

LUISA

Learning Content Management System Using Innovative Semantic Web Services Architecture

IST- FP6 – 027149



Deliverable 3.3

Annotation Tool

Fredrik Enoksson
Matthias Palmér
Ambjörn Naeve

Due date of deliverable: 28/02/2007

Actual submission date: 30/09/2007

Start date of the project: 01/03/2006

Duration: 30 Months

Fredrik Enoksson
Uppsala Learning Lab

Version 0.6, dated 05/09/2007

Change History

Version	Date	Status	Author (Partner)	Description
0.1	20070117	Initial draft	Fredrik Enoksson (ULL)	Table of contents and the first version of the chapters
0.2	20070131	Initial draft	Fredrik Enoksson (ULL), Matthias Palmér (ULL)	Finalizing the chapters and the whole document for the internal review.
0.3	20070301	Initial draft	Fredrik Enoksson (ULL), Matthias Palmér (ULL), Ambjörn Naeve (ULL)	Extending all chapters in order to improve readability and also taking into account the internal review.
0.5	20070302	draft	Fredrik Enoksson (ULL), Matthias Palmér (ULL)	Finalizing the draft.
0.6	20070905	Final	Matthias Palmér (ULL), Ambjörn Naeve (ULL)	Incorporating changes suggested by the commission review.





The deliverable has been modified with more focus on the LUISA use-case scenario, explaining the innovation of the annotation tool and its usefulness in the use-case over existing tools.



EXECUTIVE SUMMARY

This report introduces the Annotation Tool that has been developed in work package 3. The Annotation Tool is configurable by means of annotation profiles according to the principles introduced in deliverable 3.2 – the Annotation Profile Specification. As the scope of this report is on the tool itself, rather than how end user should use it, the main focus is on three different kinds of developer roles, *Code Library Developer*, *User Interface developer* and *User Interface Integrator*. For the *Code Library Developer* the design choices are described and the main ideas are presented in order to get an overview of the code. For this role more details can be found in the API documentation. The other two roles are discussed in length as a consequence of prioritizing flexibility and reuse of the annotation tool. The role *User Interface developer* is about how to tailor new user interfaces / skins for the annotation tool. In the next step, the *User Interface Integrator* role integrates a user interface /skin of the Annotation Tool into a bigger application in need of annotation support. The latter role is described in more detail as the required amount of work is smaller and better defined. For convenience, two default user interfaces are provided, one for a web-setting and one for a standalone application.

Document Information

IST Project Number	FP6 – 027149	Acronym	LUISA
Full title	Learning Content Management System Using Innovative Semantic Web Services Architecture		
Project URL	http://www.luisa-project.eu		
Document URL			
EU Project officer	Kypros Kyprianou		

Deliverable	Number	3.3	Title	Annotation Tool
Work package	Number	3	Title	Learning Object Annotation

Date of delivery	Contractual	28/02/2007	Actual	30/09/2007
Status	Version 0.6, dated 05/09/2007		final	<input type="checkbox"/>
Nature	Report <input type="checkbox"/>	Demonstrator <input type="checkbox"/>	Oth <input checked="" type="checkbox"/>	
Dissemination Level	Public <input type="checkbox"/>	Consortium <input type="checkbox"/>		
Abstract (for dissemination)	The main ideas and the structure of the Annotation Tool developed in WP3 is described in this document. The annotation tool is designed to be as generic and flexible and the document is aimed at describing the Annotation Tool for three different kinds of usage, developers of the core library, developers of user interface and user interface integrators.			
Keywords	Annotation Tool, RDF, Semantic Web, Graphical User Interface			

Authors (Partner)	Fredrik Enoksson (ULL), Matthias Palmér (ULL), Ambjörn Naeve (ULL)			
Responsible Author	Fredrik Enoksson		Email	fen@nada.kth.se
	Partner	ULL	Phone	+46 18 471 62 90

