

# EDHIBOU'S USER GUIDE



What is EdHibou ?.....	2
Key Features.....	2
Architecture .....	2
An Ontology of Graphical Components.....	2
System requirements.....	3
Tutorial : An example of EdHibou application .....	4
Downloading EdHibou.....	4
Settings Configuration.....	5
Editing the interface ontology.....	6
Presentation of the ontology.....	6
Adding “Movie” module.....	8
Adding “Movie” panels.....	10
Adding “Sport” module and Panels.....	13
Deploying EdHibou.....	14
Presentation of the available panels.....	14
OWLEditor.....	14
EmptyContainer.....	15
HTMLPanel.....	15
SimilarInstancesPanel.....	16
CommentPanel.....	16
Contributors.....	17



# What is EdHIBOU ?

The Semantic Web is becoming more and more a reality, as the required technologies have reached an appropriate level of maturity. However, at this stage, it is important to provide tools facilitating the use and deployment of these technologies by end-users. EdHIBOU is an automatically generated, ontology-based graphical user interface that integrates in a semantic portal. The particularity of EdHIBOU is that it makes use of OWL reasoning capabilities to provide intelligent features, such as decision support, upon the underlying ontology.

## ***Key Features***

- Easy OWL instance edition by means of user-friendly forms
- Intelligent behaviour provided by embedded OWL reasoner
- Interface personalization capabilities thanks to an ontology-driven GUI generation

## ***Architecture***

- EdHIBOU implements a Model-View-Controller architecture pattern and was developed using the Google Web Toolkit Java AJAX programming framework.
- The application knowledge model is externalized to a knowledge server (K-OWL) that performs reasoning on OWL ontologies.
- All configuration information is also pushed into an OWL ontology, thus implementing a model-driven design approach.

## ***An Ontology of Graphical Components***

- All configuration information is placed in a separate ontology that contains an exhaustive description of the view of the application. The application thus manages an implementation-independent model of the user interface.
- Interface personalization can be achieved by:
  - extending the default configuration ontology,
  - adding new graphical components,
  - adding a custom CSS stylesheet.

# System requirements

Java 1.5 or later,

Apache Tomcat 5.5 or later,

Internet browser that support JavaScript (Mozilla Firefox, Internet Explorer, Opera...)

# Tutorial : An example of EdHibou application

This tutorial used the ontology editor “Protégé” 3.3.1 (<http://protege.stanford.edu/>) and an Apache Tomcat Server. In this tutorial, we want EdHIBOU to edit instances in two different ontologies, the first about cinema, and the second about sport (figure 1).

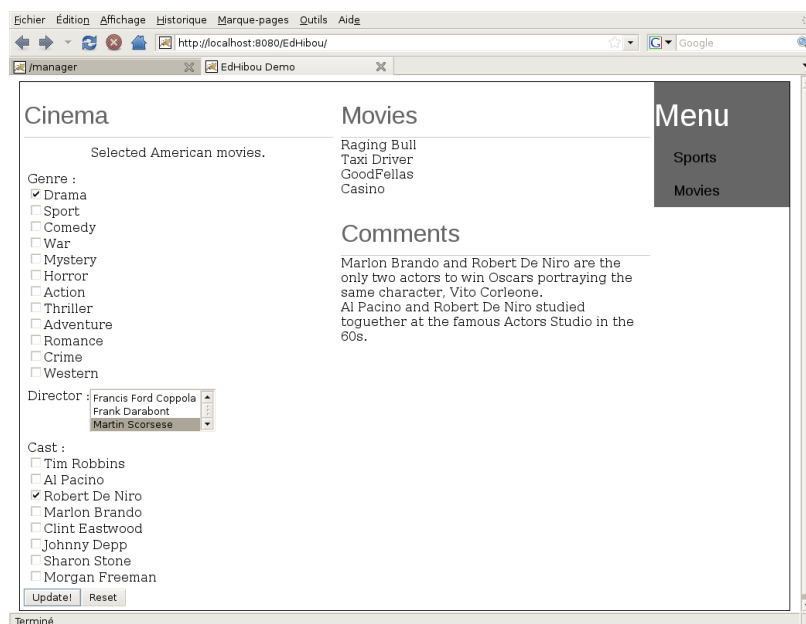


Figure 1: EdHibou Application built in this tutorial.

This tutorial has four main steps :

- downloading EdHIBOU,
- configuring settings in the “config.xml” file,
- implementing the interface ontology,
- deploying EdHIBOU.

## Downloading EdHIBOU

We will use the empty version available in the zip file “EdHibou.zip” which is available on Kasmir Project's download page (<http://labotalc.loria.fr/~kasimir/downloads/>). Let's unzip it!

The folder (figure 2) contains all needed element for EdHIBOU :

- two folders (META-INF and WEB-INF) containing Java libraries and running information for the server,
- HTML, XML, CSS and Javascript files that contained instruction for the client side application (particularly “config.xml”),
- Two OWL ontology files (“interface.owl” and “interface2.owl”) : we will edit the first to create our application whereas the second the result we will obtain at the end of this tutorial.

