Adichatz

Version 0.8.8 of 03-10-2015

Current version of Adichatz plugins is a beta version i.e. a development version.

www.adichatz.org

Sommaire

1 Introduction

1.1 Foreword

An ideal application should be:	**Adichatz**, in its own ways, gathers all these features:
➢ Rich and efficient to bring maximum of added value to user.	➢ Final application rallies Eclipse 4,5 Mars rich features, many SWT, Forms and Nebula controls and advanced Adichatz components.
➢ Flexible and pluggable to ensure future requirements.	➢ Adichatz components are customizable and extensible. Your development can be divide in structured plugins.
➢ Maintainable to avoid entropy.	➢ Most of development process is to drive generation tools, assemble components and customize them.
➢ End-user and developer friendly to provide an easy support.	➢ No particular skills are needed to develop and delivery your first application.

Adichatz covers a very broad spectrum of development needs for a Company:

- ➢ After beginner creates its database, without any knowledge of Java, it is possible to create and customize a RCP application in a 3-tier or Client Server context. Similarly, the difficult phase for creating layout in java is radically simplified by the use of MigLayout Layout manager.
- ➢ A Java specialist use its own components, will build their own scenarios. He very quickly will design an application with rich and flexible ergonomics. However, his work remains flexible, maintainable and scalable.
- ➢ We invite Eclipse experts to reflect on the matter of how to make Eclipse 4 RCP features and 3-tier technologies easy to be used and shareable. We believe that our approach is a real response.

1.2 Purpose of this document

This document is intended to introduce Adichatz product. It shows:

- ➢ How to install Adichatz framework.
- ➢ How to create your first Adichatz application.
- ➢ First manipulations to customize the project.
- ➢ Main features.

1.3 Get a deeper insight

1.3.1 Prerequisites

- ➢ Java version 7 from web site.
- ➢ A SQL database with JDBC driver. You can download the example here. This test database is an arranged version of the "sakila" test database provided by MySql.

1.3.2 Installation

- ➢ WildFly 9.0.1.Final (recommended): download and unzip (**10 mn**).
- ➢ Choose a version of Eclipse IDE from Eclipse Mars download web site (**15mn**):
  - • Eclipse for RCP and RAP developers 4.5(Mars) for Windows (258 MB).

  or
  - • Eclipse IDE for Java developers 4.5 (Luna) for Windows (168 MB).
- ➢ Install a database: MySql (recommended) or Postgresql (**15mn**).

1.3.3 Development and deployment

- ➢ Generate my first complete application: **3mn**.
- ➢ Customize my application: **5mn** to **1h**).
- ➢ Deliver an Eclipse executable application: **5mn**.

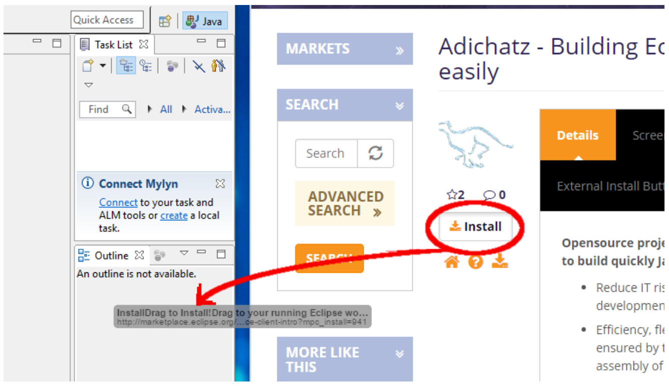
2 *Use of Adichatz*

2.1 Installation

2.1.1 Install Adichatz plugins

2.1.1.a Use 'drag to install' (1rst option)

1. Start Eclipse.
2. Open Adichatz page in Eclipse marketplace.
3. Drag Install button inside your eclipse workspace as shown beside.

