SeamWebAppGen

Installation Guide & Tutorial

Version 0.2

Table of Contents

1. Change History

Date	Document Version	Changes
19 March 2008	**0.1**	First edition
05 May 2008	**0.2**	Update Linux installation procedure Added Windows installation procedure Postgresql installtion paragraph move from Tutorial_1 to Installation chapter Updated Tutorial 1 to Windows platform

2. Preface

This document describes the key functions of the SeamWebAppGen tool and provides guidance to programmers for developing Web applications with minimum effort and time.

3. Introduction

3.1 Why SeamWebAppGen ?

The majority of the Web applications are based on lists of business objects that need to be created, edited, deleted, presented, searched, printed, etc. , but also related to other objects. These functions applied to business objects, in most instances, represent the prevailing part of the functional content of Web applications. This is the area in which SeamWebAppGen can help by producing code automatically, in response to specifications provided by programmers, and by providing hooks to insert custom code developed on purpose, to satisfy specific functional requirements.

SeamWebAppGen is, then, a web applications generator tool. As already mentioned, it enables quick and easy development (generation) of Web based applications requiring management of lists of objects, on which users need to perform frequently and intensively a variable number of predefined operations, such as: creation, editing, deletion, searching, presentation, printing, managing of relationships, etc. In doing this SeamWebAppGen relieves the programmer from developing on his own functions to be applied to business objects in his applications which are, rather, generated automatically, just by providing minimum specifications.

In more details, the objectives of this tool are:

1. Accelerate time-to-market and minimize the effort required for the development and testing of Web based applications;

2. Provide hooks for easy insertion in the Web applications of custom functional code, developed on purpose for each application;

3. Attain maximum transparency of the underlying technologies to developers (and to end users);

The following very basic example should help understanding how this tool could help in your daily work as a developer.

Let’s assume that in your business area you have to manage a pool of teachers and allocate them to a set of classes. Roughly speaking you can imagine your application as based on a list of teachers and a list of classes. In relation to that, the application to be developed will have to provide functions for creating, editing e deleting teachers (objects from one of the two lists) and for creating, editing and deleting classes (objects from the other list). Moreover, this application will have to provide means to relate objects from the first list (teachers) with

objects from the second list (classes).

Using SeamWebAppGen you can develop (generate) your basic application inclusive of the described functionalities in few minutes, without writing a single line of code and with no need to deal with the complexity of the underlying technologies.

SeamWebAppGen uses the EJB3 technology for the creation of an EJB3 Entity Bean for each Business Object you need to consider in your application (referring back to the example: to each teacher and to each class in your two lists).

JSE annotations are employed to define the user interface and/or object behavior. After completion of the generation process your new Web application is ready for deployment, unless specific unique functions, not provided by the generated application, need to be applied to some or all your objects (see below).

Sounds very simple, isn't it? In practice the development work required to deliver new Web applications based on management of business objects using SeamWebAppGen is really simple, implies minimization of repetitive tasks and minimum propensity to errors. However, a simple work does not mean that implies trivial tasks. As you move on in this document you will realize that development work of Web applications based on business objects without SeamWebAppGen is much more complex, time consuming and error prone.

In terms of underlying technologies, applications generated by SeamWebAppGen makes use of the following:

1. JSF 2.1;
2. JBoss SEAM;
3. JBoss RichFaces;
4. EJB 3.0;

and can be deployed on the JBoss Application Server (4.x).

However, since all adopted technologies run on other J2EE application servers brands, the SeamWebAppGen generated web applications should also run in these environments, although not yet tested. (Due to lack of time availability, we have not been able to do it yet)

As already mentioned above, although SeamWebAppGen enables development of web applications that should satisfy a wide variety of business objects processing requirements, there might be situations requiring specific functions that cannot be provided by the SeamWebAppGen generated code. These situations will have to be supported through insertion of, on purpose developed, custom code in specific points within the generated code. Your custom code will not be affected and will be preserved unchanged by the SeamWebAppGen generation process.

At this point you might want to start immediately to deal with SeamWebAppGen without learning more about its functional characteristics on paper. If this is the case, please, jump to Chapter 5 (Installation), then to subsequent Chapters (Tutorials).

3.2 Anatomy of a generated application - end user functions

As mentioned in the previous Chapter, SeamWebAppGen generated web applications are based on business object lists. For example a music CD management application is based on a list of CD's, as shown in Figure 1 *Bean list*

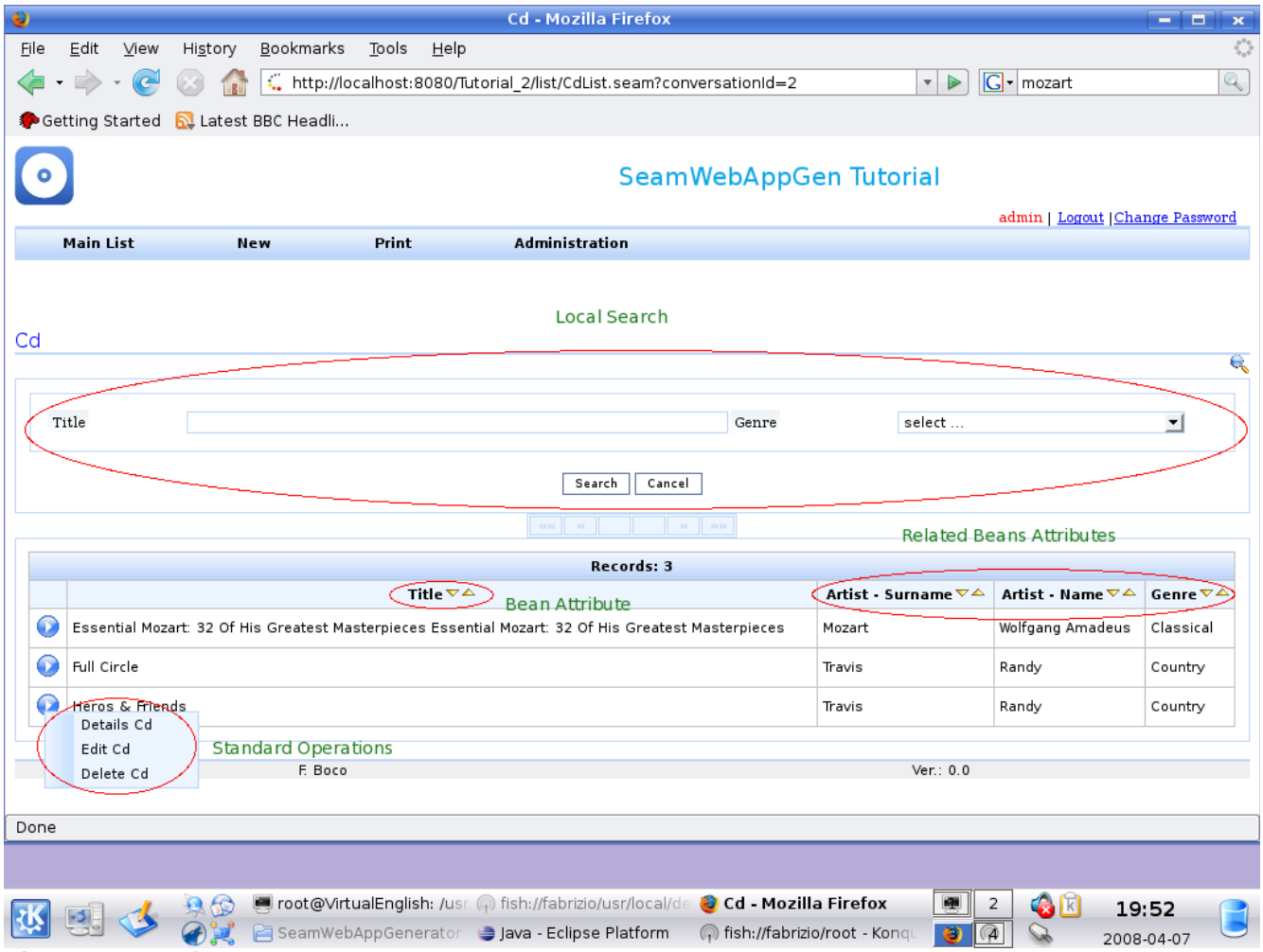


Figure 1: Bean List

These lists, on the basis of the options selected at application development time, are equipped with various functions and attributes. These function and attributes are assigned to each list and activated at generation time, with no further involvement of the developer (who is not required to write any code). At this point functions and attributes selected for each list are available to the web application end user.

In a list you, as a developer, can decide to show both attributes belonging to the addressed business objects and attributes belonging to related business objects. Annotating business objects attributes you can decide which of them will be shown in the list.

Each attribute shown in a list can be ordered in ascending or descending order.

For each business object in a list, SeamWebAppGen prepares a set of default functions available to the end user, as follows:

- Details – To quickly see all the information related to the Bean instance;
- Edit – To edit the information related to the Bean instance;
- Delete - To delete the Bean instance.

In addition, a set of standard functions can be assigned to each one of the lists of business objects of your web application as follows:

- Main List – To quickly go to main application list;
- New – To create a new business object;
- Print – To print a basic PDF list of all business objects included in the list. Obviously, many business applications need sophisticated reporting. If this is the case, you need to write custom code by yourself and put it in the application on purpose placeholder (hook) to apply it.
- Administrator – To access to the user administration and application configuration functions.

Management of each Business Object (mainly create, edit and delete) is achieved through a form, which includes all the attributes and relationships to be entered by the end user (and is referred to as “Bean Form”).

Attributes and relationships may be required or not and may have an help window or not associated to each one of them. If they are expressed in numeric form, can be formatted as you like, if they are expressed as dates, they have a calendar.

For each attribute a standard validation is provided. However, you, as a programmer, can add additional validations on your own.

Rules may be associated to each attribute to enable hiding when some functions are applied (i.e. creation, editing, etc).

The Bean Form has a menu option to PDF print all the bean attributes.

So far we have shown how SeamWebAppGen can provide you, as a developer, with the capability to include functions and attributes related to single business object in a list or to entire lists. This is achieved without writing a single line of code (see later Tutorial 1, Chapter 6). However, as already mentioned in the previous Chapter, if you need to apply specific additional functions to some or all your business objects, your own on purpose developed

custom code must be provided and inserted in specific points within the SeamWebAppGen generated code.

3.3 Anatomy of a generated application – back end

As already mentioned, each one of your business objects is described by an EJB3 Entity Bean.

The EJB3 Entity Bean can be annotated with a variety of SeamWebAppGen annotations to control its behavior and the end user interface and the select the functions made available to him.

Through the annotations SeamWebAppGen creates an EJB3 Conversation Scoped Stateful Session Bean which acts as a manager of the functions applied to the Entity Beans (Business Objects) of your Web application, as follows:

- Entity Bean CRUD operations;
- Entity Bean list creation;
- Entity Bean search;
- Entity Bean pop up management;
- etc.

The EJB3 Conversation Scoped Stateful Session (which is referred to as Entity Bean Manager) is equipped with various placeholders (application hooks) available to you, as programmer, to insert custom code to extend or modify its behavior. This is one of the "hooks" provided to enable integration of the basic SeamWebAppGen functions as mentioned in the Chapter 3.2.

3.4 Benefits of using SeamWebAppGen

This is a brief list of the benefits, you as a programmer, will attain using SeamWebAppGen:

- Drastically compress time-to-market of the basic functions of Web applications (it can be as little as few hours!);
- Basic J2EE Seam and JSF project skeleton ready for compilation in few minutes;
- No code writing for functions that can be directly generated by this tool, that we might refer to as "standard" (no code developed, no bugs injected);
- A variety of "standard" functions can be applied to each business object, within object lists of Web applications, under your control, as a programmer. The list of the available

standard functions is as follows:

- Lists with related selection conditions;
- CRUD operations;
- PDF List print;
- PDF Bean print;
- Global Search (search into all beans);
- Local Search (search into beans belonging to a specific list);
- Field formatting;
- Easy file upload / attachment;
- etc;

- Easy customization of Web applications behavior and related end user interfaces through extensive availability of annotations;
- Availability of an Eclipse Plug in to help you, as a programmer, to insert and update annotations;
- Simplified inclusion of specific additional functions, to be applied to some or all of your business objects, through your own on purpose developed custom code, inserted in placeholders. Your custom code will not be affected and will be preserved unchanged by the SeamWebAppGen generation process.

3.5 Brief history of SeamWebAppGen

First version of this tool (2004) was based on EJB2.1 and XML files to describe graphical interface and objects behavior.

It was used for very small projects and the management of XML files was very difficult and error prone.

Few months later, the use of JSE 1.5 annotations reduced the complexity and enabled the extension of the number of options provided to the programmer to control the tool behavior and the end user interface generation. The use of JSE 1.5 extended to 30 the number of different main options available to the programmer for controlling the application generation.

Immediately after that a major migration to the Struts infrastructure and to EJB3 was done, and an Eclipse plug-in was developed.

The Eclipse plug-in has the objective to guide the developer in the insertion of the correct

annotations and of the related options. The Eclipse plug-in supports most of the SeamWebAppGen annotations and related options. However, for priority reasons, the Eclipse plug-in could be left one o more update steps behind with respect to the functional changes applied to SeamWebAppGen.

The Struts based generator was used to develop a couple of Web applications that are still running and are still maintained. The larger application supports 50 objects, complex relationships and a variety of business processes.

As the Seam technology emerged the decision to move to it was taken and, during the first quarter of 2007, the migration to JSF, Trinidad and Seam was performed.

The Seam based generator was used to develop a medium size Web application (about 20 business objects) extensively customized. The JSF / Seam infrastructure was found easier and faster than the Struts. In other terms the migration to Seam / JSF / Trinidad was recognized as an advantage.

In my opinion, however, the integration between Trinidad and Seam was not very stable since a number of problems were still outstanding in the then latest versions of the two products.

A couple of months ago a new migration was done (hopefully the last!) to replace Trinidad with JBoss RichFaces that has a better integration with Seam and some interesting ajax features.

As of the end of February 2008, I decided to share the SeamWebAppGen as Open Source product and since then I have been involved into writing and translating some documentation and messages from Italian to English.