Project Consortium Information


Partner	Acronym	Contact
Atos Origin S.A.E. (Coordinator)	ATOS 	Nuria de Lama Atos Origin S.A.E. c/ Albasanz 12 E-28037 Madrid, Spain Email: nuria.delama@atosorigin.com Tel.: +34 91 214 9321 Fax: +34 91 754 3252
University of Alcalá de Henares	UAH 	Dr. Miguel-Angel Sicilia Information Research Unit University of Alcalá Ctra. De Barcelona, Km 33.6 E-28871Alcalá de Henares (Madrid), Spain Email: msicilia@uah.es Tel.: +34 91 886 6603 Fax: +34 91 885 6646
University of Uppsala	ULL 	Dr. Ambjorn Naeve University of Uppsala Kyrkogårdsgatan 2 C Uppsala Email: amb@nada.kth.se Fax: +46 184-716-294
Open University	OU 	Dr. John Domingue Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes, MK7 6AA, United Kingdom Email: j.b.domingue@open.ac.uk Tel.: +44 1908 655014 Fax: +44 1908-653-169
University Henri Poincaré	UHP 	Dr. Monique Grandbastien University Henri Poincaré Vandoeuvre les Nancy 54506, PO Box 239, France. Email: monique.grandbastien@loria.fr Fax: : +33 383-278-319
Giunti Interactive Labs S.r.l.	GIUNTI 	Dr. Fabrizio Giorgini Giunti Interactive Labs S.r.l. Abbazia dell'Annunziata Via Portobello Baia del Silenzio 16039 Sestri Levante (GE), Italy Tel.: +39.0185.42123 Fax: +39.0185.43347
EADS FRANCE – Innovation works	EADS 	Anne Monceaux EADS FRANCE – Innovation works Avenue Didier Daurat - Centreda 1, Toulouse, 31700, France. Email: anne.monceaux@airbus.com Tel.: +33 561-184-725 Fax: +33 561-187-611



TABLE OF CONTENTS

EXECUTIVE SUMMARY	3
1 INTRODUCTION	3
1.1 Abbreviations Used in this Document	3
2 DESIGN CHOICES OF THE ANNOTATION TOOL LIBRARY	3
2.1 Design Overview	3
3 USING THE ANNOTATION TOOL IN A WEB-SETTING	3
3.1 Eluisa Preparation and Deployment	3
3.2 Eluisa Configuration and Building	3
4 USING THE ANNOTATION TOOL IN AN APPLICATION	3
5 CHANGING THE GRAPHICAL USER INTERFACE	3
5.1 The Form Model	3
5.2 Guidelines for Presenting Multiplicity	3
6 CONCLUSION	3
APPENDIX A: SOME EXAMPLE CODE OF HOW TO USE THE ANNOTATION TOOL LIBRARY	3



1 INTRODUCTION

This is the first version of the documentation of the LUISA Annotation Tool (AT), developed in work package 3. The LUISA Annotation Tool is a specific user interface built on top of the Annotation Tool Library¹ which is constructed to work with Annotation Profiles that are specified in deliverable 3.2 – the Annotation Profile Specification.

An important aim of the Annotation Tool Library is that it can be reused by other applications that need to construct their own specific user interface for metadata editing. Applications, including the LUISA Annotation Tool, that are based on the Annotation Tool Library and hence upon Annotation Profiles, can easily change the set of metadata elements used, without having to redesign or further develop the application. Hence, the use of Annotation Profiles simplifies the development of more advanced editing scenarios as it provides editors as ready made building blocks.

Furthermore, the Annotation Tool is carefully designed to be easily integrated as a module into a surrounding application. This requires a technical adaption as well as a change in the appearance of the Annotation Tool to get a unified look and feel.

In LUISA, both the industrial and the academical use-case need to allow different users to edit different sets of metadata elements depending on their role in the editing process of a resource. The academical use-case requires librarian, teacher and administrator roles while the industrial use-case requires LO author, administrator, teacher, learner, training manager and team manager roles. Hence, providing specific support for all of these roles within the corresponding domain would lead to a total of 9 different annotation tools to be developed and integrated within corresponding authoring environment of each use-case. With the design of the LUISA Annotation Tool, it will be sufficient to specify 9 different annotation profiles and do the integration into each use-case authoring environment only once.