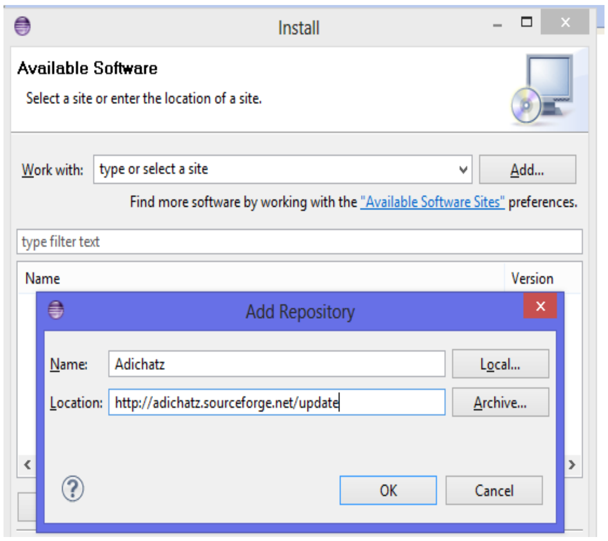

Quick Access
Java
Task List
Find
All
Activa...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
InstallDrag to Install!Drag to your running Eclipse wo...
MARKETS
SEARCH
Search
ADVANCED SEARCH
MORE LIKE THIS
Adichatz - Building Ec easily
Details
External Install But
Install
Opensource proje to build quickly J
Reduce IT ris developmen
Efficiency, fle ensured by t assembly of

2.1.1.b Install from Eclipse update site (2nd option)

Open Eclipse and start the install manager:
Help / Install new Software...

A window opens.
Add a new Site clicking button Add... .

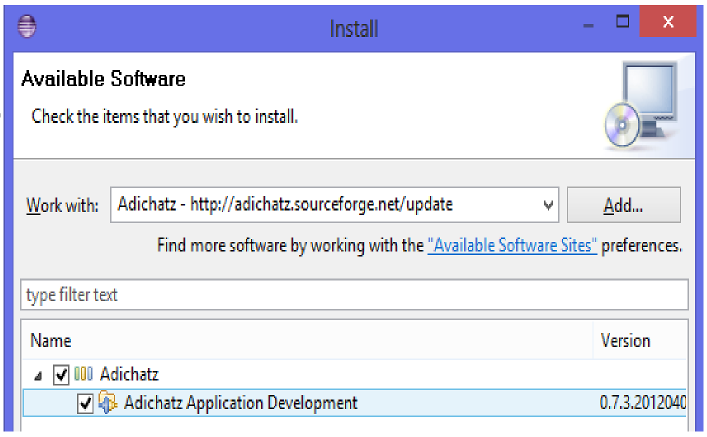
For eclipse 4.5 **Mars**, type:
Name: Adichatz
Location: http://adichatz.sourceforge.net/update
Click on button **OK** .


Install
Available Software
Select a site or enter the location of a site.
Work with: type or select a site
Find more software by working with the "Available Software Sites" preferences.
Add Repository
Name: Adichatz
Location: http://adichatz.sourceforge.net/update

Then select all items,

Click on button next at the bottom, then accept license, click **Finish** .

Valid warning message on unsigned content.
Adichatz plugins are installing.


Install
Available Software
Check the items that you wish to install.
Work with: Adichatz - http://adichatz.sourceforge.net/update
Find more software by working with the "Available Software Sites" preferences.

Name	Version
☑ Adichatz	
☑ Adichatz Application Development	0.7.3.2012040

2.1.2 Install WildFly (optional but recommended) -

Adichatz was tested with WildFly 8 and 9, JBoss 7.1.1, EAP 6.1 and 4.2.3 (JEE or 3-tier context). You can choose to use client/server version (JSE context). Adichatz provides in standard a plain solution for managing locks in JEE context so WildFly 9.0.1 is recommended.

Download **WildFly 9.0.1 final** (optional but recommended) from web site.
Unzip downloaded zip file on your computer in a specific folder (e.g. C:\ApplicationServer).