3.6 3rd party software used SeamWebAppGen

This is the list of all 3rd party software used by SeamWebAppGen:

- Apache commons;
- Apache FileUtils;
- JBoss Seam;
- JBoss Rich Faces;
- TynyMCE (Rich Text field formatting);
- Free Icons by Axialis Software;
- Any more ?

4. Disclaimer

SeamWebAppGen is provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE.

You are solely responsible for determining the appropriateness of using SeamWebAppGen and whatever is generated through it.

You solely assume any risks associated with the use of SeamWebAppGen and whatever is generated through it.

5. Installation

5.1 Linux

The following installation instructions assume that you are running either Ubuntu or a Debian based Linux and that most of the required applications can be installed via your preferred package manager.

If you are running other Linux distributions you should be able to adapt the installation instructions provide here to your preferred distribution.

5.1.1 Java installation

To install the java jdk 6.0 you have to enter the following command:

```
sudo apt-get install sun-java6-jdk
```

5.1.2 Eclipse installation

First of all download the last Eclipse release from its site.

You can download it manually following the instructions on the site:

```
http://www.eclipse.org/downloads/
```

Choose Eclipse IDE for Java EE Developers, follow instructions and save the file in a temporary directory (we assume ~/temp), then install it using the following commands:

```
cd ~/temp
tar -xvzf eclipse-xxxx-gtk.tar.gz  /usr/local
```

After that, download last stable version of the jBoss Tools for Linux (JBossTools-xxxx-ALL-linux-gtk.zip) from:

```
http://sourceforge.net/project/showfiles.php?group_id=22866&package_id=242269&release_id=583838
```

save them in your temporary directory and then install them:

```
cd ~/temp
unzip -x -o JBossTools-xxxx-ALL-linux-gtk.zip -d /usr/local/
```

5.1.3 JBoss AS Installation

Download the JBoss Application Server last available version from:

```
http://labs.jboss.com/jbossas/downloads
```

Choose the last version of family 4.2.x and click Download.

Install it with the commands:

```
cd ~/temp
unzip -x -o jboss-xxx.zip -d /usr/local/
```

To be independent from future releases, you have to create the following link:

```
ln -s /usr/local/jboss-xxx /usr/local/jboss-current
```

From now on, when a new jBoss Application Server release is available you can update your development environment by just installing it and changing the link.

5.1.4 JBoss Seam Installation

Download last available version of the Jboss Seam from:

```
http://seamframework.org/Download
```

and then install it:

```
cd ~/temp
unzip -x -o jboss-seam-xxx.zip -d /usr/local/
```

To be independent from future releases, you have to create the following link:

```
ln -s /usr/local/jboss-seam-xxx /usr/local/jboss-seam-current
```

5.1.5 JBoss Rich Faces Installation

Download last available version of the JBoss Rich Faces from:

```
http://labs.jboss.com/jbossrichfaces/downloads/
```

and then install it:

```
cd ~/temp
unzip -x -o richfaces-ui-xxx-bin.zip -d /usr/local/lib/
```

To be independent from future releases, you have to create the following link:

```
ln -s /usr/local/lib/richfaces-ui-xxx-bin   /usr/local/lib/richfaces-current
```

5.1.6 SeamWebAppGen Installation

Download last available version of SeamWebAppGen from:

```
http://sourceforge.net/project/showfiles.php?group_id=225359&package_id=272601
```

to the temp directory (~/temp).

Open terminal and enter the following commands

```
cd ~/temp
java -jar install.jar
```

Leave all default options unchanged and click on next until the end of installation process.

That's it.

5.1.7 Install and Configure PostgreSQL DBMS

The assumption made is that you are using PostgreSQL as DBMS, and that it is not installed.

If you have a different database accessible via the JDBC driver, you have to adapt the procedure accordingly.

First of all, you have to install PostgreQSL using the following command:

```
apt-get install postgresql
```

Open the file

```
/etc/postgresql/8.2/main/postgresql.conf
```

and replace:

```
#listen_addresses = 'localhost'
```

with:

```
#listen_addresses = 'localhost'
```

In this way your database will be accessible from the network.

Now you have to set the superuser password:

```
su – postgres
psql
alter user postgres with encrypted password 'postgres';
```

Last step in DBMS preparation is installing the JDBC driver into the JBoss application server.

Download last version from:

```
http://jdbc.postgresql.org/download.html
```

save in the temp directory and then install it:

```
cp ~/temp/postgresql-8.x-yyy.jdbc4.jar /usr/local/jboss-current/server/default/lib/postgresql-8.x-yyy.jdbc4.jar
```

Security Note: This PostgreSQL configuration is very basic and doesn't provide enough security in a production environment. Check PostgreSQL documentation for more information.

5.2 Windows

5.2.1 Java Installation

Downloaded last jdk stable 1.6.x version for Windows from:

http://java.sun.com/javase/dowloads/index.jsp

and save it in a temporary directory (we assume C:\temp).

Open the temp directory and run:

```
jdk-xxx-windows-xxx.exe
```

Install the JDK with the proposed default options (click next each time is required).

Now, the JAVA_HOME have to be configured for JBoss Application Server.

Select:

```
Start -> Settings -> Control Panel
```

Click on System icon and then:

```
Advanced (tab) -> Environment Variables
```

Click on New button into the System Variables area and enter:

```
Variable name: JAVA_HOME
Variable value: C:\Program Files\jdk1.6.x
```

then click OK.

5.2.2 Eclipse Installation

First of all download the last Eclipse release from its site.

You can download it manually following the instructions on the site:

```
http://www.eclipse.org/downloads/
```

Choose Eclipse IDE for Java EE Developers, follow instructions and save the file in a temporary directory (we assume C:\temp), then unzipp it into the default application directory (usually C:\Program Files)

After that, download last stable version of the jBoss Tools for Windows (JBossTools-xxx-ALL-win32.zip) from:

```
http://sourceforge.net/project/showfiles.php?group_id=22866&package_id=242269
```

save them in your temporary directory and then install unzip the file into:

```
C:\Program Files
```

5.2.3 JBoss AS Installation

Download the JBoss Application Server last available version from:

```
http://labs.jboss.com/jbossas/downloads
```

Choose the last version of family 4.2.x and click Download.

Install it unzipping the downloaded file to:

```
c:\usr\local
```

Rename the JBoss AS installation directory into:

```
c:\usr\local\jboss-current
```

5.2.4 JBoss Seam Installation

Download last available version of the Jboss Seam from:

```
http://seamframework.org/Download
```

Install it unzipping the downloaded file to:

```
c:\usr\local
```

Rename JBoss Seam installation directory into:

```
c:\usr\local\jboss-seam-current
```

to the JBoss Seam installation directory.

5.2.5 JBoss Rich Faces Installation

Download last available version of the JBoss Rich Faces from:

```
http://labs.jboss.com/jbossrichfaces/downloads/
```

Install it unzipping the downloaded file to:

```
c:\usr\local\lib
```

Rename JBoss Rich Faces installation directory into:

```
c:\usr\local\jboss-richfaces-current
```

5.2.6 SeamWebAppGen Installation

Download last available version of SeamWebAppGen from:

http://sourceforge.net/project/showfiles.php?group_id=225359&package_id=272601

to the temp directory (C:\temp).

Open a command line window and enter the following commands

```
c:
cd \temp
java -jar install.jar
```

Leave all default options unchanged and click on next until the end of installation process.

That's it.

5.2.7 Install and Configure PostgreSQL DBMS

The assumption made is that you are using PostgreSQL as DBMS, and that it is not installed.

If you have a different database accessible via the JDBC driver, you have to adapt the procedure accordingly.

Download the last 8.2.x version from:

http://www.postgresql.org

and save the downloaded file to the temp directory.

Unzip the file:

postgresql-8.2.x.zip

then run:

postgresql-8-2.msi

to start the installation procedure. Let all the default options unaltered and click on Next when required.

When superuser password is asked, insert:

postgres

and click Next.

Copy the file:

C:\Program Files\PostgreSQL\8.x\jdbc\postgresql-x.jdbc3.jar

to the directory:

C:\usr\local\jboss-current\default\deploy\lib\

Security Note: This PostgreSQL configuration is very basic and doesn't provide enough

security in a production environment. Check PostgreSQL documentation for more information.

6. Tutorial 1 – Your first application

In this tutorial you will test your development environment and develop your first very basic application by means of the SeamWebAppGen tool.

6.1 Step 1 – Create the Eclipse Java Project

Open Eclipse entering the following command into a shell window:

```
/usr/local/eclipse/eclipse                    (Linux)
c:\program files\eclipse\eclipse              (Windows)
```

create a Java project:

```
File -> New -> Java project
```

Write:

```
Tutorial_1
```

and check that the option 'Create separate folders for sources and class files' is selected. Then click:

```
Finish
```

From now on we will assume that your project has been created in the Eclipse workspace and that the workspace is located into the following directory:

```
/usr/local/development                        (Linux)
c:\usr\local\development                      (Windows)
```

If your environment is different, you have to modify the tutorial accordingly.

6.2 Step 2 – Run SeamWebAppGen

Type the following commands to start the generator:

```
cd /usr/local/SeamWebAppGen                   (Linux)
cd c:\usr\local\SeamWebAppGen                 (Windows)
java -jar SeamWebAppGen.jar
```

You can start the generator from the interface too, by example on Windows:

```
Start -> Programs -> SeamWebAppGen -> SeamWebAppGen
```

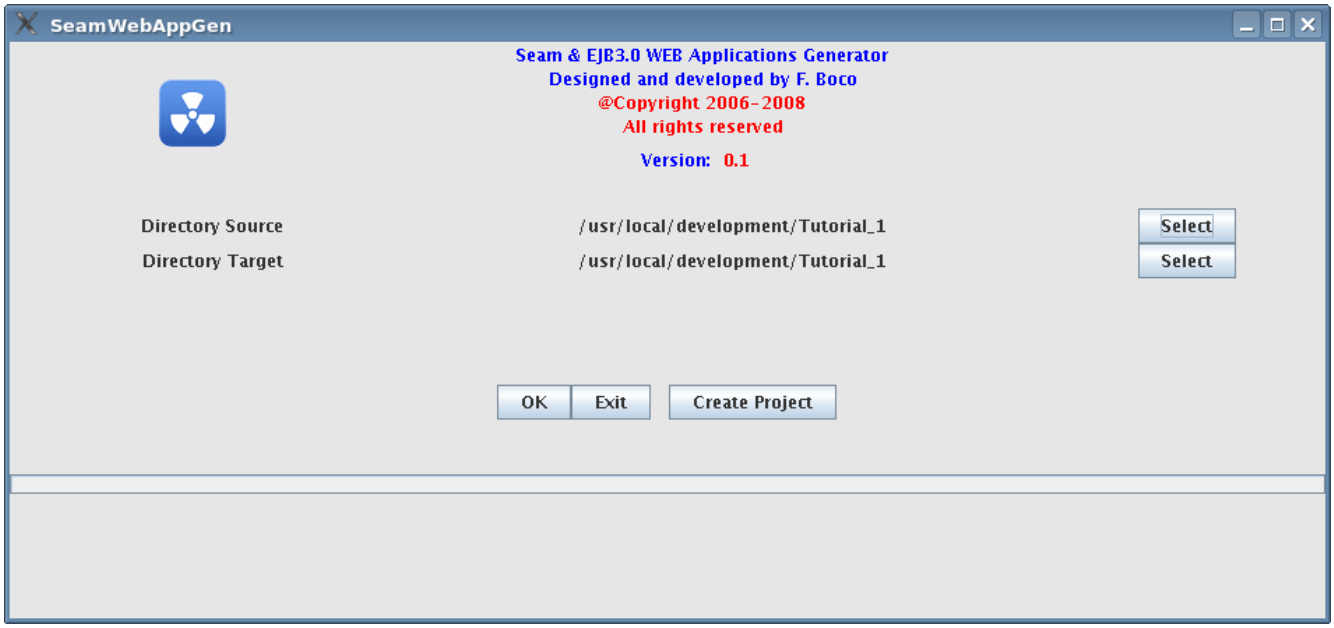


Figure 2: SeamWebAppGen Main Window

After that you should see the SeamWebAppGen main window (see Figure 2).

Click Select to select the source directory:

/usr/local/development/Tutorial_1	(Linux platform)
c:\usr\local\development\Tutorial_1	(Windows platform)

then click Create Project, the window will change into what is shows in Figure 3.

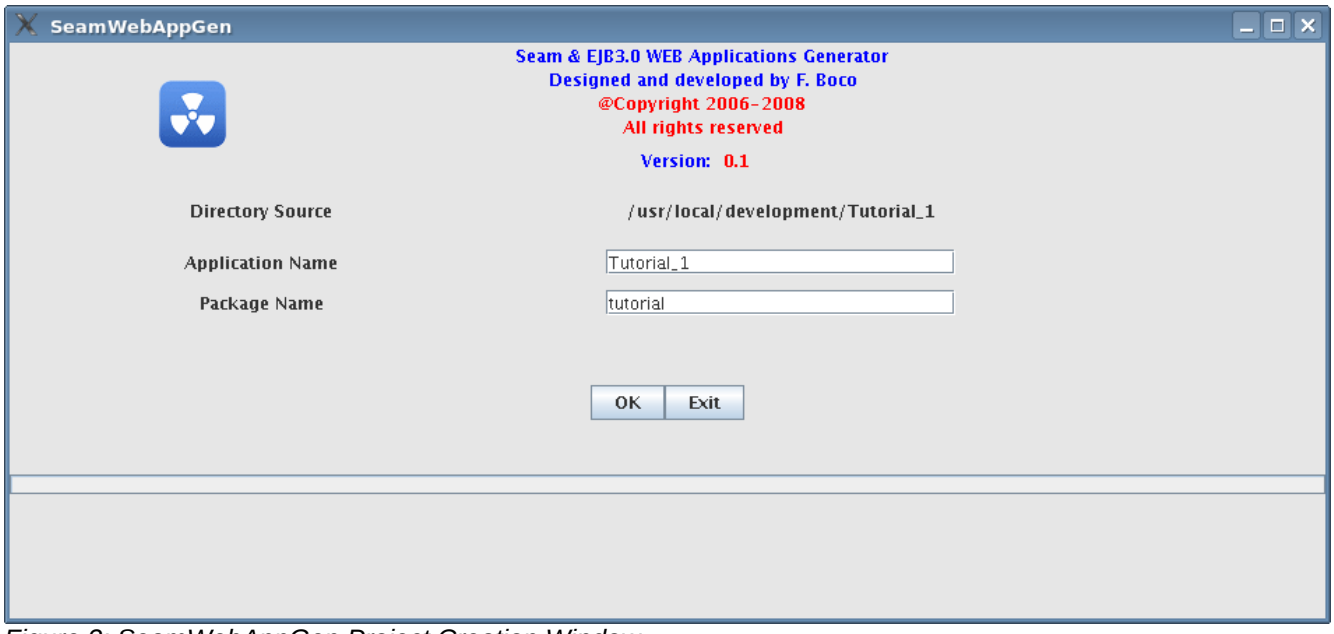


Figure 3: SeamWebAppGen Project Creation Window

Enter:

- Application Name: Tutorial_1 – This will be part of the URL used to access the application
- Package Name: tutorial

then click OK.