For a longer discussion of existing tools and the benefits of configurable annotation tools that are reusable, see deliverable 3.1 – State of the art.

The Annotation Tool fits into the proposed LUISA architecture according to figure 1 below

1A further development of the SHAME library (<http://kmr.nada.kth.se/shame>) created by KMR

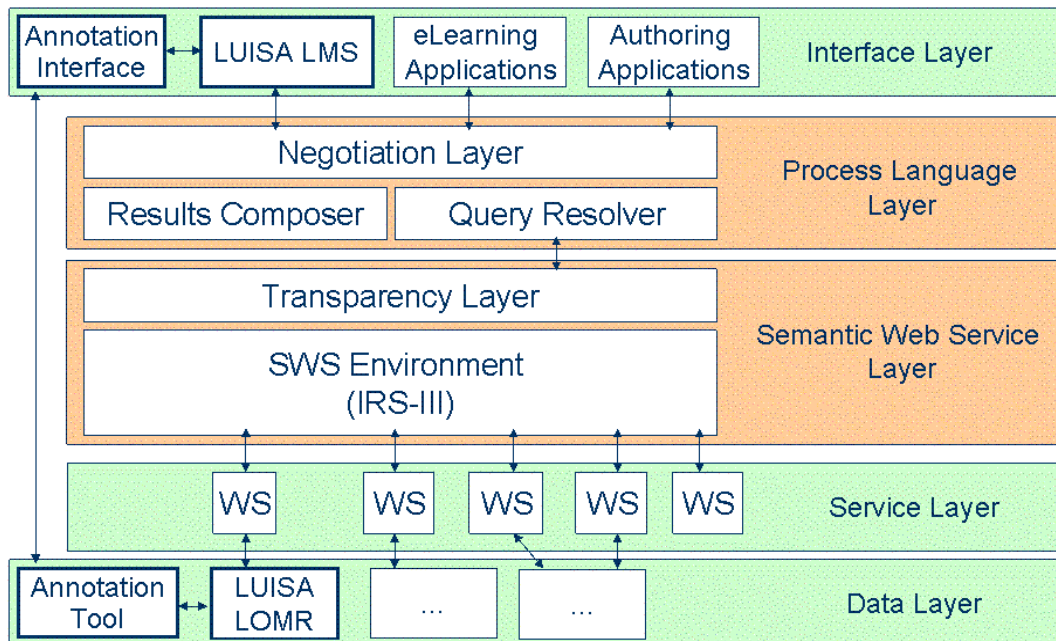


Figure 1: The proposed LUISA architecture with the Annotation Tool and its interface

The Annotation Interface (a specific annotation tool) can be built into other application like a Learning Management System (LMS) where it communicates with the LOMR (Learning Object Repository) for retrieval and storage of data.

The Annotation Process as introduced in deliverable D3.2, describes three new roles when using Annotation Profiles for editing. The *Annotation profile author*, the *Annotation profile facilitator* and the *end user*. Focusing on the development of the Annotation Tool and similar applications built upon the Annotation Tool Library, we see three additional developer roles.

1. **AT Library Developer** – Develops the Annotation Tool library used by the annotation tool
2. **AT User Interface Developer** – This role includes developing interfaces to the AT with the help of the AT library
3. **AT User Interface Integrator** – In this role an interface developed by the interface developer is put into another setting, for example an application or a web-setting

This document is split accordingly into three different parts with the aim of providing useful information for these three roles mentioned above. Section 2 describes the design choices made for the *library* (role 1). Section 3 and 4 outlines typical *integration* (role 3) into a web-setting and a rich application-setting correspondingly. Section 5 outlines how to *develop an interface* (role 2) with the help of the AT library.