2.1.3 Install JBoss Tools plugins for Eclipse (optional)

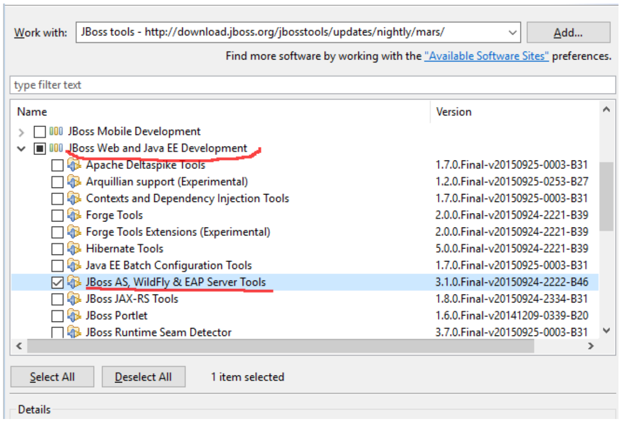
Start the install manager:

Help / Install new Software...

As for Adichatz plugin add a new site:

Name: JBoss
Location: http://download.jboss.org/jbosstools/updates/nightly/mars/

Select only item JBossAS Tools
In category JBoss Web and Java EE Development, select item Jboss AS, WildFly & EAP Server Tools (only this one is needed).



Work with: JBoss tools - http://download.jboss.org/jbosstools/updates/nightly/mars/

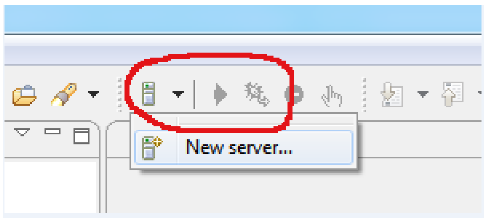
Find more software by working with the "Available Software Sites" preferences.

type filter text

Name	Version
☐ JBoss Mobile Development	
■ JBoss Web and Java EE Development	
☐ Apache Deltaspike Tools	1.7.0.Final-v20150925-0003-B31
☐ Arquillian support (Experimental)	1.2.0.Final-v20150925-0253-B27
☐ Contexts and Dependency Injection Tools	1.7.0.Final-v20150925-0003-B31
☐ Forge Tools	2.0.0.Final-v20150924-2221-B39
☐ Forge Tools Extensions (Experimental)	2.0.0.Final-v20150924-2221-B39
☐ Hibernate Tools	5.0.0.Final-v20150924-2221-B39
☐ Java EE Batch Configuration Tools	1.7.0.Final-v20150925-0003-B31
✓ JBoss AS, WildFly & EAP Server Tools	3.1.0.Final-v20150924-2222-B46
☐ JBoss JAX-RS Tools	1.8.0.Final-v20150924-2334-B31
☐ JBoss Portlet	1.6.0.Final-v20141209-0339-B20
☐ JBoss Runtime Seam Detector	3.7.0.Final-v20150925-0003-B31

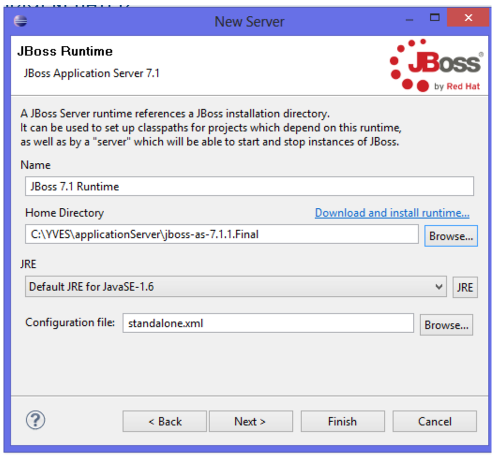
1 item selected

2.2 Customize Installation



Three new icons appear in Tool bar.
Select **New Server...** option.