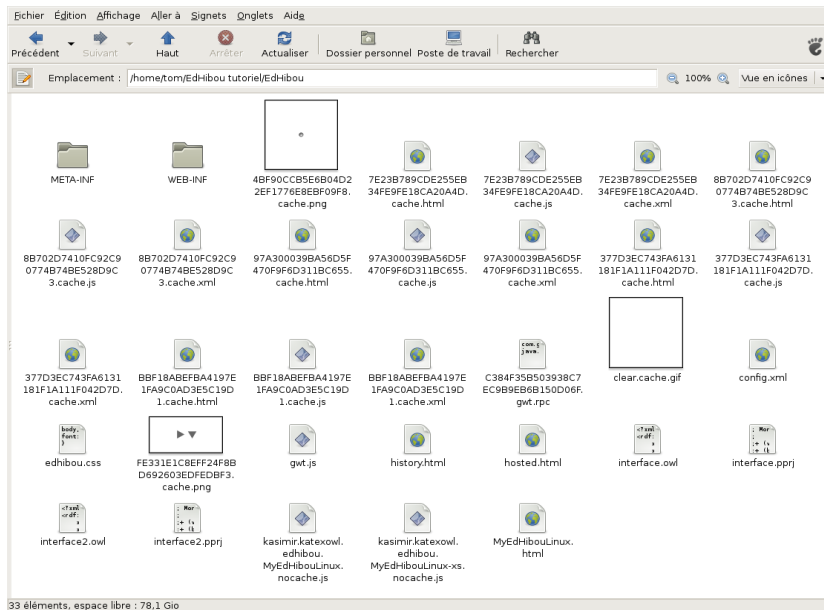


Figure 2: EdHibou root folder after unzipping.

## Settings Configuration

We will now configure the used elements by editing “config.xml”.

```

<Config>
  <HTTPClient>
    <Class>
      kasimir.katexowl.edhibou.server.kserver.KowlInternClient
    </Class>
  </HTTPClient>
  <Interface>
    <Address>
      interface.owl
    </Address>
    <NameSpace>
      http://kasimir.loria.fr/2008/07/interface#
    </NameSpace>
  </Interface>
</Config>

```

Figure 3: "config.xml" file content.

The first part in the “HTTPClient” tag defines the knowledge server we will use. This one has to implement the Sparql protocol (<http://www.w3.org/TR/rdf-sparql-protocol/>), and a few more functions such as adding instances, properties in ontology. Until now KOWL (<http://katexowl.loria.fr/>) is the only compatible server. We can use it as an internal server by choosing the class “kasimir.katexowl.edhibou.server.kserver.KowlInternClient”. KOWL can be used externally via HTTP protocol. This point will explained in the Developer Guide.

The second part defines the interface ontology. The first tag contains the ontology URL (or the name of the file if the ontology is at the EdHIBOU folder root) and the second contains the interface ontology namespace.

In this tutorial, we can let default settings.

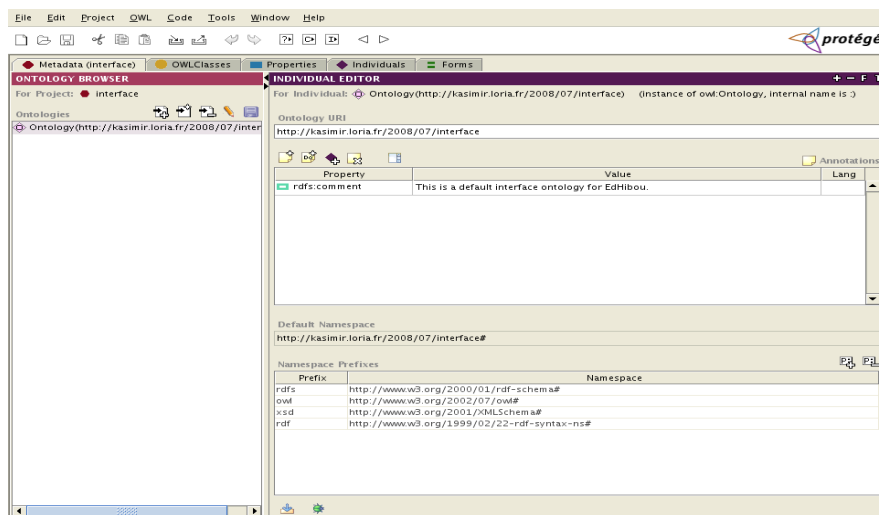
## ***Editing the interface ontology***

### **Presentation of the ontology**

We will now edit the OWL ontology that contains the interface and describes how the application works. This ontology is the one that has been named in “config.xml”, and it has been written in OWL.

To edit the file, we will use Protégé.

The final result can be downloaded on Kasmir Project's download page (<http://labotalc.loria.fr/~kasimir/downloads/>) and is present in the archive, named “interface2.owl”.



*Figure 4: Editing the interface ontology.*

We can see here the classes used in this ontology.