1.1 Abbreviations Used in this Document

GUI – Graphical User Interface

AT – Annotation Tool

AP – Annotation Profile

API – Application Programming Interface

2 DESIGN CHOICES OF THE ANNOTATION TOOL LIBRARY

The AT is designed to make it possible for an arbitrary user to edit RDF-metadata in a form-based manner. It does so by using Annotation Profiles (AP), that are created in advance and act as input to the AT. The overall design goal of the AT is to be as generic as possible, that is why the following choices were made:

- **Flexible use of AP syntax** – An AP could be expressed in more than one way and this has been taken into consideration in the design. The draft version of the AT supports only one syntax that is different from the one specified in the Annotation Profile Specification, but due to the design a change to that specification of AP will be easy to implement in the last version of the AT.
- **Work with any RDF-API** – The easiest way to handle RDF would be to use an established / standardized generic API for java. Unfortunately, no such API exists at the moment, but two fairly stable Open Source implementations exist: *Jena*² that has been developed by HP, and *Sesame*³. The AT currently uses Jena, but has been designed with the intention to make an easy switch to any RDF-API.
- **Express syntax for RDF in several ways** – RDF can be serialized in several ways, where the most popular are N3⁴ and RDF/XML⁵. The AT can currently output RDF in both those syntaxes and the idea is to support any kind of serialization of RDF.
- **Well defined Form Model API** – Since one of the purposes of the AT is to be a pluggable component in many different applications, the API has been carefully defined. The API allows new GUI implementations as well as easy reuse of existing into new settings.

Furthermore the AT is implemented in Java, since it is one of the most widely used programming language and it is more or less independent of operating system. The code can be run on the Java Virtual Machine that is implemented for the most commonly used operating systems.

2.1 Design Overview

To make the annotation tool library useful in many contexts we have made it configurable in four ways. This is the same flexibility that the AT User Interface Integrators need to consider:

2 <http://jena.sourceforge.net/>

3 <http://www.openrdf.org/>

4 <http://www.w3.org/DesignIssues/Notation3>

5 <http://www.w3.org/TR/rdf-syntax-grammar/>

- The **source** of the metadata, where to load and save from. As well as which resources to edit metadata for.
- The **RDF library** used to manipulate metadata within the application.
- Which **Annotation Profile** to use when editing, may depend on the context, i.e. user roles, access rights, editing purpose.
- Which **GUI** (Graphical User Interface) to use.