A new window appears.



New Server

JBoss Runtime
JBoss Application Server 7.1

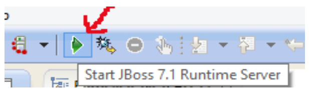
A JBoss Server runtime references a JBoss installation directory.
It can be used to set up classpaths for projects which depend on this runtime, as well as by a "server" which will be able to start and stop instances of JBoss.

Name: JBoss 7.1 Runtime
Home Directory: C:\YVES\applicationServer\jboss-as-7.1.1.Final
JRE: Default JRE for JavaSE-1.6
Configuration file: standalone.xml

In JBoss Community category select **WildFly 9.0.1 final** item.
Click **Next...** .

Specify JBoss AS 7.1.1 home directory.
Specify the JRE you want to use for AS server.
Click Finish .

You can launch and stop WildFly server from Eclipse



2.3 Creating my first Adichatz project

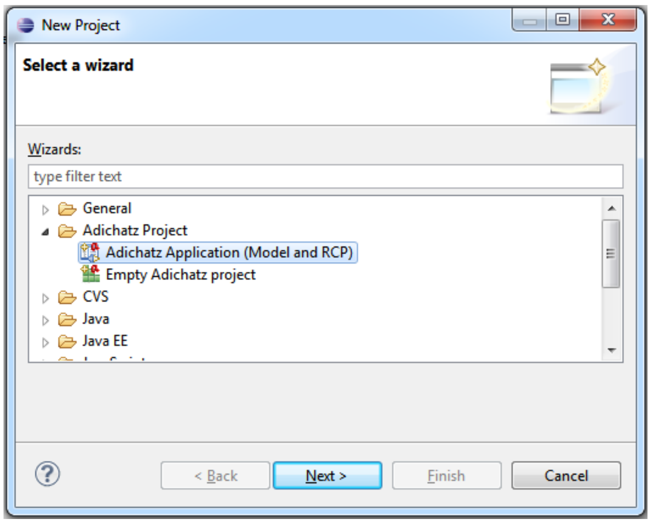
2.3.1 Creation of the project

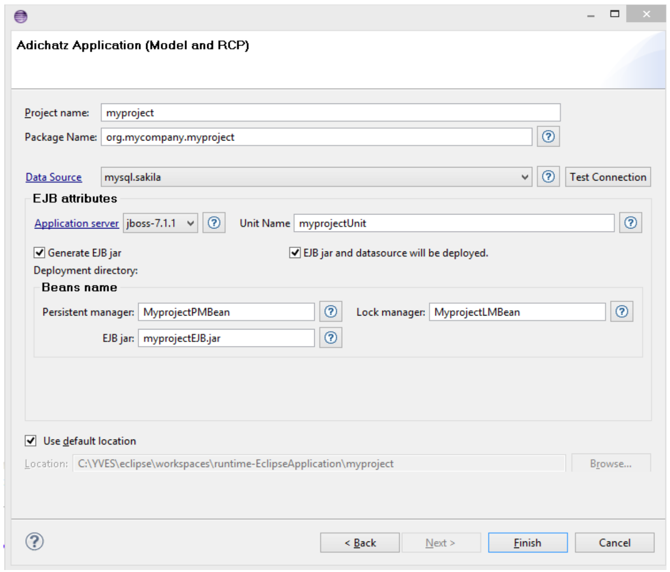
In the Package Explorer right-click and select:

- new / Project…
- Adichatz Application (Model and RCP).

This option provides the quickest way to generate an Application wich contains Model and UI aspects.

You can choose to create one or several Empty projects and decide to split in a Model project and a RCP project (UI).