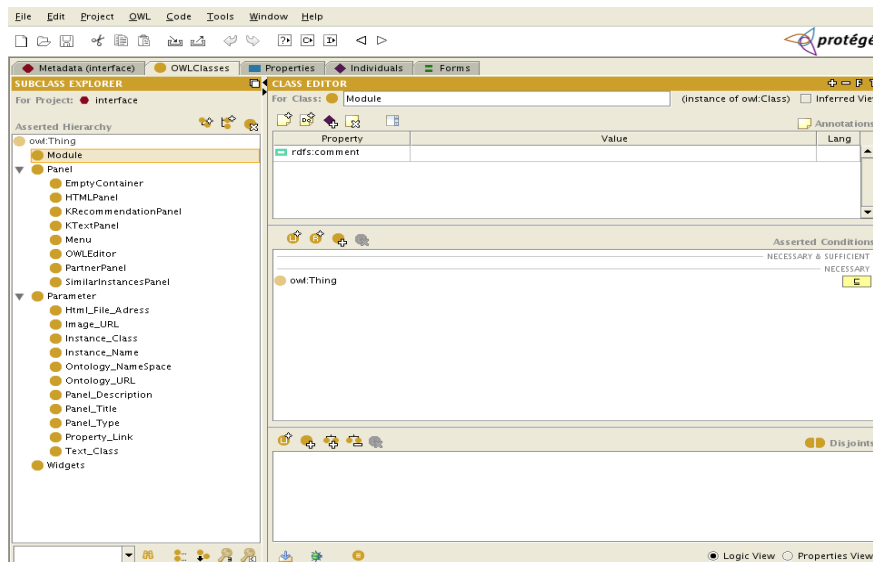


Figure 5: Subclasses of owl:Thing.

Each subclass of the class owl:Thing represents a part of the application. The first one named “Module” can be viewed as a frame that contains a set of functions which the results are displayed together and are generally linked (green in the figure 6). In this tutorial, we will have two modules, the first concerning the cinema ontology and the second concerning the sport ontology. “Panels” (blue) are functions that the results are displayed, for example a OWL editor, a text displayer, a menu... They can be controlled thanks to the class “Parameter”. At last, “Widgets” (red) represents components of the forms.

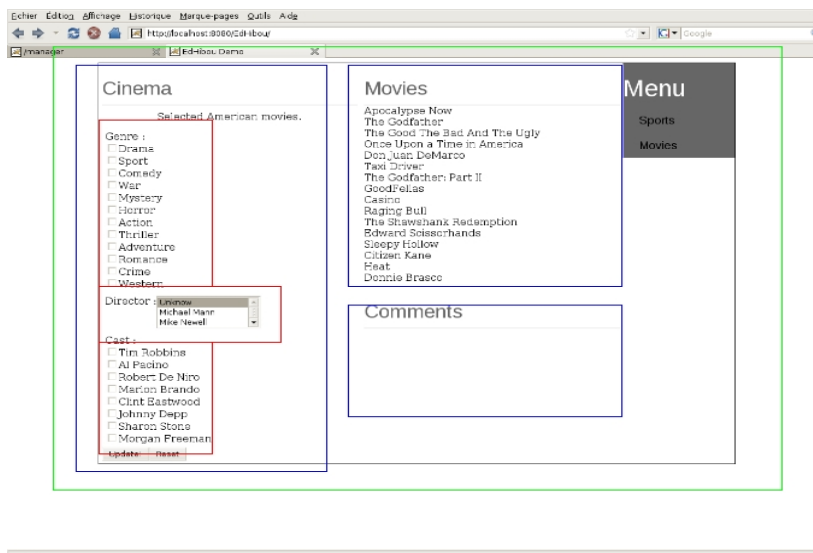


Figure 6: EdHibou's module (Green), panels (blue) and widgets (red).

We can see that the “Widget” class already contains individuals. They represent the different form components that are exploited. Adding components is explained in the developers guide.

## Adding “Movie” module

We will edit an instance in the ontology “cinema.owl”. At first, we create a module by adding an individual in “Module” class which will contain our future panels.