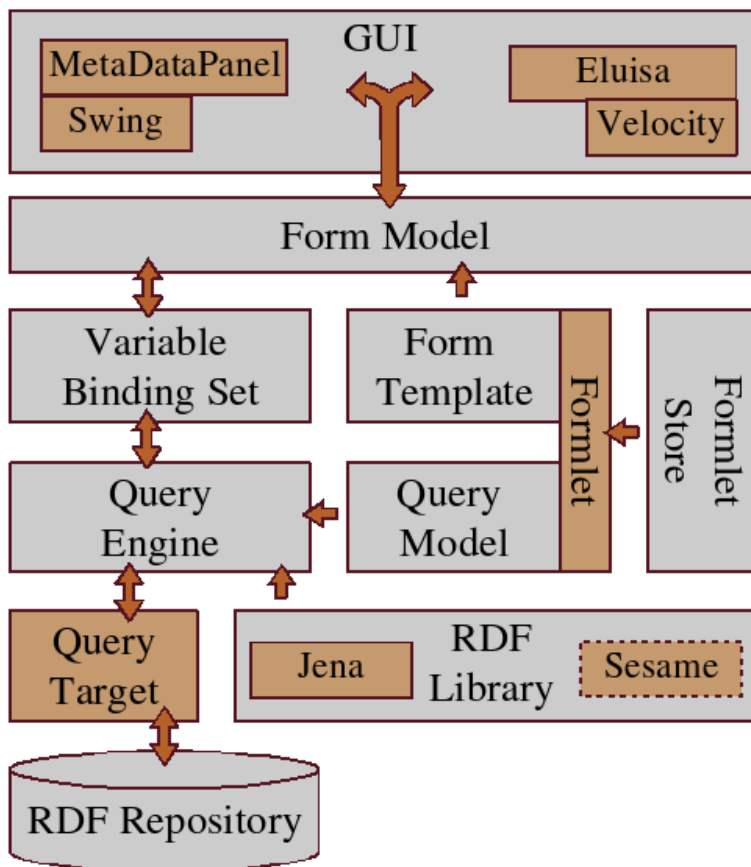


Figure 2: An overview of the main parts of the Annotation Tool Library, the parts colored in orange (Metadatapanel, Swing, Speed, Velocity, Formlet, QueryTarget, Jena and Sesame) are configurable.

Hence, when using the annotation tool library these are the main aspects to consider for a successful adaption. An overview of the design of the annotation tool library is given in figure 2, the source of the metadata is the QueryTarget, the alternatives for RDF library are *Jena* and *Sesame*⁶, the Annotation Profile is represented by the *Formlet* class⁷, and the GUI choices are here represented by the *MetadataPanel* and the LUISA annotation tool called *eluisa*.

6 The Sesame integration remains to be completed in the Annotation Tool Library.

7 The formlet class may be renamed to AnnotationProfile in the future.



In addition, the FormModel provides an API which is of interest for the AT user interface developers. The FormModel provides access to the abstract model of the form that should be mirrored in the GUI.

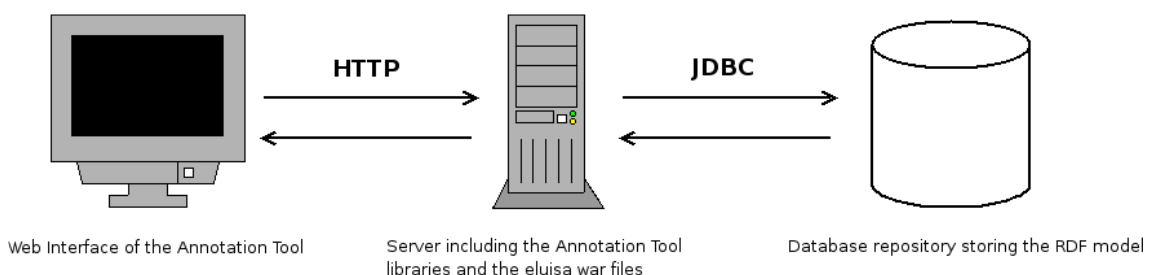
The remaining grey boxes (not yet mentioned) in figure 2, i.e. Template Form, Query Model, Query Engine, and Variable Binding Set, all represent the inner workings of the Annotation Profile and are only included for the sake of completeness.

3 USING THE ANNOTATION TOOL IN A WEB-SETTING

When using the AT in a web-setting you are most likely interested in integrating it into an existing application. However, for simplicity we will consider a simplified application, the eluisa application, that provides a minimal GUI for choosing a resource and then editing/presenting it according to a set of predefined APs. Hence, for eluisa, we will need the following:

- a **servlet container** that supports Web Application Archive (WAR).
- A relational **database** like MySQL, PostgreSQL or Oracle.
- The **eluisa war file** to upload to the servlet container.

The architecture of this is quite straightforward with a server with the war file that is connected to a relational database via JDBC. The client requests runs the sever on a browser, all according to figure 3



Chapter 3.1 will explain how to deploy the eluisa war file, but if the default configurations needs to be changed the steps in chapter 3.2 should be carried out first.

3.1 Eluisa Preparation and Deployment

The following steps should be followed with the eluisa.war file and run the AT in a web-setting:

1. Install the servlet container that the eluisa.war-file should be uploaded to.
2. Install the database and make sure it is accessible from the machine where of the servlet container is running.
3. Create a database and make sure you have access to it via a suitable user.
4. Finally, when done with the configuration and building steps outlined below, upload the eluisa.war file to the server (check your specific server implementation how to do that) and your application should be up and running on the address {your domain}/eluisa/main (unless you specified



otherwise in the web.xml).