Your project will be populated with everything you need to create a full application.

6.3 Step 3 – Create your first bean

First of all, select the Tutorial_1 project and then press F5 to refresh it and load everything you have created in Step 2.

Locate the package:

tutorial.entity

and right click on it, and then click on:

SWAG New Bean -> New EJB

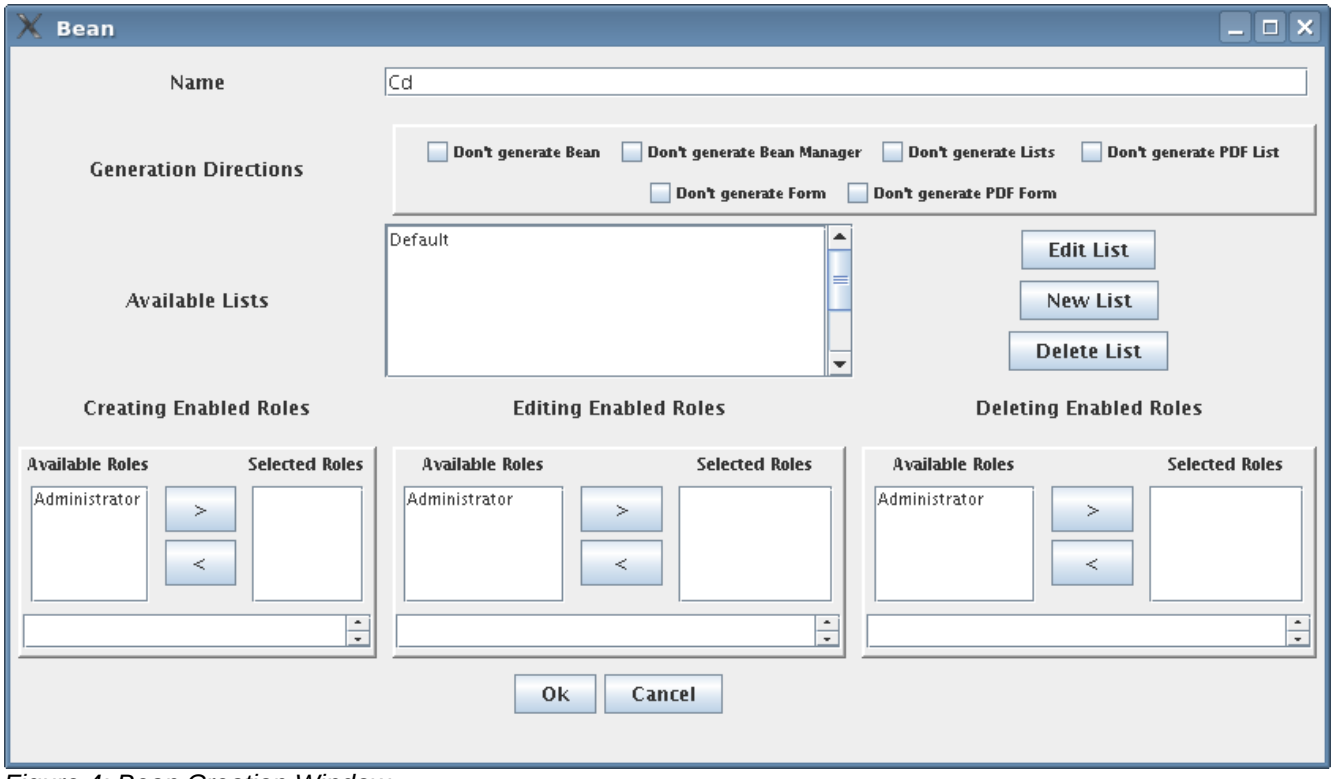


Figure 4: Bean Creation Window

A new window opens and you can enter the information related to the new bean to be created

(see Figure 4).

For this tutorial you will enter only the bean name (Cd).

Then select the Default list and click Edit List (see Figure 5)[1].

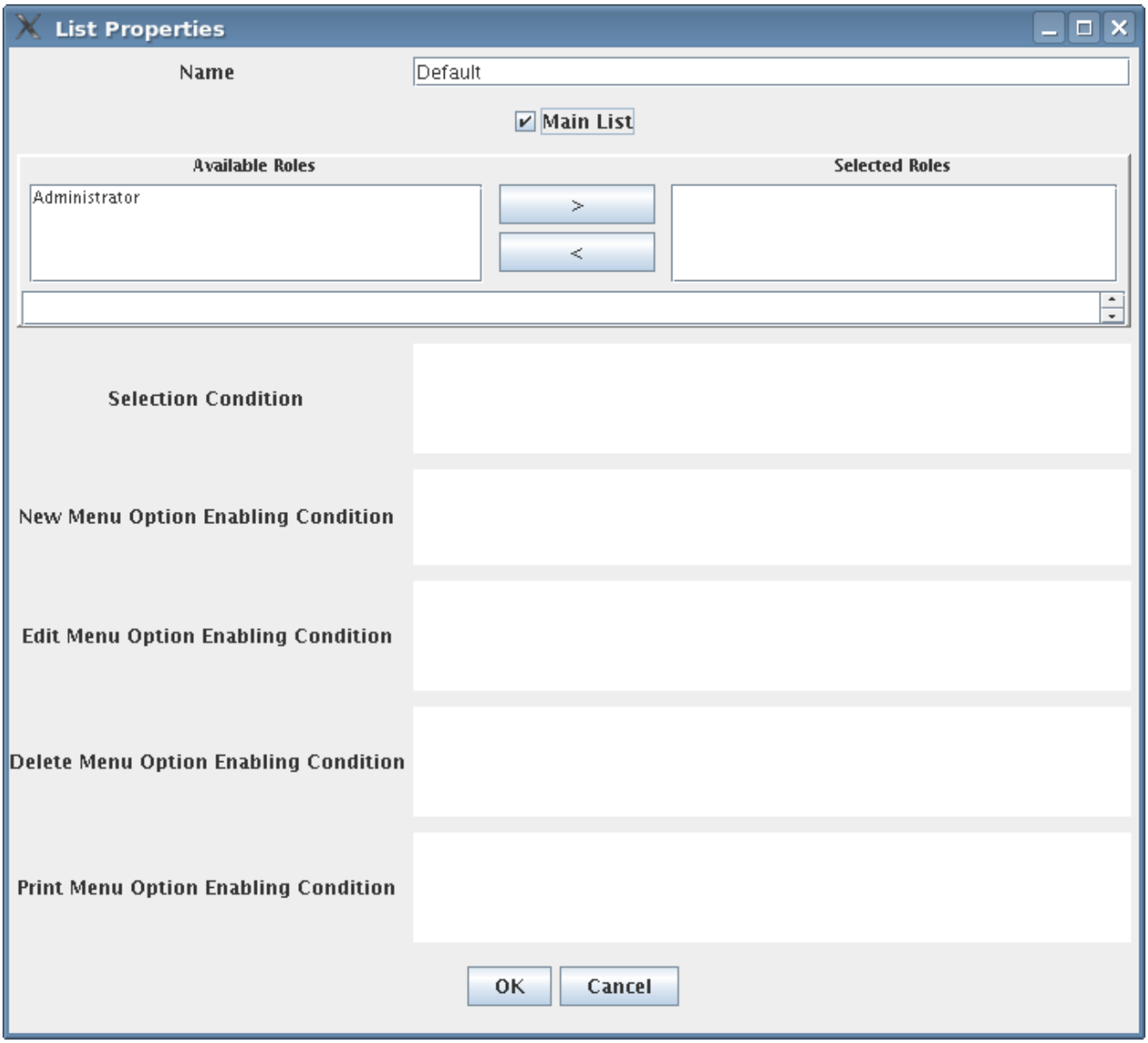


Figure 5: List definition Window

Check Main List option and then click Ok.

You have just created a new bean called Cd, the list of all Cd's will be shown as the main list in the application. At the moment, no security restrictions are applied to the bean.

1 Please, note that Edit Menu Option Enabling Condition and Delete Menu Option Enabling Condition are not used yet and they are disabled.

Your bean has no attribute fields yet. You will create them and select the related management options to inform the generator in a moment.

Open the Cd.java and add the following lines:

```
private String cdTitle;
private Date purchaseDate;
private Double price;
```

A this point you have to add “Getter” and “Setter” for each attribute. You can do it manually or using an Eclipse feature (press Shift+Alt+S and click on Generate Getter and Setter, Eclipse will set them up for you).

Check that the Eclipse Outline View is available (Window -> Show View -> Outline) and click on it.

Select the title field and right click on it, select SWAG -> Show In Form Window, Figure 6, is shown. Here you can define the property of title field in the form that is used for managing the bean.

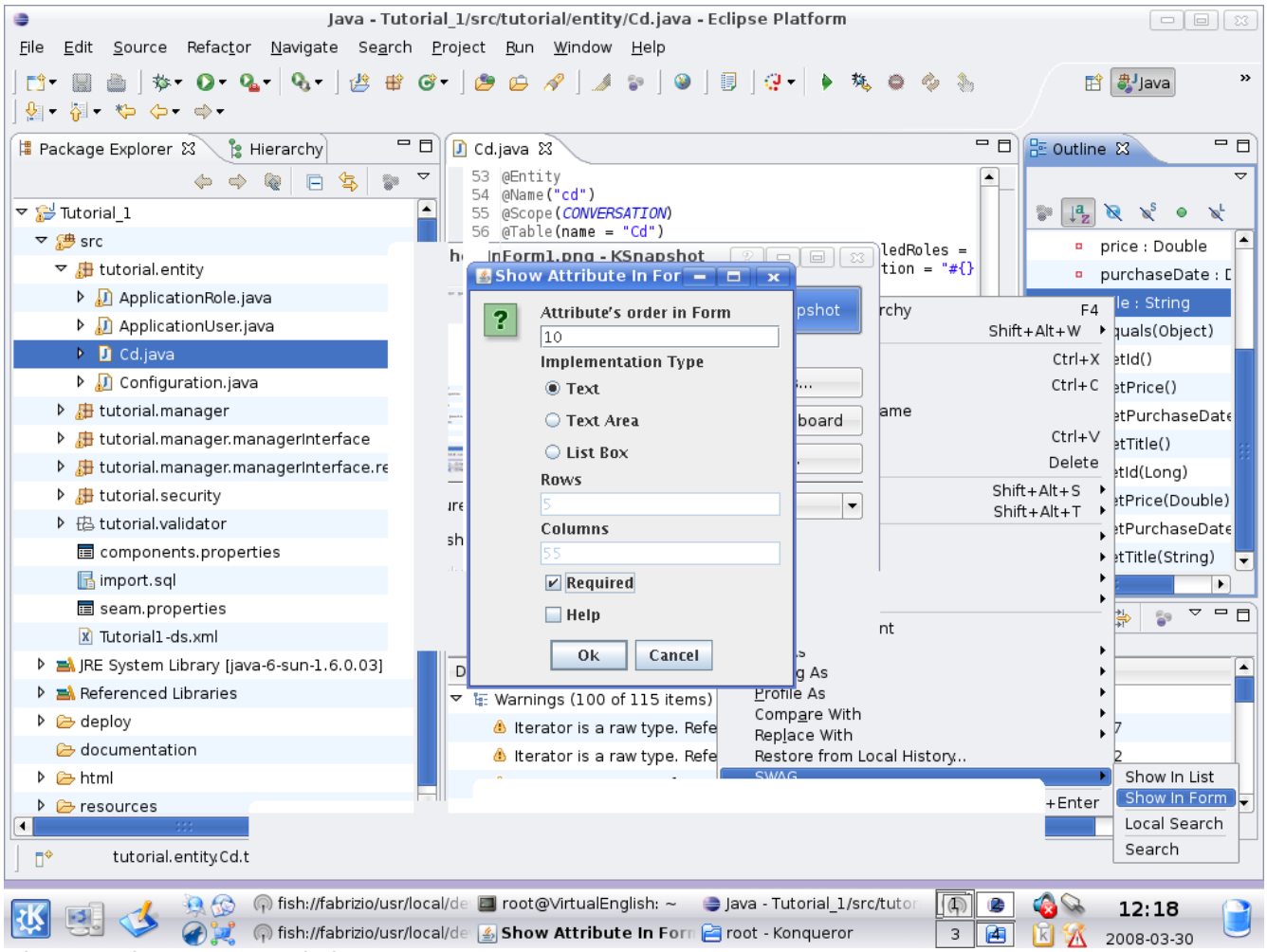


Figure 6: Show In Form Window

The “Attribute's order in Form” is the position of the title in the form, the implementation type instructs the generator to use a text field and “Required” obviously means that the field is required and it has to be filled before close the form.

Click on Ok and the field will be annotated as follows:

```
@ShowInForm(order = 10, type = AttributeImplementationType.TEXT, required = true)
private Double price;
```

proceed with each one of the other fields until you get:

```
@ShowInForm(order = 10, type = AttributeImplementationType.TEXT, required = true)
private Double     price;

@ShowInForm(order = 20, type = AttributeImplementationType.TEXT)
private Date       purchaseDate;

@ShowInForm(order = 30, type = AttributeImplementationType.TEXT)
private Double     price;
```

Now you have to define which field goes in the bean list and in which position. Locate the title field in the Outline View, right click and enter information in the appearing window (see Figure 7).