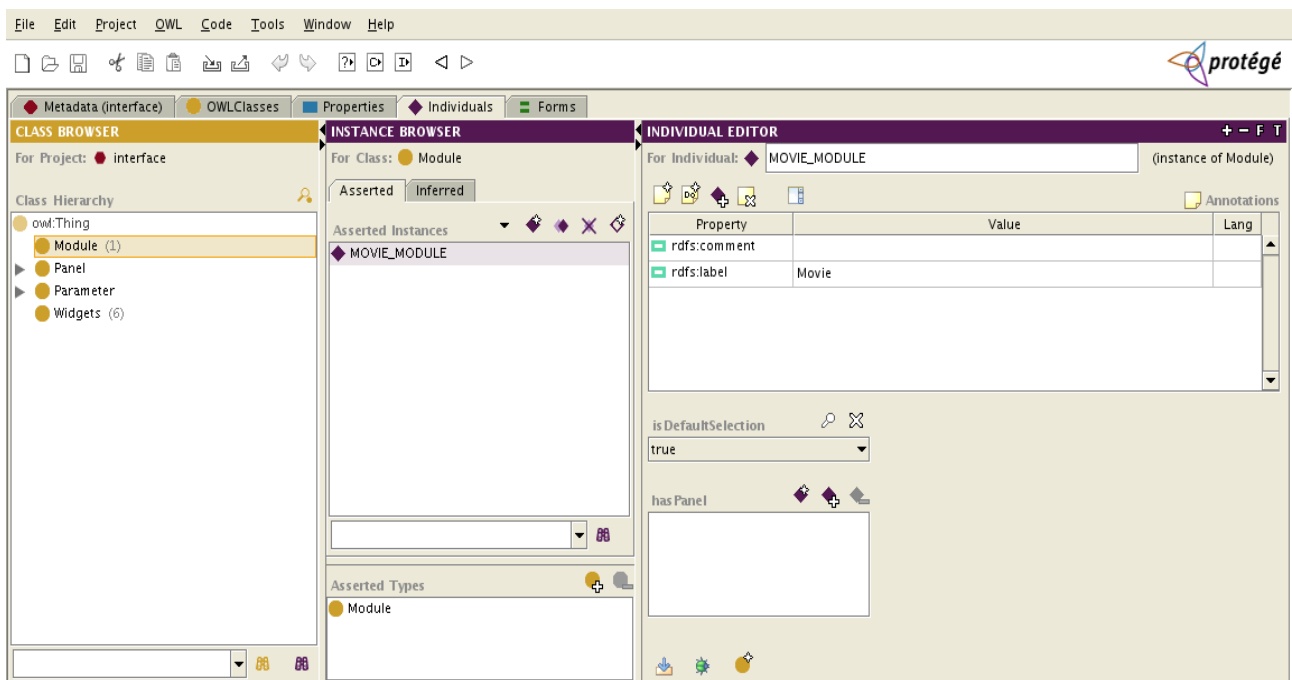


Figure 7: Adding a module.

We can note that a rdfs:label has to be fill, to appear in the future menu.

Then, in the same manner, we create an empty panel called “MOVIES\_GENERAL\_CONTAINER” in the class “EmptyContainer”. This one will not directly appear but it represents the general page.

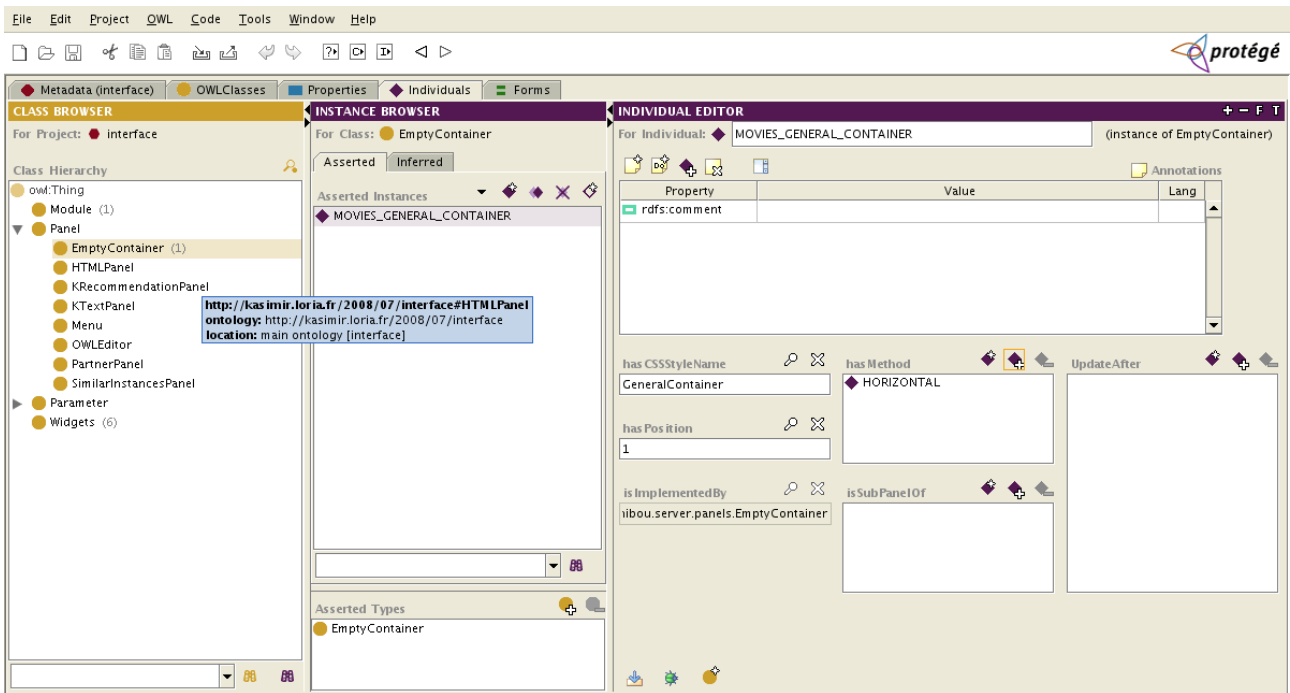


Figure 8: Adding a panel.

To link the previously created module with this panel, we have to use the property “hasPanel” which has the class “Module” as domain and “Panel” as range.