When you go to that address then you will first be prompted with a drop down-menu to choose what resource you want to edit. This list will be empty if the database does not contain any suitable resources. By default the eluisa tool looks for all resources that are of the type learning object⁸.

3.2 Eluisa Configuration and Building

The configuration that can be done in this draft-version is restricted to settings of the database to store the RDF model and the password to it. The file is called eluisa.properties and can be found in the \$ELUISA-DIR/include/vm/WEB-INF/ folder.

In the configuration file it is necessary to specify how to connect to the database:

- A jdbc URL (containing which machine, user, and password to use).
- A name for the RDF model in the database to load and save from. If you have preloaded RDF, make sure to use the same name.

Don't forget to include appropriate database driver for your database in the library as well as specify this in the config file.

The file eluisa.war keeps all the APs inside a file called eluisa-formlets.jar, this file needs to be replaced with a file that contains your APs. To create a new such file you will need to construct the underlying folder structure that contains only two folders, one named **formlets** and the other one **ontologies**. Put all the APs inside the formlets folder and inside the folder ontologies you will put all supporting ontologies. The APs and ontologies will be discovered automatically by the Annotation Tool. Create the file eluisa-formlets.jar with the folder structure and put it into the dist folder.

When finished configuring and placed the files needed in the correct folders, just run ant. The default target will produce the eluisa.war in the dist folder.

8 Currently the type is identified with the URI <http://www.cc.uah.es/ie/ontologies/LUISA#learningObject>, the default base will however change in the future to something that identifies the project better

4 USING THE ANNOTATION TOOL IN AN APPLICATION

If the web setting that has been provided is not suitable it is also possible to use the libraries of the AT to build it into your own application. This chapter will give an overview of the necessary steps. An example is also given in Appendix ???. A prerequisite for easy integration of the annotation tool into an application is that it is a java based program and that it can be integrated with Swing-components. If not, you probably need to build your own GUI, see next chapter for some hints of how to do this.

Looking into the source code and its structure you will find a class called `MetadataPanel` in the `se.kth.nada.kmr.shame.applications.util` package. This class will make it easy for you to create a form-based editor for RDF. The input needed is:

- A container for the metadata E.g. a file or an in memory representation. Implement the `EditContainer` interface.
- the AP to use identified via a URI, make sure the AP is loaded in advance via the `FormletStore` singleton.
- The URI of the resource to edit.
- If the AP specifies ontologies they will be loaded automatically.

Inside the code in the package `se.kth.nada.kmr.shame.examples` there is a class called `MetadataPanelExample` that can be used as an example on how to use the `MetadataPanel`. The first thing the example does is to use a class that detects the APs that are stored in the folder from where the AT is run. This way a certain AP can be loaded once the instance of the `MetadataPanel` has been instantiated, which is done in the code. The last thing that the `MetadataPanel` needs is a place to store the edited RDF, so an instance of a container is initiated with the file to edit and the resource in that file to edit. When the method `edit` is called the `MetadataPanel` is ready to be displayed, it is a subclass of the Swing-class `JPanel` and can therefore easily be put into any interface created by the Swing-library. The interesting parts of the `MetadataPanelExample` is included as an example in Appendix A.

5 CHANGING THE GRAPHICAL USER INTERFACE

The `MetadataPanel` described in the previous chapter provides a fixed GUI according to each AP. In order to create your own GUI you need to make use of the underlying annotation tool library. In the following, familiarity with Annotation Profiles is required for correct understanding, Deliverable 3.2 provides the necessary background. How the important classes of the annotation tool library are related is shown in figure 4 and they are further described in the rest of this chapter

For the terminology introduced in deliverable 3.2, Graph Pattern, Variable Binding, Form Template, Form Model, Form Item, etc. there are corresponding classes. The class `QueryModel` introduced in figure 2 is a superclass of the `GraphPattern` class introduced here. The `Value` class is an abstraction so that there is no direct dependencies to a specific RDF library.