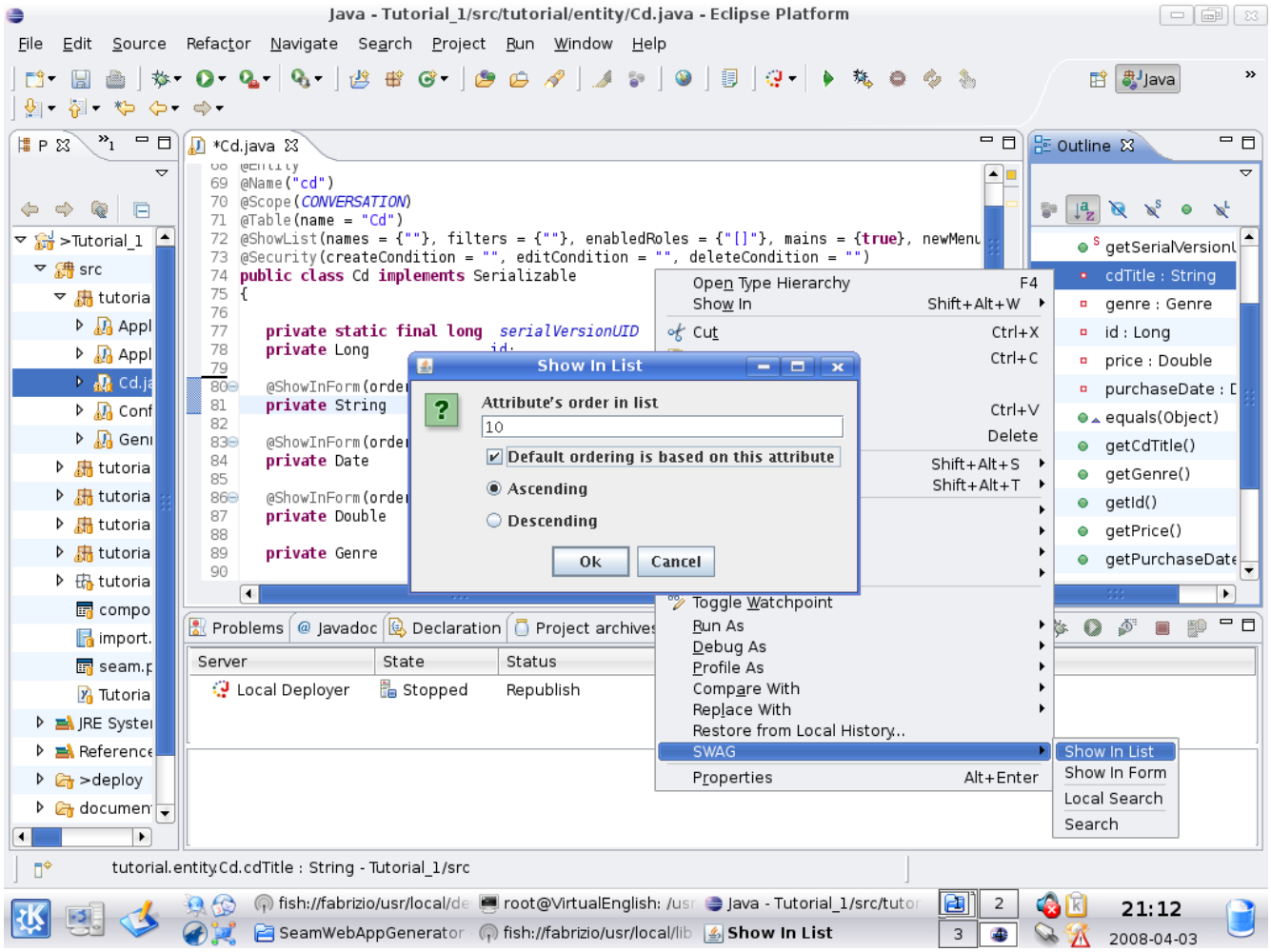


Figure 7: Show In List Window

At this point locate , into html/WEB-INF/classes directory, the file:

messages.properties ,

open it and edit the following keys accordingly:

```
Application_DateFormat = MM/dd/yyyy
Application_DecimalSeparator = .
Application_GroupingSeparator = ,
```

6.5 Step 5 – Generate your first application

Start the generator again:

java -jar SeamWebAppGen.jar

and click Ok.

The application is generated and ready for deployment.

6.6 Step 6 – Configure the application data source

Go to the source directory and locate the data source file:

Tutorial_1-ds.xml

open the file and write the database user and the related password:

```
....
<user-name>postgres</user-name>
<password>postgres</password>
....
```

6.7 Step 7 - Create your database

Now is time to create the database to store your data, select:

Start –> Programs--> PostgreSQL -> Command Prompt

to start a command window in which the PostgreSQL environment variables are correctly set.

Enter the following command:

createdb -h localhost -U postgres Tutorial_1

If you are using a different DBMS, you have to adapt the command as needed.

6.8 Step 8 – Deploy your application

Open the JBoss Server view in Eclipse (window --> show view --> other, write JBoss and select JBoss server view).

Right click on the view and select:

```
New Server
```

and then

```
JBoss Deploy-only server.
```

In the Deploy Directory field write

```
/usr/local/jboss-current/server/default/deploy                (Linux)
c:\usr\local\jboss-current\server\default\deploy               (Windows)
```

and then click finish.

Right click on the created server and select:

```
Add and remove projects ...
```

and add the Tutorial_1.ear project.

Right click again on the server and select

```
Publish
```

Now your application is deployed on the application server.

6.9 Step 9 – Start the JBoss application server

To start the JBoss application server issue the following commands (Linux platform):

```
cd /usr/local/jboss-current
./run.sh -b 0.0.0.0
```

or (Windows platform):

```
C:
cd \usr\local\jboss-current
run.bat -b 0.0.0.0
```

After about one minute, you should see the following message:

```
[Server] JBoss (MX MicroKernel) [4.2.2.GA (build: SVNTag=JBoss_4_2_2_GA date=200710221139)] Started in 27s:183ms
```

The application has been deployed successfully and it's running.

The required table have been automatically created into the database, before you can use the application you have to populate some tables.

6.10 Step 10 – Prepare database – Linux Platform

You have to populate some tables, related to users and profiles entering the following commands:

```
cd /usr/local/development/Tutorial_1/src/
psql -h localhost -U postgres -d Tutorial_1 -f import.sql
```

6.11 Step 10 – Prepare database – Windows Platform

You have to populate some tables, related to users and profiles.

Start a command prompt for PostgreSQL:

```
Start –> Programs--> PostgreSQL -> Command Prompt
```

then enter the following commands:

```
psql -h localhost -U postgres -d Tutorial_1 -f c:\usr\local\development\Tutorial_1\src\import.sql
```

6.12 Step 11 – Test the application

At this point you can test your application to check if it works. Open your favorite browser and go to the following URL:

```
http://localhost:8080/Tutorial_1
```

To log on to the application use:

- user: admin
- password: admin

You have done it. Your first Web application generated by SeamWebAppGen is now up and running. You can start interacting with it.

6.13 Step 12 – Refine your first application

In this step you will refine some of the graphical aspects of your new Web application.

First of all you have to set the application name. Open the message properties:

```
html/WEB-INF/classes/messages.properties
```

and enter new values for the following keys:

```
Application_name = SeamWebAppGen Tutorial
```

```
Application_copyright = Your Name

form_Cd_cdTitle = Title
form_Cd_price = Price $
form_Cd_purchaseDate = Purchase Date
list_Cd_cdTitle = Title
```

Then replace:

```
html/img/logo.jpg
```

with your favorite icon or image.

Now your application user interface should look like Figure 8.

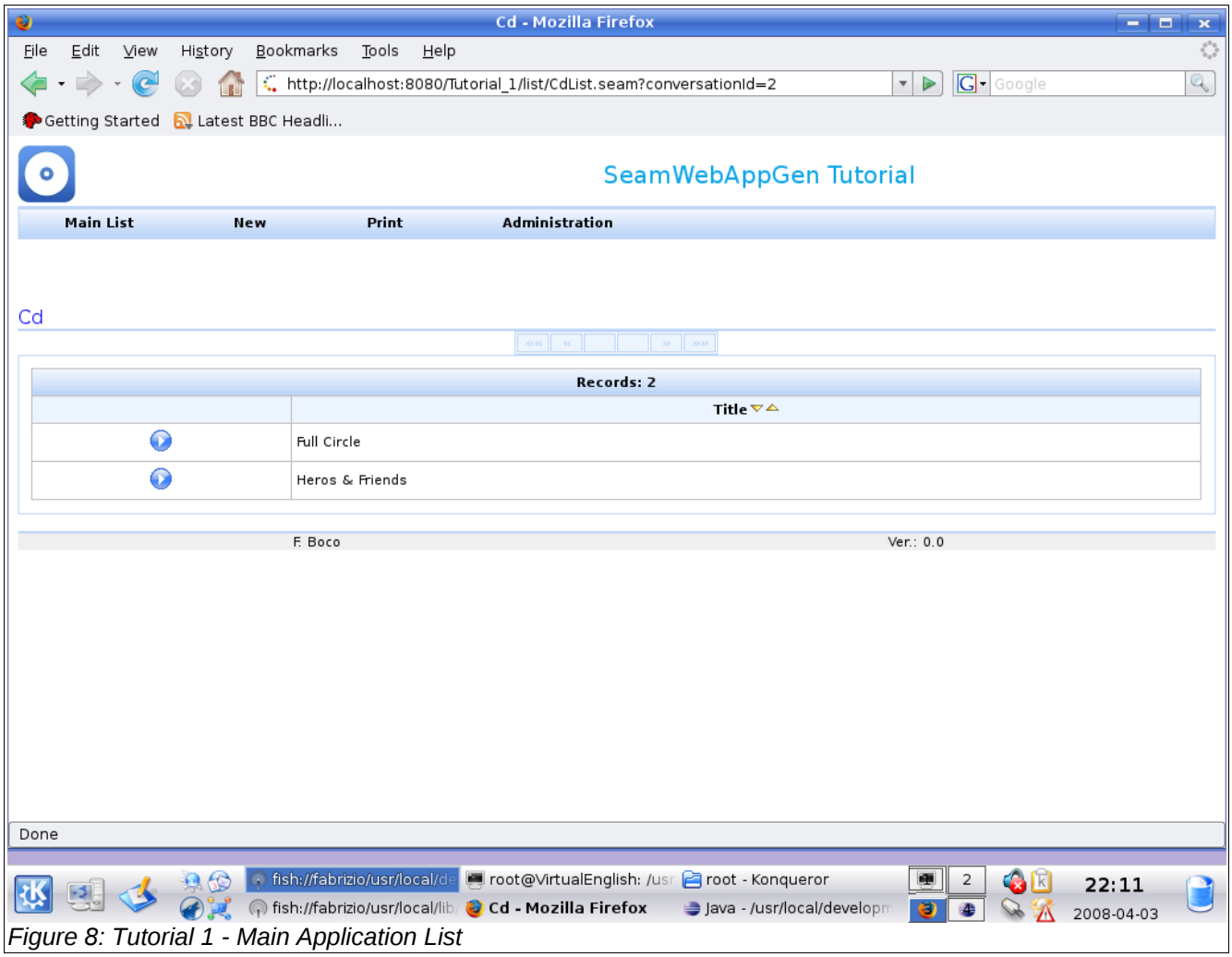


Figure 8: Tutorial 1 - Main Application List

7. Tutorial 2 – Relationships between business objects

In this tutorial you will learn how to create relationships between business objects and how to use main relationship implementation options that SeamWebAppGen make available.

From this point on, refer to the following Class model:

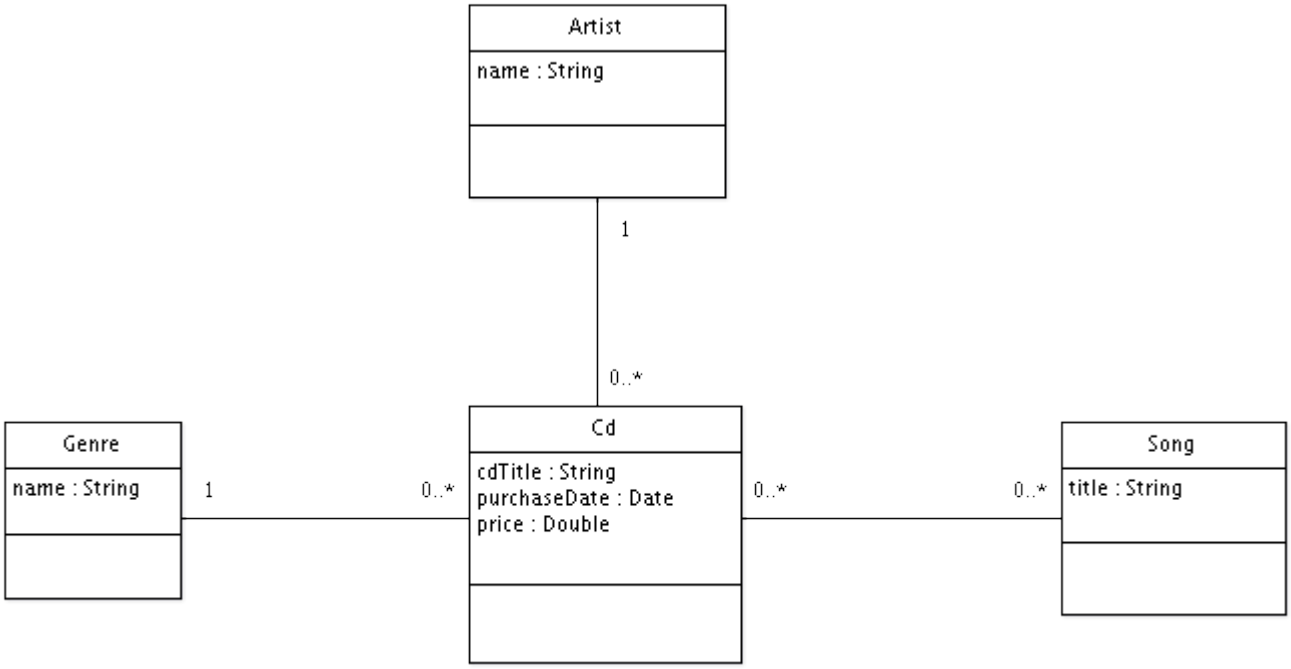


Figure 9: Tutorials Class Model

7.1 Step 1- Create a new tutorial from Tutorial_1

Let's select the Eclipse project, then right click on it and choose the copy option. When requested enter the new project name: Tutorial_2.