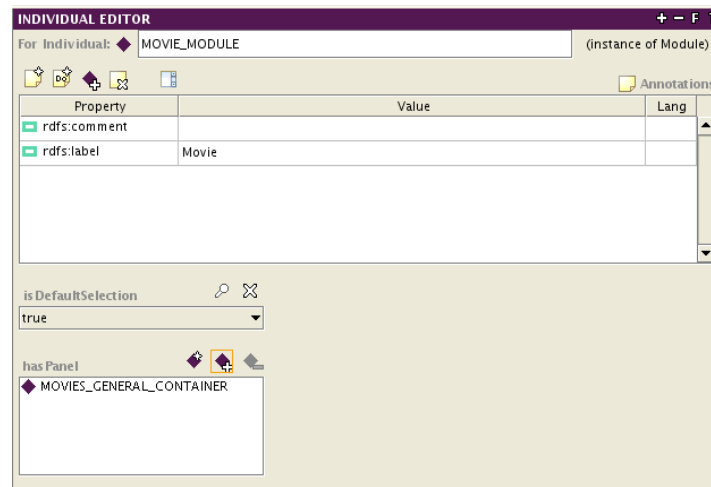
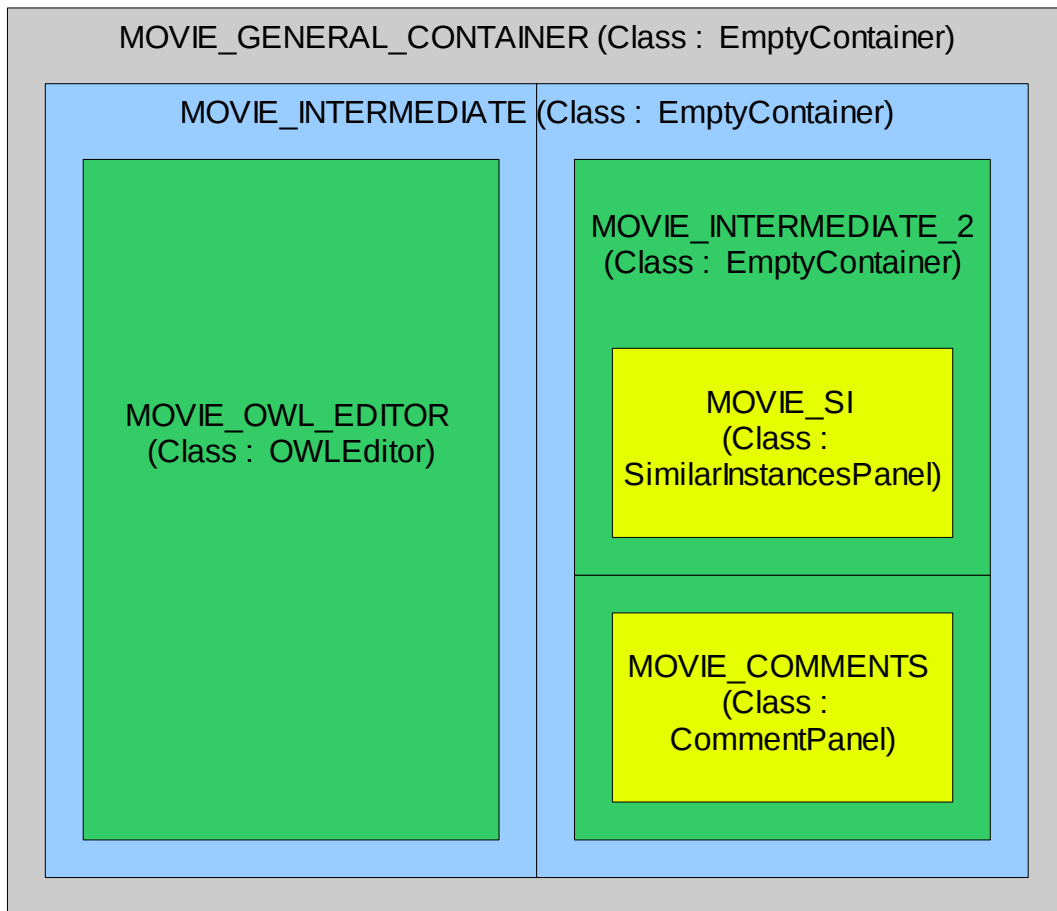


Figure 9: Editing a "Module" class individual.

The datatype property “isDefaultSelection” has to be set to true. It decides which is the first module to appear on the final website.

## Adding “Movie” panels

Now, we will try to build the application which can be viewed on this way :



*Figure 10: Panels view of the first module.*

The panel we have just built is represented in grey. We will create some other panels, each one has a particular function. To place the panel, we will use the property “isSubPanelOf” that means that the panel A is in the panel B, and the property “hasPosition” which range is an integer that represents the panel place in its super-panel.

So we create a new individual called “MOVIE\_INTERMEDIATE” in “EmptyContainer” class. We set “hasPosition” to “1” and “isSubPanelOf” to “MOVIE\_GENERAL\_CONTAINER”. We can also set to “GeneralContainer” the property “hasCSSStyleName”. This will be the name of the class to use in the CSS file to adapt the view.