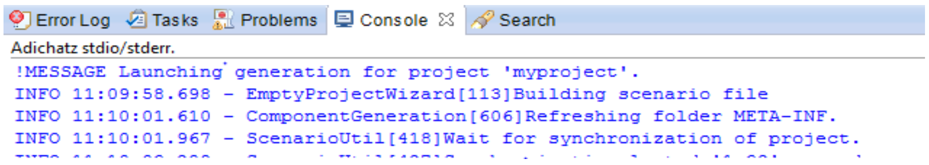

Adichatz Application (Model and RCP)

Project name: myproject
Package Name: org.mycompany.myproject
Data Source mysql.sakila
EJB attributes
Application server jboss-7.1.1
Unit Name myprojectUnit
☑ Generate EJB jar
☑ EJB jar and datasource will be deployed.
Deployment directory:
Beans name
Persistent manager: MyprojectPMBean
Lock manager: MyprojectLMBean
EJB jar: myprojectEJB.jar
☑ Use default location
Location: C:\YVES\eclipse\workspaces\runtime-EclipseApplication\myproject

- Type the name of the project
- Datasource parameters can be changed by clicking Data Source hyperlink.
- Test the connection.
- Choose a connector between (*jboss-7.1 is recommended*):
  - jse
  - jboss-4.2.3
  - jboss-5.0.0
  - Jboss EAP 6.
  - **jboss-7.1**
- then click Finish.
- Persistent manager and lock manager are default name for bean components of the EJB jar. JNDI name and datasource file name are needed. These four properties must be unique on the application server which can host several Adichatz projects

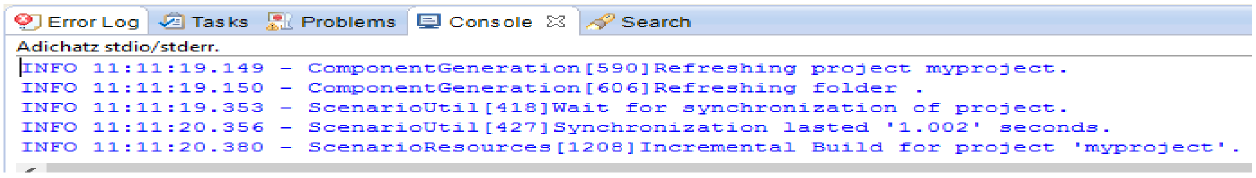
You can change datasource specifications in clicking Data Source hyperlink field.
You can change application server specifications in clicking Application server hyperlink field.
*Specifications are stored in file **{eclipse.dir}\plugins\org.adichatz.template\template\connectors\connector.xml**.*

In the console view, you can see the unfolding of the creation of the project. Logs start like this:



```
Adichatz stdio/stderr.
!MESSAGE Launching generation for project 'myproject'.
INFO 11:09:58.698 - EmptyProjectWizard[113]Building scenario file
INFO 11:10:01.610 - ComponentGeneration[606]Refreshing folder META-INF.
INFO 11:10:01.967 - ScenarioUtil[418]Wait for synchronization of project.
```

and finish like this:



```
Adichatz stdio/stderr.
INFO 11:11:19.149 - ComponentGeneration[590]Refreshing project myproject.
INFO 11:11:19.150 - ComponentGeneration[606]Refreshing folder .
INFO 11:11:19.353 - ScenarioUtil[418]Wait for synchronization of project.
INFO 11:11:20.356 - ScenarioUtil[427]Synchronization lasted '1.002' seconds.
INFO 11:11:20.380 - ScenarioResources[1208]Incremental Build for project 'myproject'.
```

If you have a look to the Application server logs, you can see that the EJB is deployed (*Application server must obviously be started*):

```
11:10:50,690 INFO  [org.hibernate.service.jdbc.connections.internal.ConnectionProviderInitiator] (MSC service thread 1-2) HHH000130: Instantiating
11:10:51,345 INFO  [org.hibernate.dialect.Dialect] (MSC service thread 1-2) HHH000400: Using dialect: org.hibernate.dialect.MySQLDialect
11:10:51,360 INFO  [org.hibernate.engine.transaction.internal.TransactionFactoryInitiator] (MSC service thread 1-2) HHH000268: Transaction strategy:
11:10:51,362 INFO  [org.hibernate.hql.internal.ast.ASTQueryTranslatorFactory] (MSC service thread 1-2) HHH000397: Using ASTQueryTranslatorFactory
11:10:52,741 INFO  [org.jboss.as.server] (DeploymentScanner-threads - 2) JBAS018559: Deployed "myprojectEJB.jar"
```