Locate and open the packaging file:

```
.packages
```

and replace all occurrences of Tutorial_1 with Tutorial_2.

Note: The file name starts with a dot so it can be hidden on Linux / Unix systems. An option for hiding resources that start with a dot is also provide by Eclipse.

Locate and open file:

```
html/WEB-INF/components.xml
```

and replace all occurrences of Tutorial_1 with Tutorial_2.

Locate and open files:

```
resources/META-INF/application.xml
resources/META-INF/persistence.xml
resources/META-INF/jboss-app.xml
```

and replace all occurrences of Tutorial_1 with Tutorial_2.

Locate file:

```
src/Tutorial_1-ds.xml
```

and rename it in

```
src/Tutorial_2-ds.xml
```

Open this last file and replace all occurrences Tutorial_1 with Tutorial_2 in

```
<jndi-name>Tutorial_1</jndi-name>
```

Don't replace Tutorial_1 in the <connection-url /> so that you can share the same database among the tutorials you expect to address.

Close the Tutorial_1 project and restart Eclipse.

That's it, now you have a second project on which you can do any kind of experimentations you like without destroying what you have done so far.

7.2 Step 2 - New beans – Genre & Artist

Following the same instructions of paragraph 6.3 create a bean called Genre, with the following fields:

```
@ShowAttributeInList(order = 10, isDefaultOrderingAttribute = true, ascending = true)
@ShowInForm(order = 10, type = AttributeImplementationType.TEXT,  required = true)
private String                          name;
```

Then create a bean called Artist with the following fields:

```
@ShowAttributeInList(order = 20, isDefaultOrderingAttribute = false, ascending = true)
@ShowInForm(order = 20, type = AttributeImplementationType.TEXT,  required = true)
private String                          name;
```

```
@ShowAttributeInList(order = 10, isDefaultOrderingAttribute = true, ascending = true)
@ShowInForm(order = 10, type = AttributeImplementationType.TEXT,  required = true)
private String                          surname;
```

7.3 Step 3 - Your first relationships – ListBox implementation

In this step we are going to create the relationship between Cd and Genre. As shown in the class model (Figure 9) there is a N:1 relationship from Cd to Genre.

To create this relationship you can use the Eclipse SeamWebAppGen plug-in again.

Locate the Cd bean (the source bean of the relationship) and right click on it. Then select:

```
SWAG New Relationships -> N:1
```

a new window will open as shown in Figure 10.

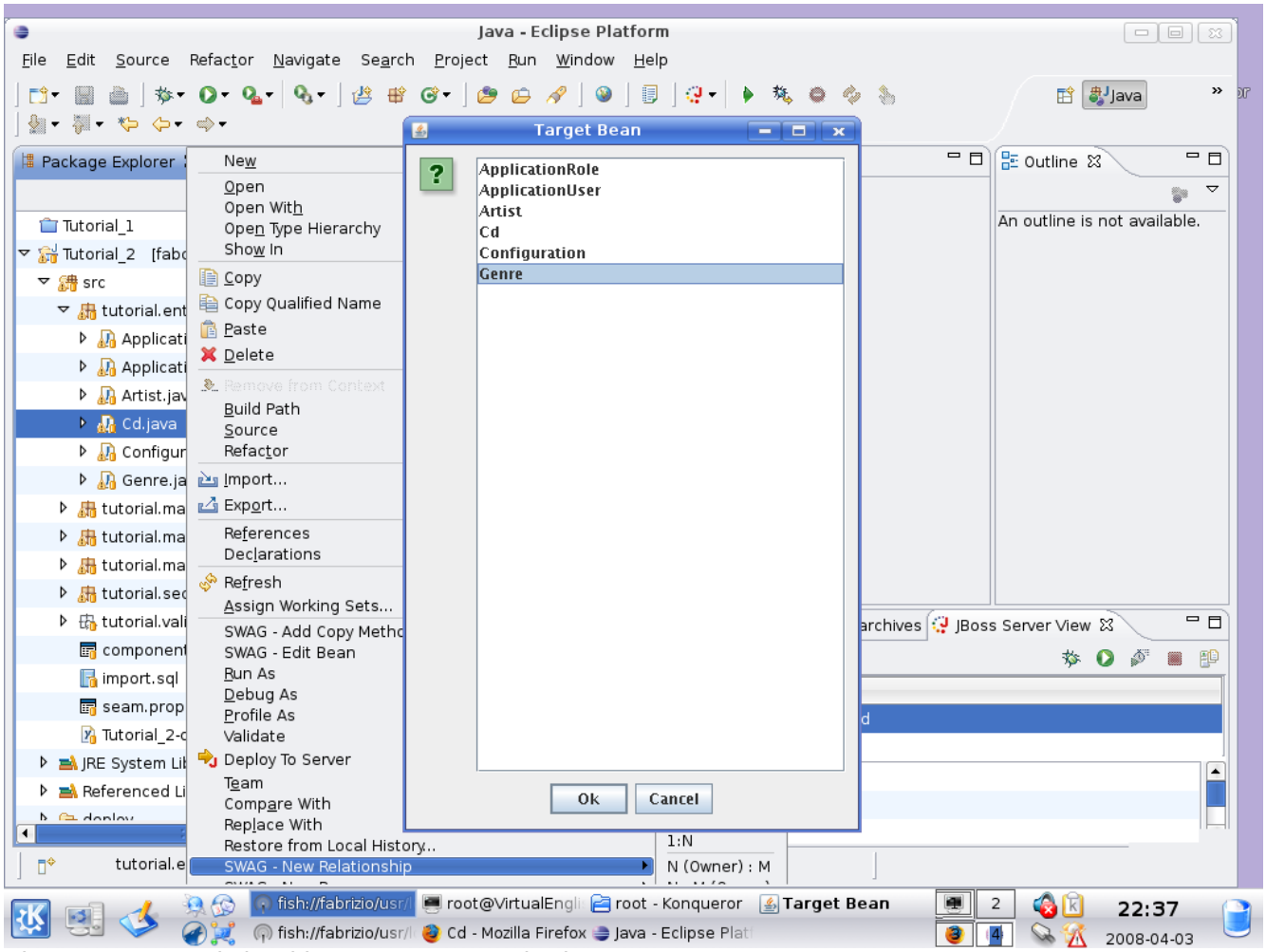


Figure 10: New Relationship - Target Bean Selection

Select the target bean (Genre) and click Ok. Now you have to define the properties of the new relationship (Figure 11).

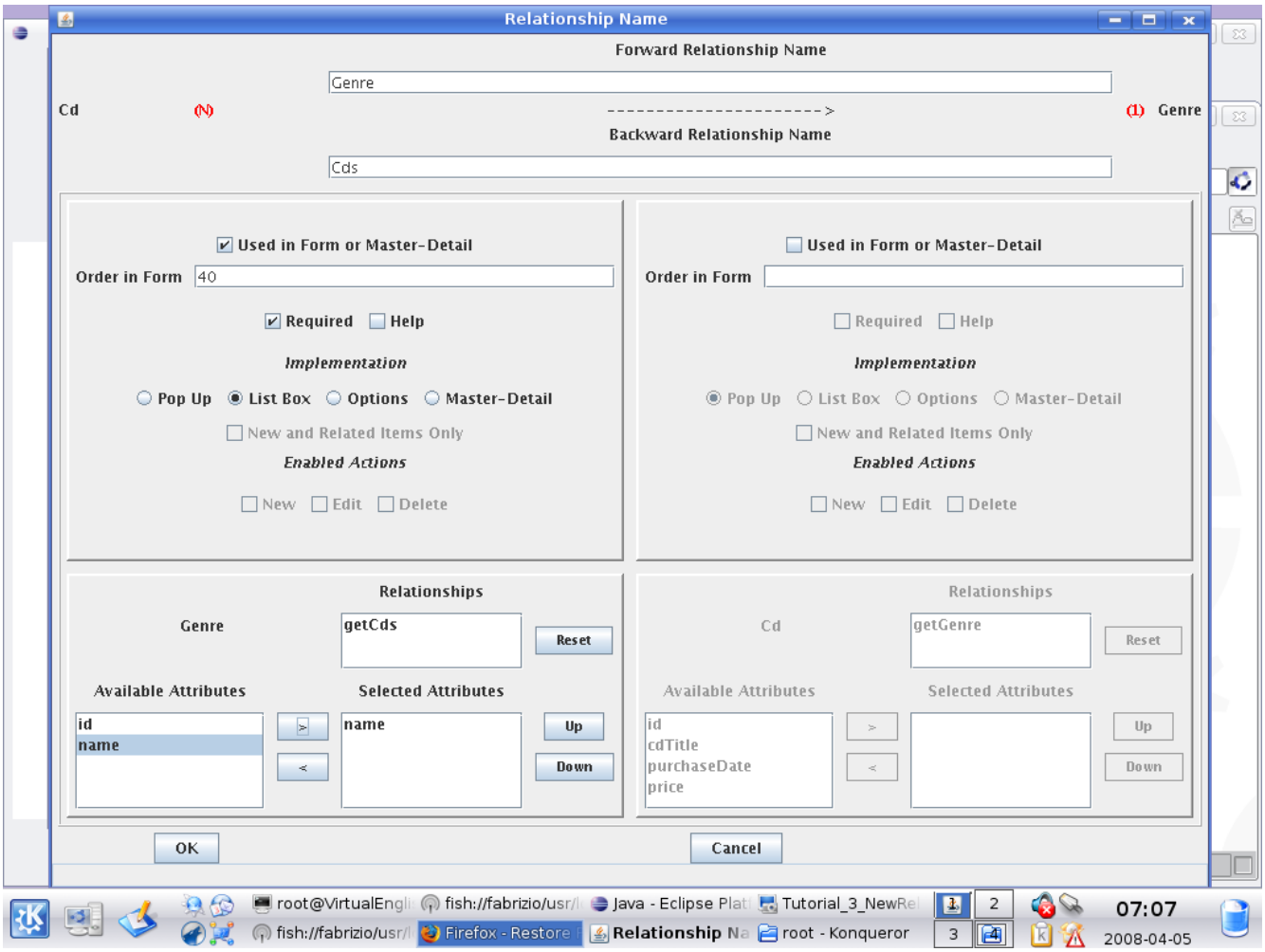


Figure 11: New Relationship - Properties

On the left side you have the relationship properties into the source bean and on the right side the relationship properties into the target bean.

Let the default forward and backward relationship names.

Select Used in Form or Master-Detail for the source bean. This means that SeamWebAppGen will insert the relationship into the form of source bean.

Enter 40 in the Order In Form, this will be the position of relationship in the source form.

Select Required, since each new Cd must to have a Genre.

Select List Box and SeamWebAppGen will generate a List Box for choosing the Cd genre.

Then you select name and then >. The relationship List Box will be fulfilled with the name of Genre.

Let the target side of relationship unselected. SeamWebAppGen will not generate anything in

the Genre form.

7.4 Step 4 – Customizing administration menu

Management of Genres will not be a day-by-day activity. So, the list of all Genres will be accessible only from the Administration menu.

Locate the Administration.xhtml file in the html project directory, open it and replace

```
<!-- <rich:menuItem immediate="true" value="#{msgs.list_XXXXX_title}"    action="/list/XXXList.seam" />  -->
```

with:

```
<rich:menuItem immediate="true" value="#{msgs.list_Genre_title}"    action="/list/GenreList.seam" />
```

This is the way you normally access any business object list.

7.5 Step 5 – Add a relationship attribute into the main application list

Now that your Cd is related to a specific Genre, probably you want to see the Cd's genre in the main application list closed to the Cd title.