We want now to specify that our panel have a horizontal disposition. We can do it with the property “hasMethod”, which links “Panel” with “Parameter” class. In the sub-class “Panel\_Type”, we select the individual “HORIZONTAL”. By this way, we set the parameter that define the panel disposition.

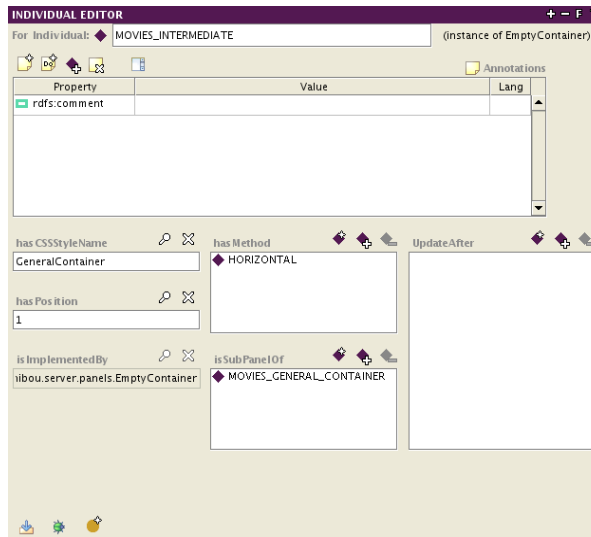


Figure 11: Adding properties while editing an "Panel" class individual.

On the same way, we create the individual "MOVIE\_INTERMEDIATE\_2":

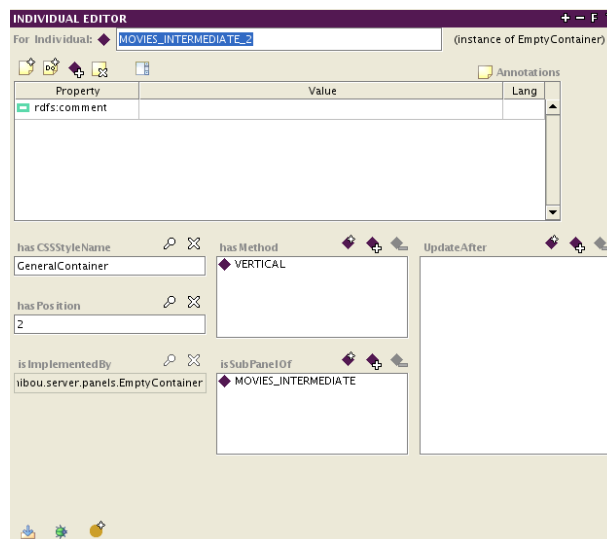


Figure 12: Adding another "EmptyContainer" individual.

Now we will add our instance editor. Let's create an individual called "MOVIE\_OWL\_EDITOR" in the class "OWLEditor". We can set "hasPosition" to "1", "isSubPanelOf" to "MOVIE\_INTERMEDIATE", and "hasCSSStyleName" to "PanelOnto".

This particular panel needs 6 parameters we will have to create in the sub-class of the class "Parameter".

- The name of the instance we want to edit : in the sub-class "Instance\_Name", create an individual called "INSTANCE\_NAME\_MY\_IND". The name is given by the property "hasValue" of this individual, that we can set to "my\_individual".

- The class of the instance : we create “MOVIE\_INSTANCE\_CLASS” in “Instance\_Class” and set “hasValue” to “Movie” which is the name of the class in the cinema ontology.
- The URL of the edited ontologie : we create “MOVIE\_ONTOLOGY\_URL” in “Ontology\_URL” and set “hasValue” to “<http://labotalc.loria.fr/~kasimir/downloads/owl/cinema.owl>” (This must be between quotes).
- The namespace of the edited ontologie : we create “MOVIE\_ONTOLOGY\_NS” in “Ontology\_NameSpace” and set “hasValue” to “<http://www.owl-ontologies.com/Ontology1224688205.owl#>” (This must be between quotes).
- The title that will appear on the view : we create “MOVIE\_EDITION\_TITLE” in “Panel\_Title” and set “hasValue” to “Cinema”.
- The description that will appear on the view : we create “MOVIE\_DESCRIPTION” in “Panel\_Title” and set “hasValue” to “Selected American movies.”.

These 6 parameters have then to be linked to the individual “MOVIE\_OWL\_EDITOR” by the property “hasMethod”.

In the same way, we can create the two last panels. “MOVIE\_SI” is a individual of “SimilarInstancesPanel”. It will select the instances that have the same properties that the edited one in the previously created panel. Logically, it has to be updated after the editor. That why we will use the property “UpadteAfter” which links a panel to another. In “MOVIE\_SI”, “UpadteAfter” is valued to “MOVIE\_OWL\_EDITOR”.