5.1 The Form Model

When implementing a new user interface, it is the `FormModel` interface and its constituents that will be utilized. The `FormModel` interface consists of the `FormItem` interface ordered into a tree of the `GroupFormItem` interface. The two other subinterfaces `TextFormItem` and `ChoiceFormItem` provide the input facilities for text input and selection of predefined values respectively. The `Value` interface provides primitives for manipulation of the data and the `Choice` interface contains available predefined values.

5.2 Guidelines for Presenting Multiplicity

From the Form Template the application knows how many text-fields or choice items that are allowed to be filled and therefore also rendered. Indications of what is possible to add or remove of the fields or items should be included in the interface and preferably with a '+' or '-' signs. The placement of these is of importance, since it tells the user how and where more values are added. For example when adding keywords to a resource, several keywords can be added in several languages. The signs where the keywords are expressed in more languages should be placed close to the languages-choice and to add another keyword the signs should be close to the label.

6 CONCLUSION

This report has focused on the three roles, AT Library Developer, AT User Interface Developer and AT User Interface Integrator. These three roles comes from the different needs on how to use the annotation tool library. A guide to the internals of the code and also how to construct your own Graphical user Interface or integrate it into another application has been provided. The aim has been to give an overview of the main parts of the Annotation Tool and the underlying library.

A few things remains to be worked out in more detail, for example, how to connects to the LOMR, as being developed in WP4, for saving and loading metadata as well as how to integrate with the future LUISA LMS, provided by WP5. However, we foresee that these issues will have simple solutions, in part due to the careful design of the underlying Annotation Tool Library as well as the Annotation Profile Specification.

APPENDIX A: SOME EXAMPLE CODE OF HOW TO USE THE ANNOTATION TOOL LIBRARY

The following code is an example on how to use the Metadatapanel with some comments given:

```
//This will automatically load all the Formlets that you
//have in the folder "Annotation_Tool"/resources/formlets
AutoDetectFormletConfigurationSets adfcs =
    new AutoDetectFormletConfigurationSets
        ("resources/formlets", "formlets.rdf");

//Initiating the metadatapanel, the argument is the name of
//the window that will appear
MetadataPanel mpanel = new MetaDataPanel("MetadataPanel");

//Sets the formlet we will use for this MetaDataPanel, is
//identified with a URI
mpanel.setFormletConfigurationId("http://kmr.nada.kth.se/sh
ame/Formulator/formlet#DescriptionField");

//The model and the resource to edit created with the help
//Jena RDF API. The model could also be taken directly from
//a database
ModelMem mem = new ModelMem();
String fileToEdit = "file://fileToEdit.rdf";
mem.read(fileToEdit);
Resource resource =

mem.createResource("http://www.luisa.org/basicOntology#Inst
anceLuisa");

//The following is only interesting if you want to save RDF
//into a file
URI fileToEditURI = null;
try{
    fileToEditURI = new URI(fileToEdit);
} catch (URISyntaxException urise){
    //Do nothing, since we manage to have the URI
    //as null for this example
    System.out.println("Did not manage to create"+
        "a URI for file "+fileToEdit);
}

//This is the container that we use for saving to file. You
```



```
//will have to implement EditContainer in order to store
//the model into a database
container = new Container(mem, fileToEditURI);

//Tells the MetaDataPanel that we want to edit the resource
//in a certain container
mpanel.edit(container, resource);

//The following is just GUI-code
JFrame.setDefaultLookAndFeelDecorated(true);
jf = new JFrame();
jf.setLayout(new java.awt.BorderLayout());
jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JButton jb = new JButton("Save & Exit");
jb.setVisible(true);
jb.addActionListener(new SaveAndExitButtonListener());
jf.getContentPane().add(jb, BorderLayout.SOUTH);
jf.getContentPane().add(mpanel);
jf.pack();
jf.setVisible(true);
mpanel.setVisible(true);
```