It is very easy to achieve this. Locate the Cd bean and open it. Locate the relationship from Cd to Genre in the Eclipse outline window. Access the function:

```
public Genre getGenre()
{
....
}
```

Right click on it and select

```
SWAG -> Show in List
```

The following window will appear (Figure 12):

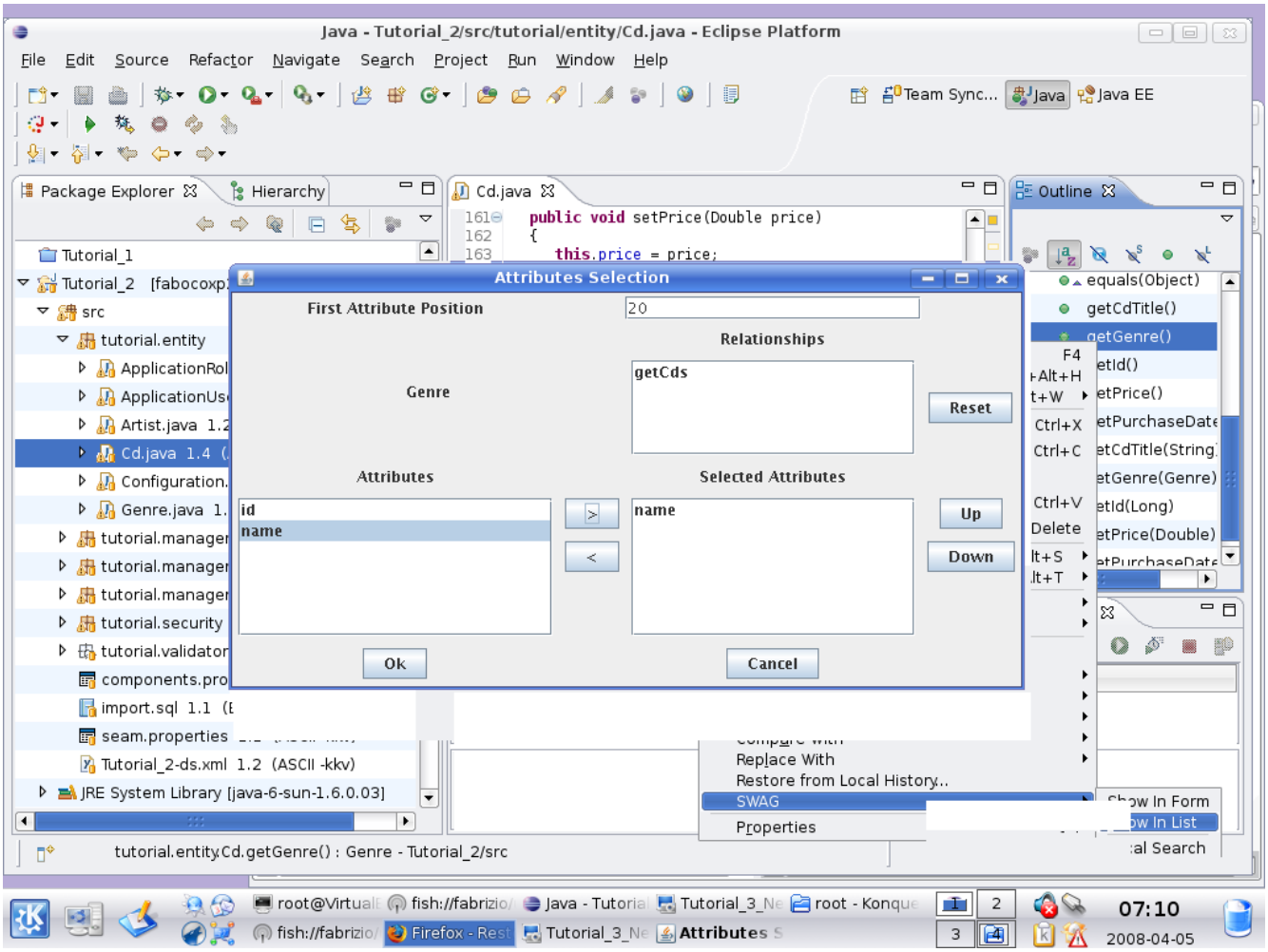


Figure 12: ShowRelationshipAttributesInList Window

Select the attribute that you need to address in the bean list (name in this case) and click Ok.

The method is now annotated with @ShowRelationshipAttributesInList too.

```
@ShowRelationshipAttributesInList(names={“name”}), orders={40}
----
public Genre getGenre()
{
....
}
```

This means that in the Cd list, the Genre name will be shown for each Cd.

7.6 Step 6 – Generate and deploy

At this point you are ready to generate the new application. Do that by following the instructions of paragraph 6.5 (don't forget to change directory name from Tutorial_1 to Tutorial_2).

Before proceeding to the generation, fix the following labels:

```
list_Cd_getGenre_name = Genre
```

Then compile and deploy the new application following the instructions shown in the paragraph 6.9.

7.7 Step 7 – Fill out Genre list

Start the application at

```
http://localhost:8080/Tutorial_2
```

and log in as administrator (password administrator).

Select

```
Administration -> Genre
```

You will be pointed to the Genre list that is empty. Select New and enter as much Genres as you like.

Now the Genre list contains the genres you entered (Figure 13).

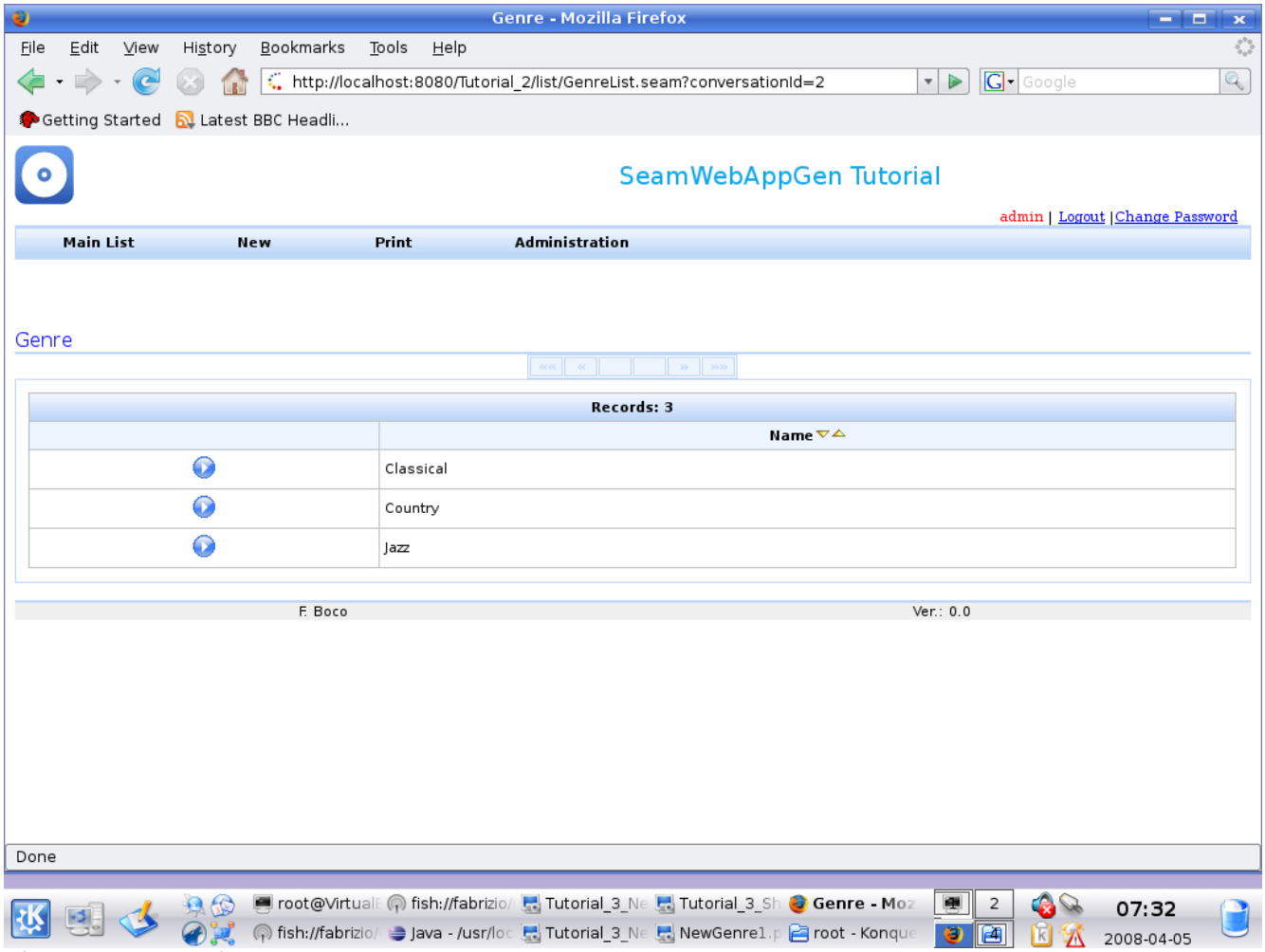


Figure 13: Genre List

Go to Main List, select a Cd and edit it. As you can see in Figure 14 there is a List Box that allows you to define the Cd's genre.

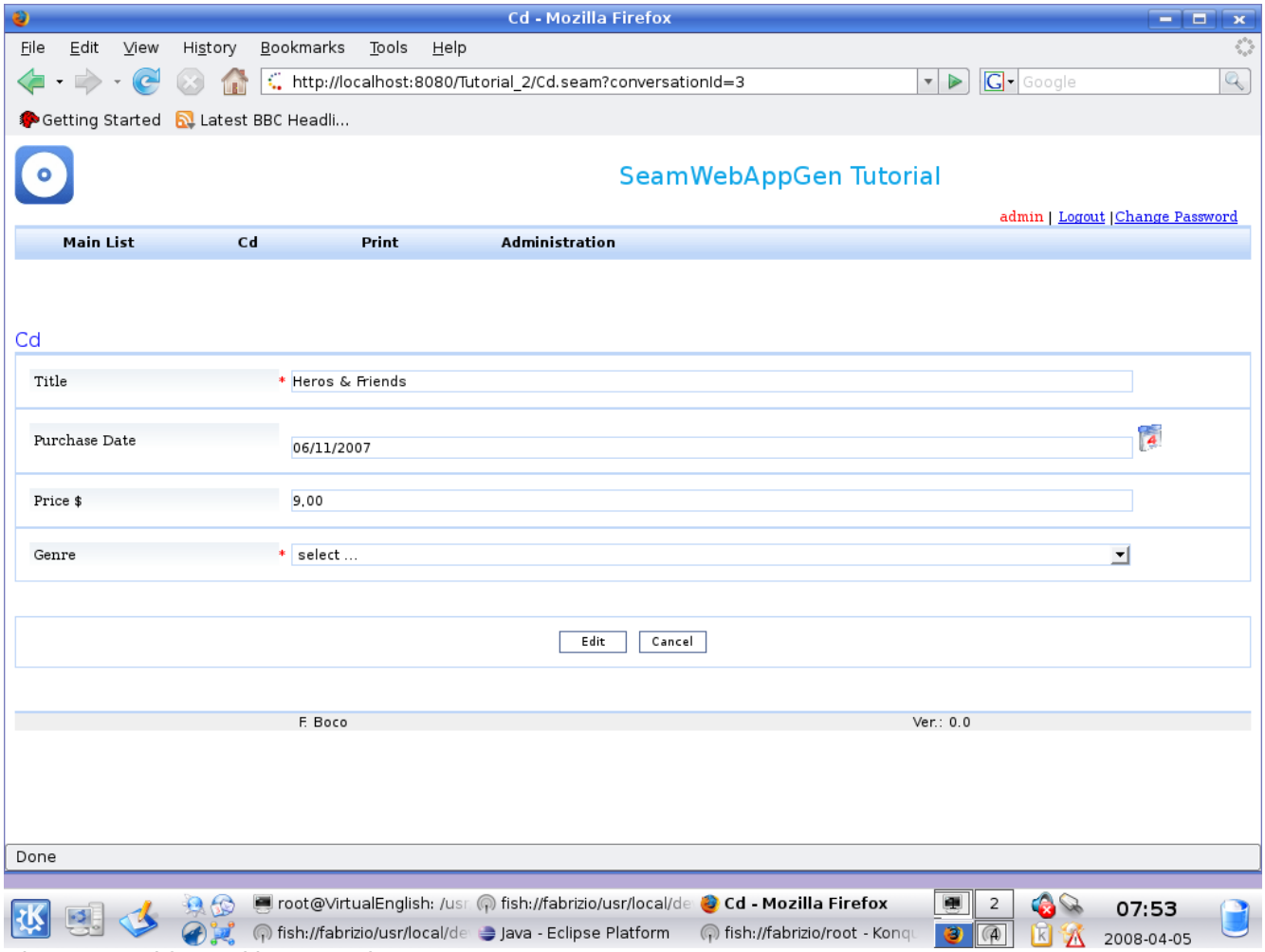


Figure 14: Cd form with Genre ListBox

If edit all your previously entered Cd, your main list look like Figure 15

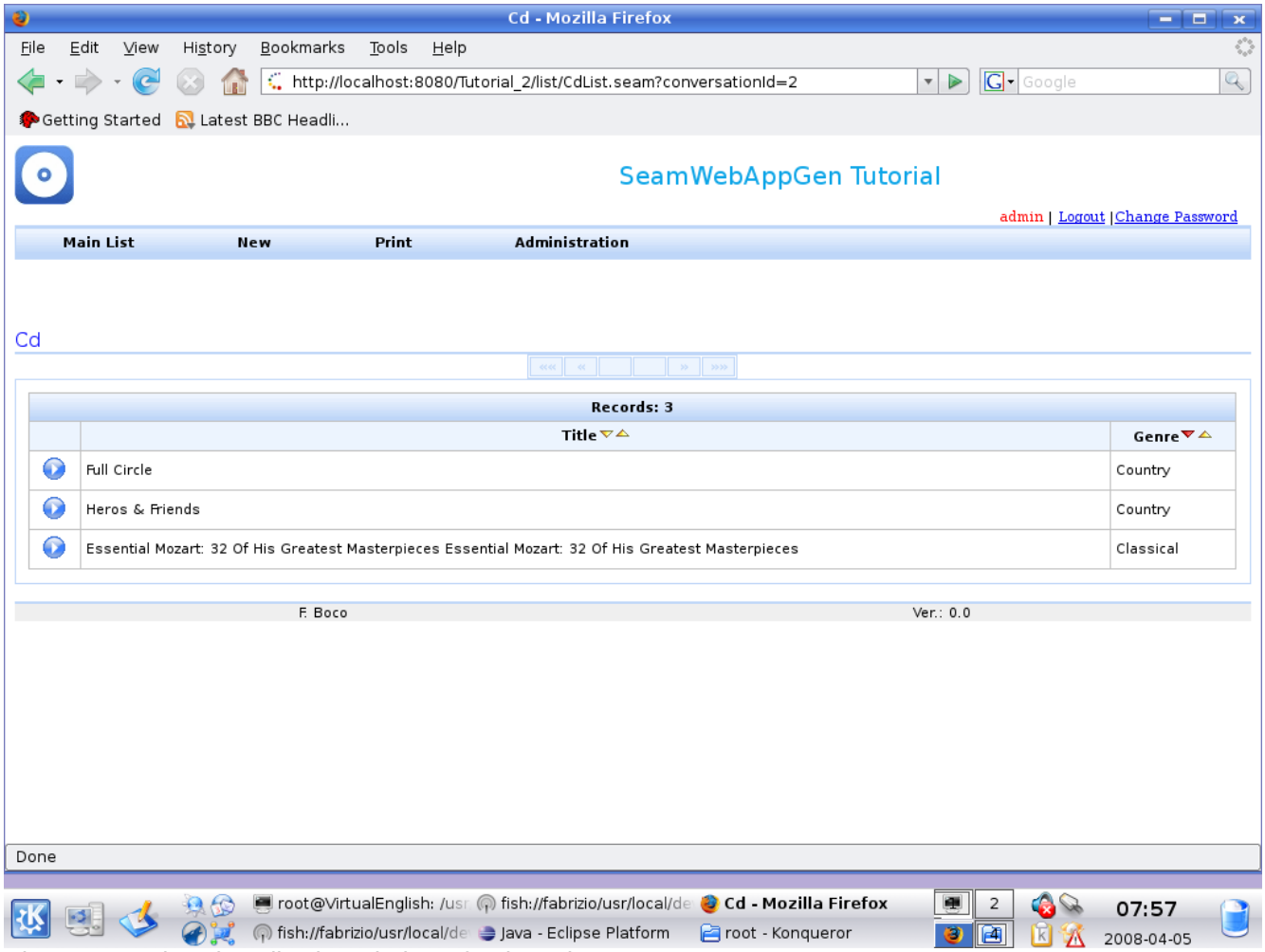


Figure 15: Main Cd application window after inserting Genre

7.8 Step 7 – Order Genre List Box by name – Customizing code

Locate the Genre Bean Manager in Eclipse and open it. Go to the function:

```
public List<Genre> getSelectionItems()
{
String query = "select o from Genre as o ";

// YOUR CODE GOES HERE
// END OF YOUR CODE

        List<Genre> tmp = em.createQuery(query).getResultList();

        return tmp;
}
```

and change it as follows:

```
public List<Genre> getSelectionItems()
{
String query = "select o from Genre as o ";

// YOUR CODE GOES HERE

        query += " order by o.name";

// END OF YOUR CODE

        List<Genre> tmp = em.createQuery(query).getResultList();

        return tmp;
}
```

The code between the tags

```
//YOUR CODE GOES HERE
//END OF YOUR CODE
```

will not be modified by SeamWebAppGen. From here on the list will be ordered by genre name.