Then we can set “hasPosition” to “1”, “isSubPanelOf” to “MOVIE\_INTERMEDIATE\_2” and “hasCSSStyleName” to “PanelOnto”. Here is a table with the individuals we have to create in “Parameter” and linked to “MOVIE\_SI” with the property “hasMethod”, knowing that the already created one can be reused :

Name	Class	“hasValue”
MOVIE_INSTANCE_CLASS	Instance_Class	“Movie”
INSTANCE_NAME_MY_IND	Instance_Name	“my_individual”
MOVIE_ONTOLOGY_URL	Ontology_URL	“ <a href="http://labotalc.loria.fr/~kasimir/downloads/owl/cinema.owl">http://labotalc.loria.fr/~kasimir/downloads/owl/cinema.owl</a> ”
MOVIE_ONTOLOGY_NS	Ontology_NameSpace	“ <a href="http://www.owl-ontologies.com/Ontology1224688205.owl#">http://www.owl-ontologies.com/Ontology1224688205.owl#</a> ”
MOVIE_SI_TITLE	Panel_Title	“Movie”

Finally, we create the last panel in the class “CommentPanel”. It aims at listing the labels of the class “Comment” of the cinema ontology. This class is linked to the class “Movie” by the property

“isCommentOf”. The same mechanism is used in Kasimir for detecting recommendation. To know more about it, edit the cinema ontology.

In the class “CommentPanel” we create the individual “MOVIE\_COMMENTS” and we set “hasPosition” to “2”, “isSubPanelOf” to “MOVIE\_INTERMEDIATE\_2” and “hasCSSStyleName” to “PanelOnto”. Here is a table with the individuals we have to create in “Parameter” and linked to “MOVIE\_COMMENTS” with the property “hasMethod”, knowing that the already created one can be reused :

Name	Class	“hasValue”
MOVIE_PROPERTY_LINK	Property_Link	“isCommentOf”
INSTANCE_NAME_MY_IND	Instance_Name	“my_ind”
MOVIE_ONTOLOGY_URL	Ontology_URL	“ <a href="http://labotalc.loria.fr/~kasimir/downloads/owl/cinema.owl">http://labotalc.loria.fr/~kasimir/downloads/owl/cinema.owl</a> ”
MOVIE_ONTOLOGY_NS	Ontology_NameSpace	“ <a href="http://www.owl-ontologies.com/Ontology1224688205.owl#">http://www.owl-ontologies.com/Ontology1224688205.owl#</a> ”
MOVIE_COMMENTS_TITLE	Panel_Title	“Comments”
MOVIE_COMMENTS_CLASS	Comments_Class	“Comment”

Now, we have just finished to create the first module of the application. If we copy the EDHIBOU folder in the “webapps” folder of Tomcat, we can see the first result.

## Adding “Sport” module and Panels

We will try to add a second module, dealing with sport.

As in the first part we have to add an individual in “Module” class, called “SPORT\_MODULE” and set the “isDefaultSelection” to false. We fill a rdfs:label (for example :”Sports”).

Now, we can create different panels from the following scheme knowing that the sport ontology has this URL : “<http://labotalc.loria.fr/~kasimir/downloads/owl/sport.owl>”, this namespace “<http://www.owl-ontologies.com/Ontology1224688205.owl#>” and that EDHIBOU has to edit an instance of the class “Sport”.

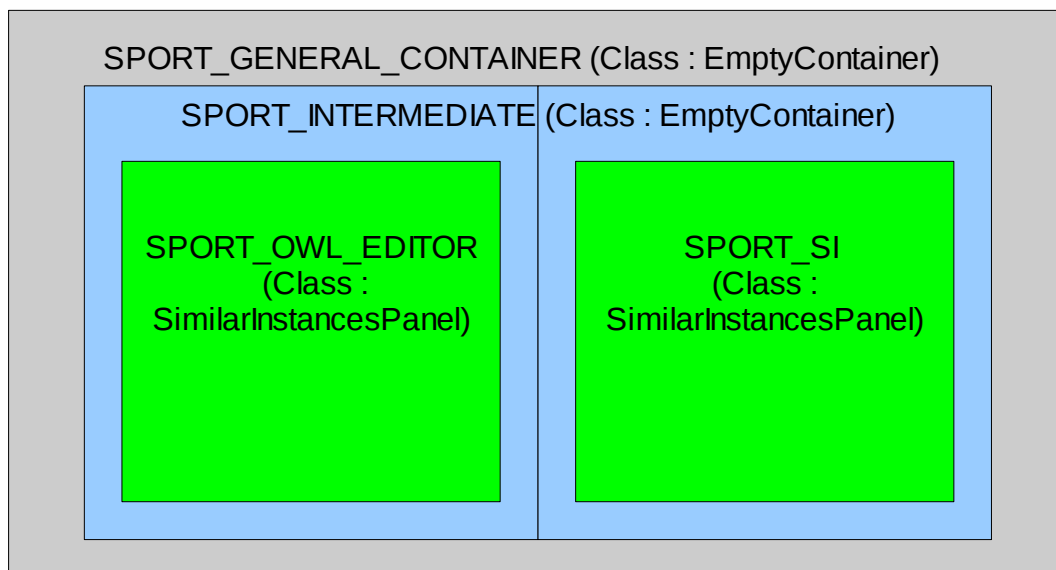


Figure 13: Panels view of the second module.