Your project is created:

- Model POJOs are generated by Hibernate tools
- an EJB is built and deployed to the Jboss Application Server.
- Queries, Table and detail parts, editors, navigator are generated.

This part need less than 2 minutes.

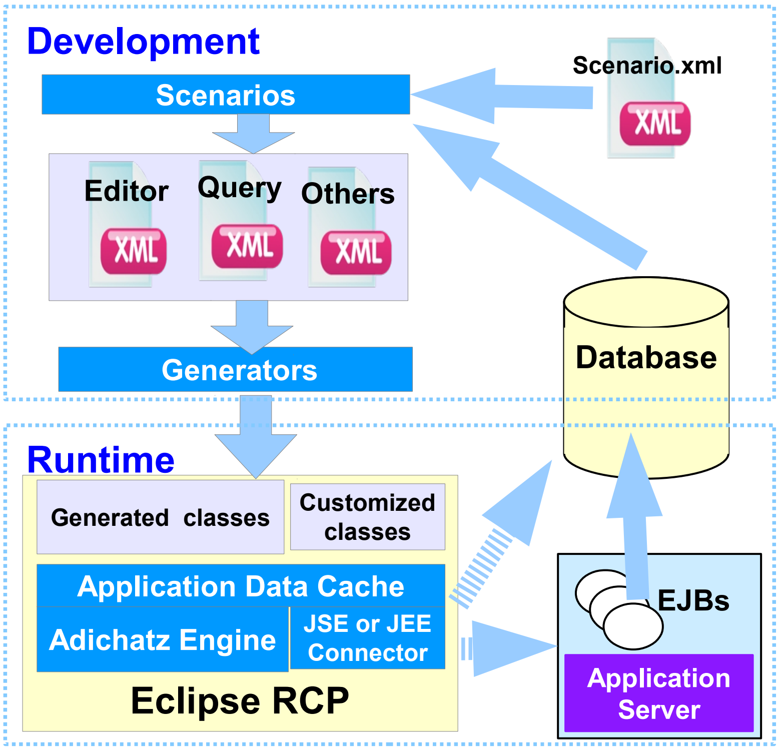
2.3.2 Global Architecture

The project generation is determined by 3 factors:

- Database schema (*POJOs are generated from tables and constraints or copied from existing folder*)
- Scenarios, which can be customized, determine how to generate XML files(*these files describe how adichatz components are assembled*).
- Generators generate Java classes used by the application at runtime from XML files.

Project could be completely customized:

- Change on generated XML files is the easiest way to customize behavior of Adichatz Components.
- All Adichatz Components can be extended.
- Listeners on component Life Cycles provide a way to complete customization.
- Code could be added inside XML file.

