
    </navigation-case>
  </navigation-rule>
  <navigation-rule>
    <display-name>user/login-failed.xhtml</display-
name>
    <from-view-id>/user/login-failed.xhtml</from-view-id>
    <navigation-case>
      <from-outcome>logged-in</from-outcome>
      <to-view-id>/home.xhtml</to-view-id>
      <redirect />
    </navigation-case>
  </navigation-rule>
  <validator>
    <display-name>existsValidator</display-name>
    <validator-id>existsValidator</validator-id>
    <validator-class>model.ExistsValidator</validator-
class>
  </validator>
  <validator>
    <description></description>
    <display-name>imageUploadValidator</display-
name>
    <validator-id>imageUploadValidator</validator-id>
    <validator-
class>model.image.ImageUploadValidator</validator-
class>
  </validator>
  <validator>
    <display-name>ontologyUploadValidator</display-
name>
    <validator-id>ontologyUploadValidator</validator-id>
    <validator-
class>model.OntologyUploadValidator</validator-class>
  </validator>
  <validator>
    <description>passwordValidator</description>
    <display-name>passwordValidator</display-name>
    <validator-id>passwordValidator</validator-id>
    <validator-
class>model.PasswordValidator</validator-class>
  </validator>
</faces-config>
```

WEB-INF/functions.taglib.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE facelet-taglib PUBLIC
  "-//Sun Microsystems, Inc.//DTD Facelet Taglib 1.0//EN"

"https://facelets.dev.java.net/source/browse/*checkout*/facelets/src/etc/facelet-taglib_1_0.dtd">
<facelet-taglib xmlns="http://java.sun.com/JSF/Facelet">
    <namespace>http://hanapin-sp-2.com/fnc</namespace>
    <function>
        <function-name>concat</function-name>
        <function-class>webapp.Functions</function-class>
        <function-signature>
            java.lang.String concat(java.lang.String, java.lang.String)
        </function-signature>
    </function>
</facelet-taglib>
```

WEB-INF/web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  version="2.5">
  <!-- The bare minimum needed for JSF 2.0 is a servlet 2.5
      declaration and the mapping for the FacesServlet.
      Setting PROJECT_STAGE to Development is highly recommended
      during initial development so that you get more helpful
      error messages.

      From JSF 2.0 tutorial at http://www.coreservlets.com/JSF-Tutorial/jsf2/
  -->
  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
  </servlet-mapping>
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <context-param>
    <param-name>facelets.LIBRARIES</param-name>
    <param-value>/WEB-INF/functions.taglib.xml</param-value>
  </context-param>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <filter>
    <filter-name>MyFacesExtensionsFilter</filter-name>
    <filter-class>org.apache.myfaces.webapp.filter.ExtensionsFilter</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>MyFacesExtensionsFilter</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
  </filter-mapping>
  <servlet>
    <servlet-name>imageServlet</servlet-name>
    <servlet-class>model.image.ImageServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>imageServlet</servlet-name>
    <url-pattern>/uploads/*</url-pattern>
  </servlet-mapping>
</web-app>
```

contact.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    template="WEB-INF/templates/main.xhtml">

<ui:define name="title">Contact Us</ui:define>
<ui:define name="body">
    <p>If you have any questions, comments, or suggestions, or if you wish to become a member of the HANAPIN-SP 2.0 curation team, you may send an email to hanapinsp2admin@upm.edu.ph.</p>
</ui:define>

</ui:composition>
```

home.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    template="WEB-INF/templates/main.xhtml">

<ui:define name="title">Home</ui:define>
<ui:define name="body">
    <p>Welcome to HANAPIN-SP 2.0.</p>

    <h6>About HANAPIN-SP 2.0</h6>
    <p>A natural product is a chemical compound or a substance produced by living organisms. Natural products are often used in the treatment of life-threatening conditions, such as cancer or malaria. The study of natural products in the Philippines is an active field, due to the archipelago's abundance and variety of plants and organisms.</p>
    <p>Consolidation and archiving of studies on natural products is an ongoing and problematic task for researchers. The structures of the data on the study and experimentation of natural products are often varying from researcher to researcher. More importantly, there is no centralized repository for such studies. Studying about existing research usually involves personally and individually contacting the respective researcher/s, instead of simply searching a centralized database, resulting to more time and effort costs. Research work is often duplicated because researchers do not know that the research has already been carried out.</p>
    <p>In answer to the stated problems, Health Application for Natural Products Information System for Plants (HANAPIN-SP) 2.0 provides a centralized repository and interactive system for storing and searching available data on natural products.</p>
</ui:define>

</ui:composition>
```

index.jsp

```
<% response.sendRedirect("home.jsf"); %>
```