We can add all these panels, by creating and adapting the parameters.

The last step of this interface ontology edition is adding a menu, in order to navigate between modules. We create an individual in the class “Menu” which is a kind of panel. We will place it at the third place (“hasPosition” property) of “MOVIE\_INTERMEDIATE” and “SPORT\_INTERMEDIATE” (“isSubPanelOf” property). It does not need any more parameter.

### ***Deploying EdHIBOU***

To consult our resulting application, we have to copy the EdHIBOU folder into the “webapps” folder of Tomcat. The result can be view at the address <http://xxx/EdHibou> (where xxx is the Tomcat server address, for example localhost:8080 for a local server).

## **Presentation of the available panels**

### ***OWLEditor***

This panel is used to edit an instance in an OWL ontology. It aims at making easy the edition by using web forms.

Parameters :

<b>Description</b>	<b>Parameter Class</b>	<b>Example of value for the “hasValue” property</b>
The name of the instance to edit.	Instance_Name	“my_individual”
The class of the instance to edit.	Instance_Class	“Movie”
The URL of the edited ontology.	Ontology_URL	“” <a href="http://labotalc.loria.fr/~kasimir/downloads/owl/cinema.owl">http://labotalc.loria.fr/~kasimir/downloads/owl/cinema.owl</a> “”
The namespace of the edited ontology.	Ontology_NameSpace	“” <a href="http://www.owl-ontologies.com/Ontology1224688205.owl#">http://www.owl-ontologies.com/Ontology1224688205.owl#</a> “”
The title of the panel.	Panel_Title	“Movie”
A comment about the edition.	Panel_Description	“Choose the movie you will watch tonight!”

### ***EmptyContainer***

This panel has no content, except others panels.

Parameters :

<b>Description</b>	<b>Parameter Class</b>	<b>Example of value for the “hasValue” property</b>
Disposition of the panel.	Panel_Type	“HorizontalPanel”

### ***HTMLPanel***

This panel displays the HTML content of a distant file.

Parameters :

<b>Description</b>	<b>Parameter Class</b>	<b>Example of value for the “hasValue” property</b>
The address of the HTML file.	Html_File_Adress	“” <a href="http://labotalc.loria.fr/~meilendt/Contact/page/foot.html">http://labotalc.loria.fr/~meilendt/Contact/page/foot.html</a> “”

## ***SimilarInstancesPanel***

The panel displays a list of instances which has the same property that the edited one.

Parameters :

<b>Description</b>	<b>Parameter Class</b>	<b>Example of value for the “hasValue” property</b>
The class of the edited instance.	Instance_Class	“Movie”
The name of the edited instance.	Instance_Name	“my_individual”
The URL of the edited ontology.	Ontology_URL	“” <a href="http://labotalc.loria.fr/~kasimir/downloads/owl/cinema.owl">http://labotalc.loria.fr/~kasimir/downloads/owl/cinema.owl</a> “”
The namespace of the edited ontology.	Ontology_NameSpace	“” <a href="http://www.owl-ontologies.com/Ontology1224688205.owl#">http://www.owl-ontologies.com/Ontology1224688205.owl#</a> “”
The title of the panel.	Panel_Title	“Movie”

## ***CommentPanel***

It aims at listing the labels of a class of the edited ontology. This class is linked to the edited instance class by a predefined property predefined. The same mechanism is used in Kasimir (<http://labotalc.loria.fr/Kasimir/>) for detecting recommendation.

Parameters :

<b>Description</b>	<b>Parameter Class</b>	<b>Example of value for the “hasValue” property</b>
The name of the edited instance.	Instance_Name	“my_individual”
Class to where the labels are extracted.	Comments_Class	“Comment”
Property which links classes.	Property_Link	“isCommentOf”
The URL of the edited ontology.	Ontology_URL	“” <a href="http://labotalc.loria.fr/~kasimir/downloads/owl/cinema.owl">http://labotalc.loria.fr/~kasimir/downloads/owl/cinema.owl</a> “”
The namespace of the edited ontology.	Ontology_NameSpace	“” <a href="http://www.owl-ontologies.com/Ontology1224688205.owl#">http://www.owl-ontologies.com/Ontology1224688205.owl#</a> “”
The title of the panel.	Panel_Title	“Movie”

# Contributors

Fadi Badra

LORIA (UMR 7503 CNRS–INPL–INRIA-Nancy 2–UHP)

Vandoeuvre-lès-Nancy, France

[badra@loria.fr](mailto:badra@loria.fr)

Mathieu d’Aquin

Knowledge Media Institute

The Open University

United Kingdom

[m.daquin@open.ac.uk](mailto:m.daquin@open.ac.uk)

Jean Lieber

LORIA (UMR 7503 CNRS–INPL–INRIA-Nancy 2–UHP)

Vandoeuvre-lès-Nancy, France

[lieber@loria.fr](mailto:lieber@loria.fr)

Thomas Meilender

LORIA (UMR 7503 CNRS–INPL–INRIA-Nancy 2–UHP)

Vandoeuvre-lès-Nancy, France

[meilendt@loria.fr](mailto:meilendt@loria.fr)