Regenerate, compile and deploy your application, just to check that everything is working as expected.

7.9 Step 8 – Local search

As your Cd collection grows, it will not be easy to locate manually a specific Cd. The local search function will automate for you the search .

Locate Cd Bean and open it. Locate the attribute CdTitle in the Eclipse outline window and right click on it.

Select:

```
SWAG -> Local Search
```

and enter the position of the attribute in the local search form: 10.

Locate the relationship between Cd and Genre (the function getGenre) and right click on it.

Select again:

```
SWAG -> Local Search
```

in the window enter 10 as attribute order, select name and click Ok.

Now the method as the @LocalSearchOnRelationship:

```
@LocalSearchOnRelationship(names={“name”}), orders={20}
----
public Genre getGenre()
{
....
}
```

Enter the custom values for the following labels:

```
localSearch_Cd_getGenre_name = Genre
localSearch_Cd_cdTitle = Title
```

Regenerate the application, compile and deploy it.

Among the results of what you have just done, the main application window is provided with magnifier icon through which you can activate the local search (see Figure 14).

Since it will be useful in the next step of the tutorial, add local search to the Artist bean too. Locate the Artist bean, open it and following the above instructions for the Genre bean and add the local search to the name and the surname of Artist. At the end, you will obtain end up with the following coding:

```
@LocalSearchOnAttribute(order = 10)
@ShowAttributeInList(order = 10, isDefaultOrderingAttribute = true, ascending = true)
@ShowInForm(order = 10, type = AttributeImplementationType.TEXT, required = true)
private String                                          surname;

@LocalSearchOnAttribute(order = 20)
@ShowAttributeInList(order = 20, isDefaultOrderingAttribute = false, ascending = true)
@ShowInForm(order = 20, type = AttributeImplementationType.TEXT, required = true)
private String                                          name;
```

If you like, you can enter the @LocalSearchOnAttribute annotation directly into your code. The same applies to all annotations of your application.

7.10 Step 9 – Relationship – Pop up Implementation

In accordance with the class model (*see Figure 9 above)* you have to create a relationship between the Cd and Artist.

Before you go any further, let's consider the simple action of entering a new Cd. Typically, in this case you have to search for the artist and if she/he is not present in the application you have to enter the related information,

This scenario is supported by SeamWebAppGen by generating a pop up. Let's see how.

Following the instructions provided at the step 3, create a new relationship N:1 between Cd and Artist. To do this choose the Pup up option in the properties window (Figure 16).

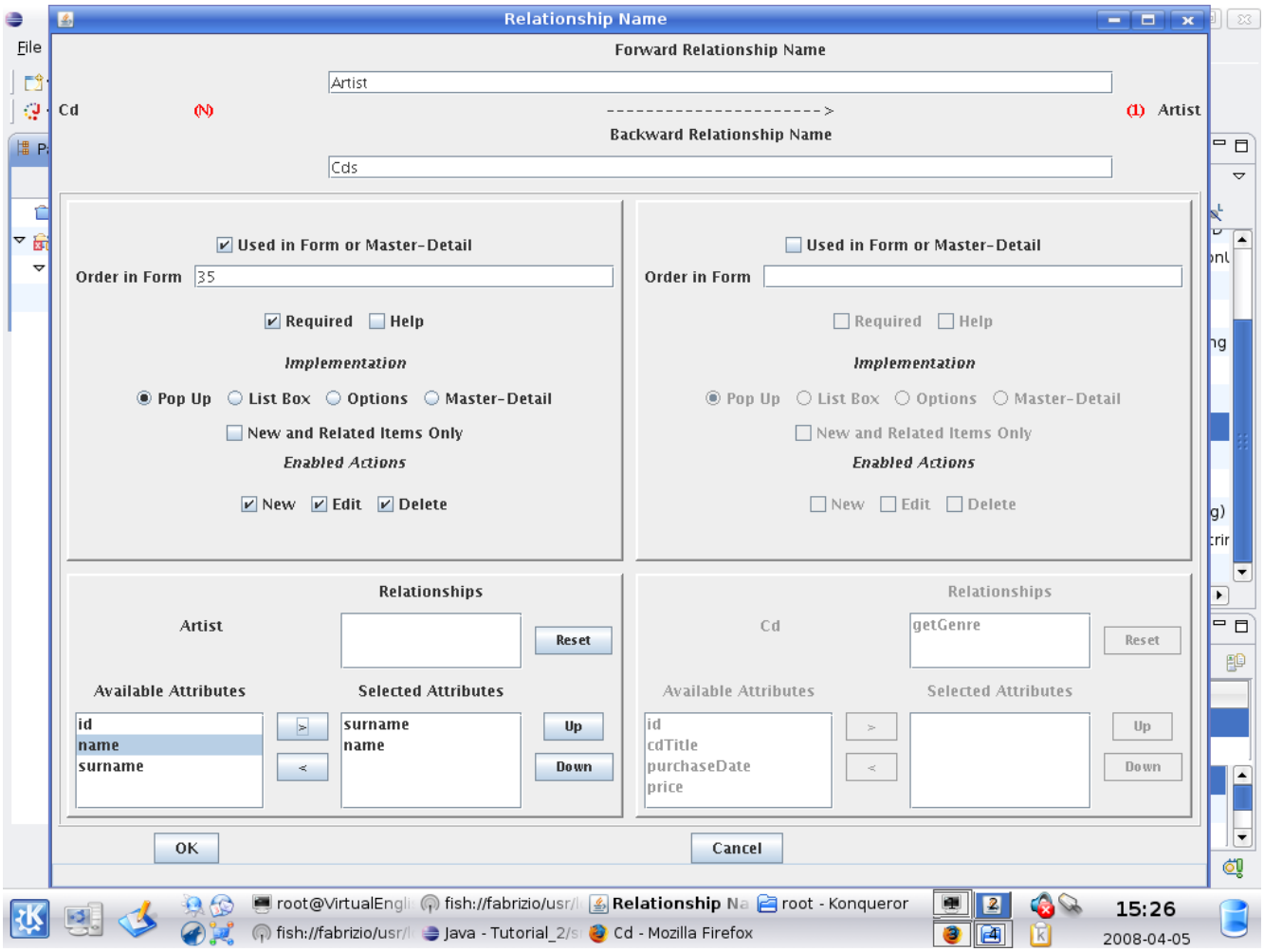


Figure 16: Pop up relationship - Properties window

Select all the actions (New, Edit, Delete) since you need to create, edit and delete artists during the entering / editing Cd phase.

Select the two attributes surname and name: they will be inserted into the Cd form and in the Pop up.

Then click Ok.

Before testing this new application version, now that we have the Artist of the Cd, it would be a good idea to show the Artist name and surname into the main application list. To do that add the annotation @ShowRelationshipAttributeInList:

```
.....
@ShowRelationshipAttributesInList(names = {"surname", "name"}, orders = {20, 30})
public Artist getArtist()
```

```
{
.....
}
```

locate the messages.properties file and change the following keys:

```
list_Cd_getArtist_surname=Surname
list_Cd_getArtist_name=Name
```

generate, compile and deploy this new version of the application.

Now editing one of the existing cd, you can see that into the form you have the name and surname of the Artist (Figure 16).

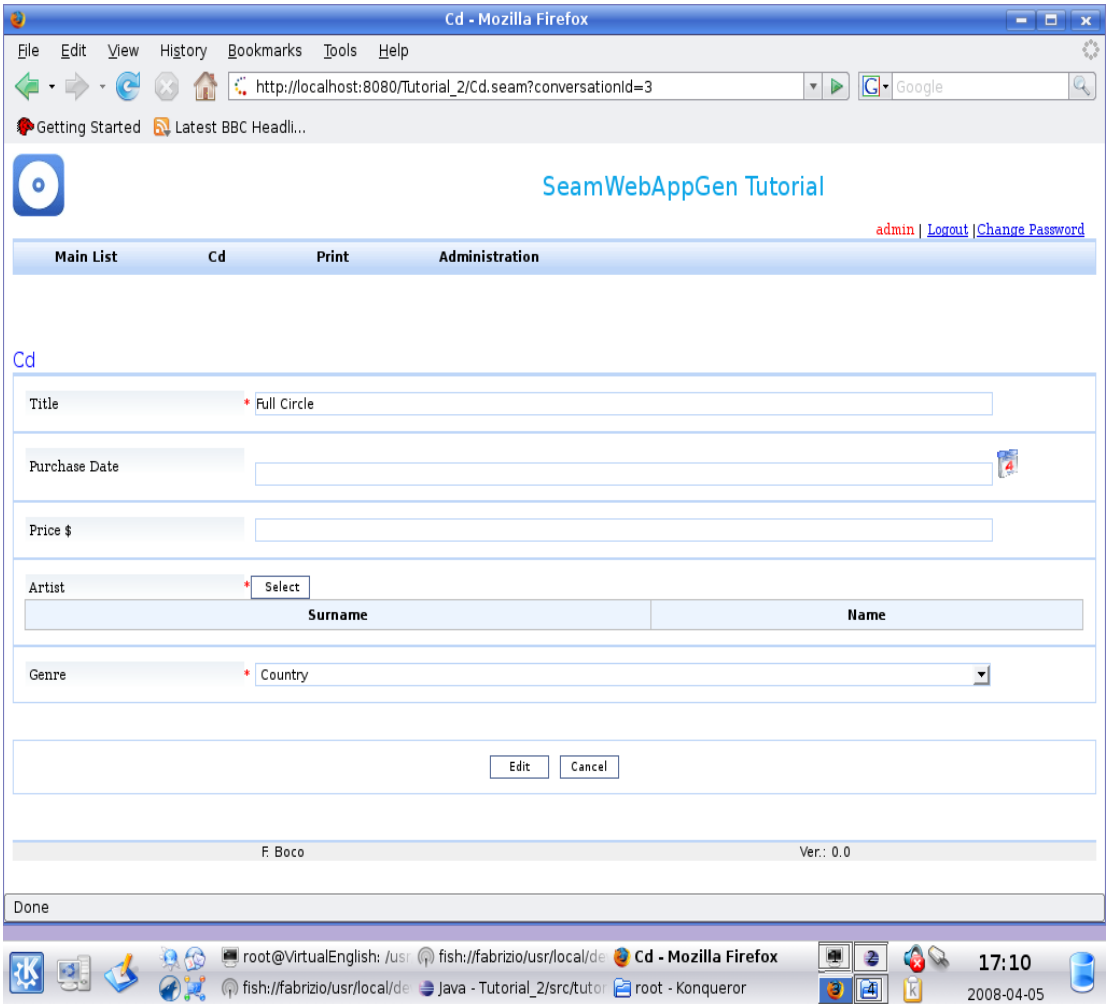
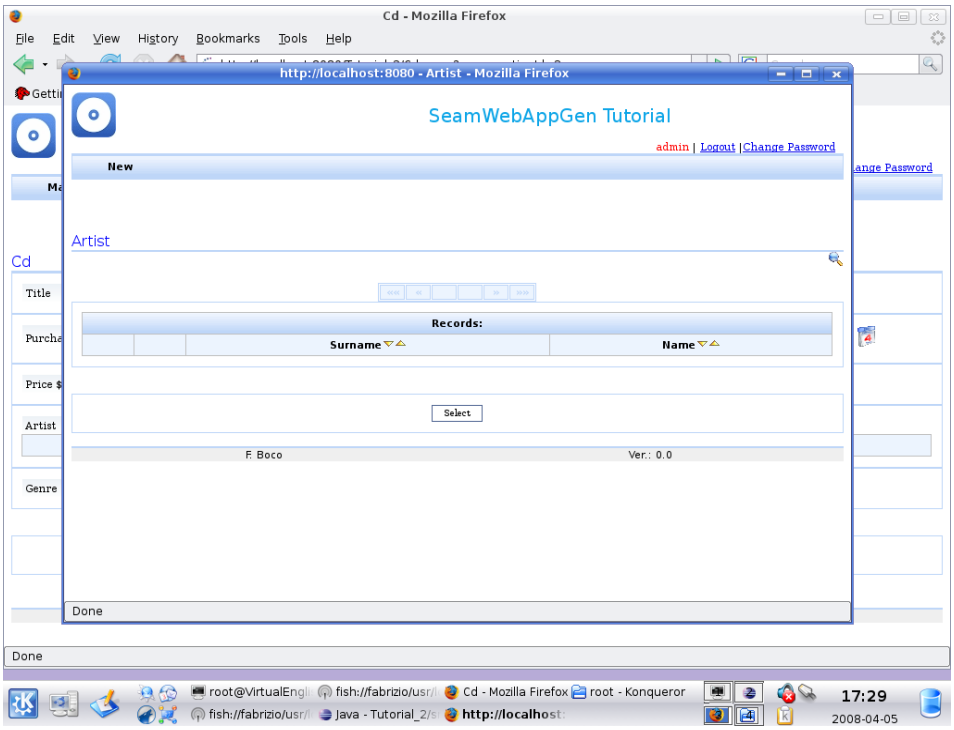


Figure 17: Relationship to Artist implemented as pop up

Click on the Select button and a new window will pop up to enable you to search for the Artist you want and the select her/him. If the Artist is not present in the application you will be able

to create it (Figure 17).