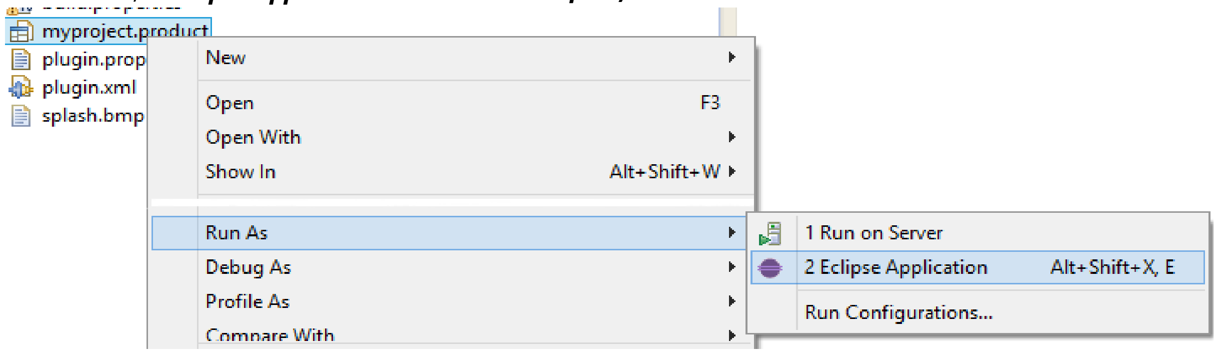


2.4 Running my first Adichatz Application

2.4.1 Run Eclipse Application

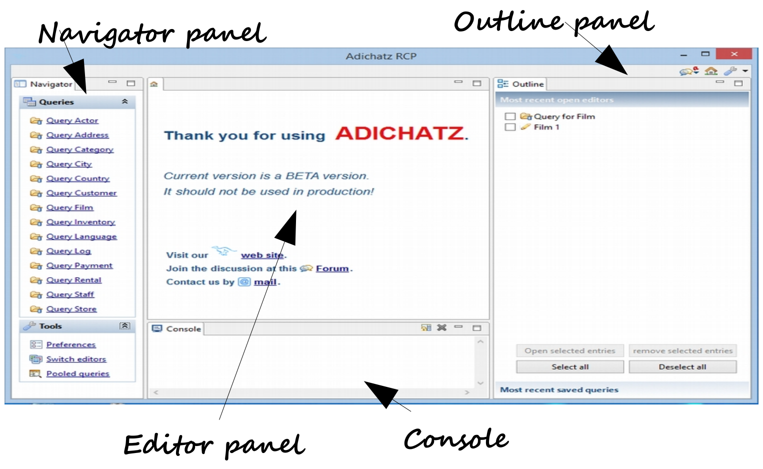
Select the product file (**myproject.product**) in your new project in the package explorer and select option:

- ***Run As / 1 Eclipse Application*** ***Alt+Shift+X, E***



Default Application is divided in 3 vertical panels, the middle is divided in 2 horizontal panels:

- **Navigator panel**: a default navigator with one items by object and tools.
  A menu for specifics tools is added.
- **Editor panel**: Main panel where business data are managed (here, the intro editor).
- **Outline panel**: This panel shows additionnal data depending on active panel (here, recent open editors panel linked to the intro editor).
- **Console panel**: info, warning, error are displayed in this panel.



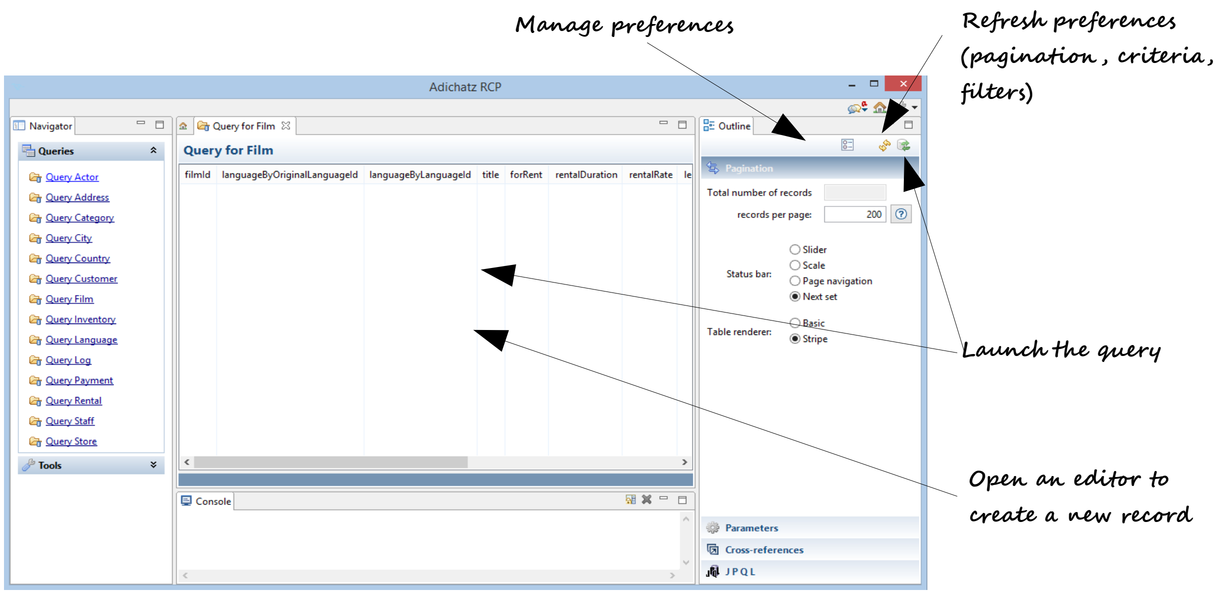
Thank you for using ADICHATZ.