Figure 18: Pop up



Select New, enter surname and name of the new Artist and the click new. The new artist is created and automatically selected (Figure 18)

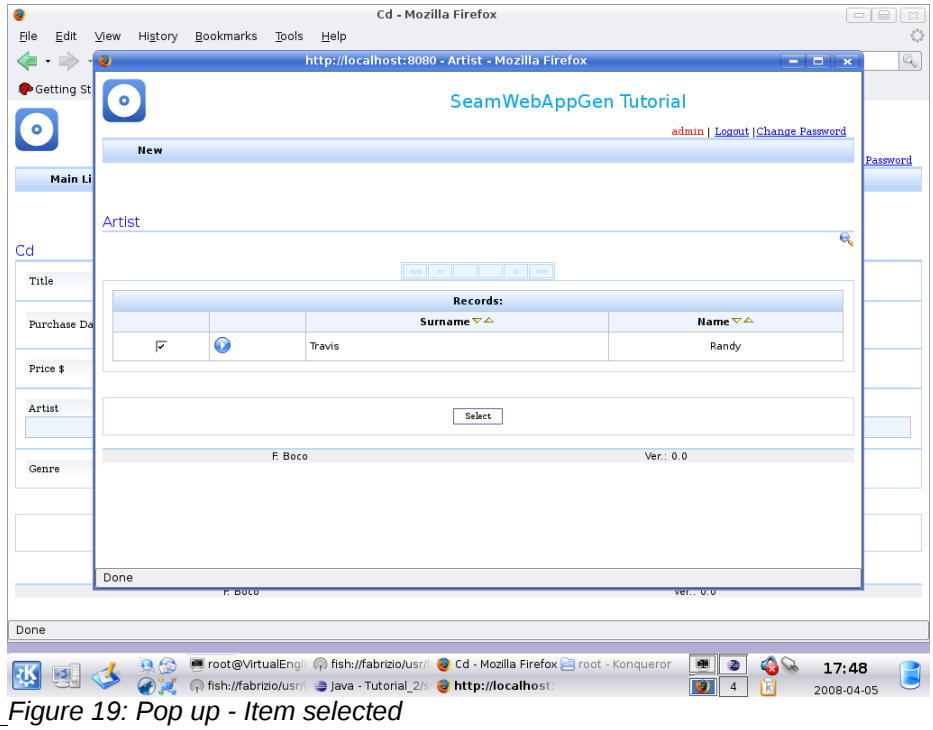


Figure 19: Pop up - Item selected

Click select and the selected Artist is inserted into the Cd form and it's related to it too.

8. Tutorial 3 – Master-Detail Relationship

The main application list contains a column in which the Genre of each Cd is present. Ordering that column you can group together all the Cd's with the same Genre.

Let's assume that your primary usage scenario is that you access the Cd information starting from the Genre. In this case a Master-Detail relationship between Genre and Cd may be more suitable.

That's what we are going to do in this tutorial.

8.1 Step 1- Create a new tutorial from Tutorial_2

Following the same instructions of paragraph 7.1 create a new project: Tutorial_3.

8.2 Step 2- Deleting the existing Genre – Cd Relationship

Unfortunately, although the SeamWebAppGen Eclipse plug-in can help you in the creation of a new relationship, it is of no help when changing its properties.

So, the first step is to manually delete the existing relationship.

Locate the Genre Bean, locate the Cds private property in the outline window and delete it.

When requested to remove the related getter and setter methods answer:"yes all".

Delete the following methods too:

```
public void addCds(Cd cds)
public void removeCds(Cd cds)
public void removeAllCds()
```

Locate the Cd Bean, locate the Genre property in the outline window and delete it answering "yes" to the question related to deleting getter and setter methods.

At this point there are no relationships between Genre and Cd.

Don't worry about errors that will show up in the Bean Managers. They will disappear as you regenerate the code.

8.3 Step 3- Creating a new Genre – Cd Relationship

Locate the Genre Bean and, following the instructions of paragraph 7.3, create a new 1:N relationship with Cd.

When the relationship details windows appear enter the information shows in Figure 20.

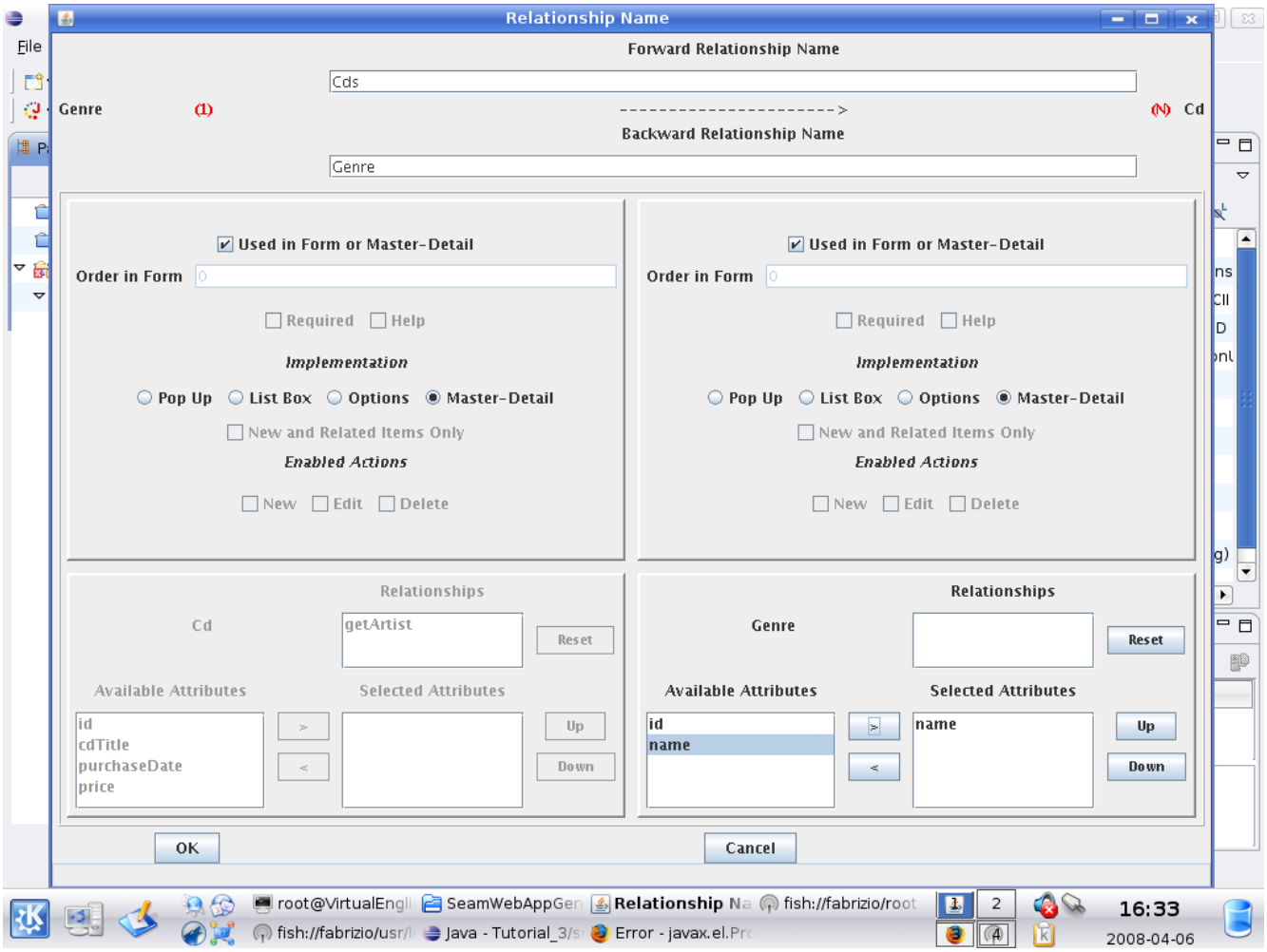


Figure 20: New Master-Detail Relationship

Now the main application list doesn't contain Cd anymore, it contains Genre instead.

Locale the Cd Bean, select and right click on it, then select:

SWAG – Edit Bean

A window appears and you can edit the Bean properties, select the Default list end edit it, changing the main option, then click Ok. Do the same change to the Genre Bean. As a result, the annotations for the two beans should be*:*

```
.....
@ShowList(names = {""}, filters = {""}, enabledRoles = {"[]"}, mains = {false}, newMenuOptionConditions = {""},
printMenuOptionConditionsInList = {""})
@Security(createCondition = "", editCondition = "", deleteCondition = "")
public class Cd implements Serializable
{
.....
}


.....
@ShowList(names = {""}, filters = {""}, enabledRoles = {"[]"}, mains = {true}, newMenuOptionConditions = {""},
printMenuOptionConditionsInList = {""})
@Security(createCondition = "", editCondition = "", deleteCondition = "")
public class Genre implements Serializable
{
.....
}
```

Generate, compile and deploy the application.

Now the main application list is Genre and from the context menu of each Genre you can access the Cds which have that specific and only that Genre (Figure 21).

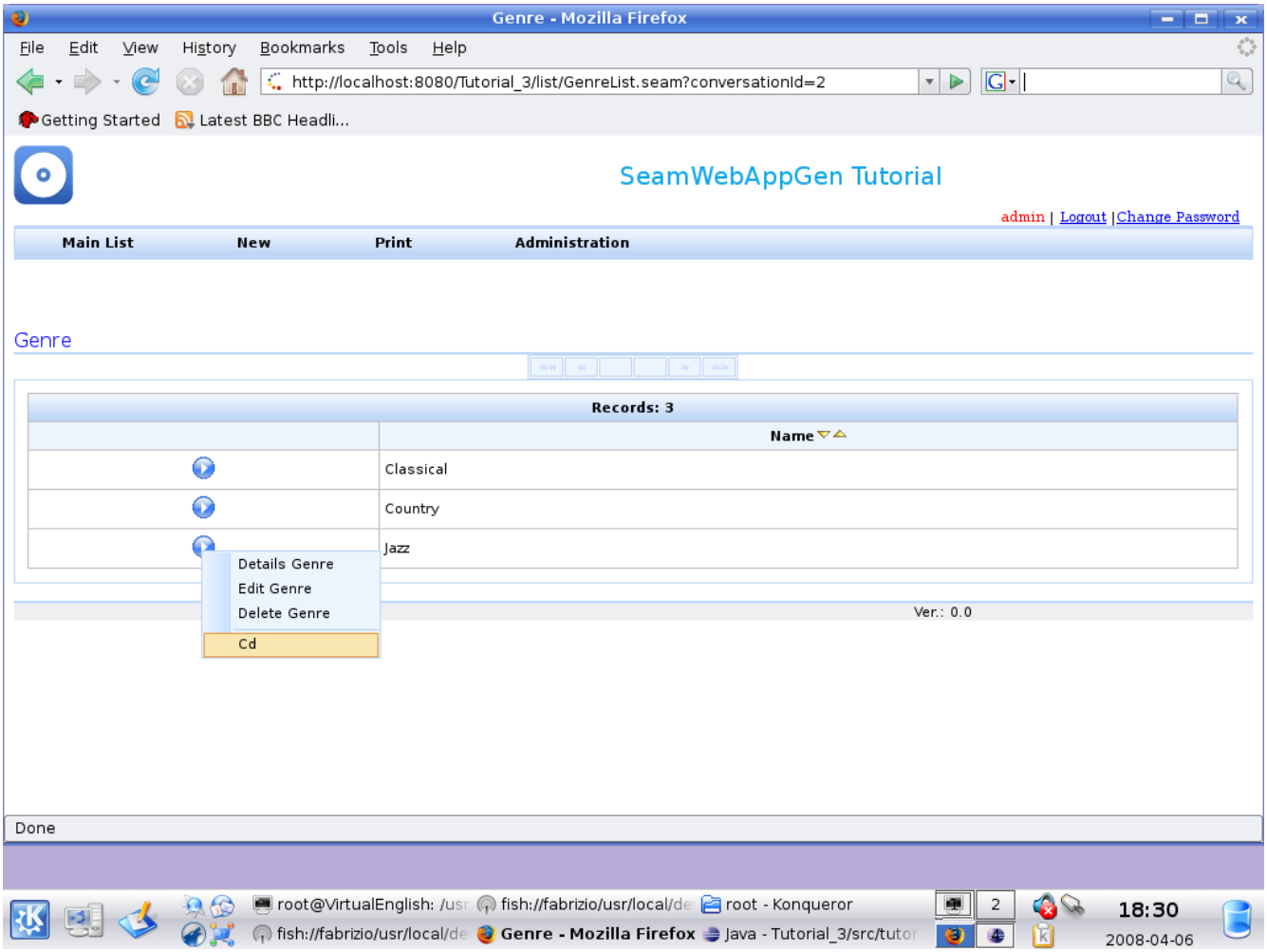


Figure 21: Access the Detail (Cd) from the Master (Genre)

9. Tutorial 4 – Validation & Custom Validation

This tutorial will be available in one of the future versions of this document.

10. Tutorial 5 – Managing Security

This tutorial will be available in one of the future versions of this document.

11. Tutorial 6 – More Lists for the same Bean

This tutorial will be available in one of the future versions of this document.