Current version is a BETA version.
It should not be used in production!

Visit our web site.
Join the discussion at this Forum.
Contact us by mail.

2.4.2 Start a query editor

For example, if you are using the imported schema provided here, you can choose the option Query Film, a new editor is opened, it is named Query for Film with only one page named Query (see the bottom of the editor). Right-click on the table to obtain the context menu.



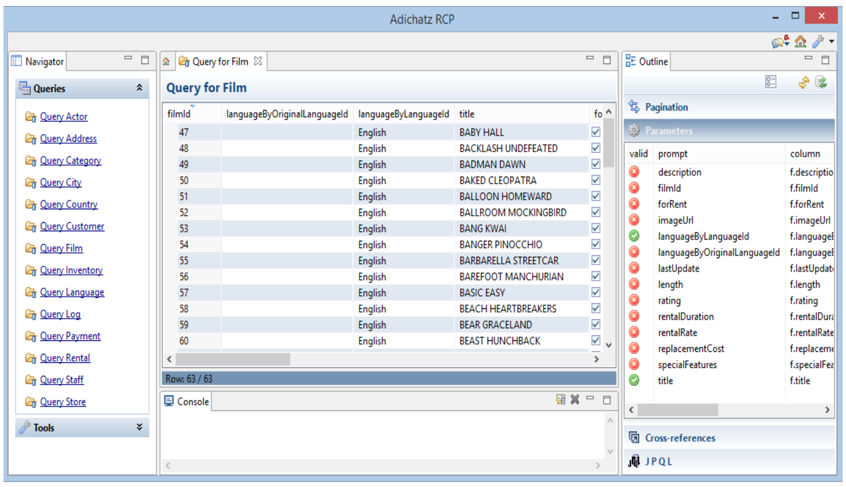
Total number of records:
records per page: 200
Status bar: ( ) Slider ( ) Scale ( ) Page navigation [✓ Next set]
Table renderer: ( ) Basic [✓ Stripe]

Launching the query, the 200 first films are displayed. 200 is the default maximum number of rows for a query. The value is stored in file 'resources/xml/AdichatzRcpConfig.xml' and could be changed.

Pagination and criteria could be changed thru the right panel.

2.4.3 Manipuling Query editor

Query editor provides an Outline view to manage queries.



Query for Film

filmId	languageByOriginalLanguageId	languageByLanguageId	title	fo
47		English	BABY HALL	✓
48		English	BACKLASH UNDEFEATED	✓
49		English	BADMAN DAWN	✓
50		English	BAKED CLEOPATRA	✓
51		English	BALLOON HOMEWARD	✓
52		English	BALLROOM MOCKINGBIRD	✓
53		English	BANG KWAI	✓
54		English	BANGER PINOCCHIO	✓
55		English	BARBARELLA STREETCAR	✓
56		English	BAREFOOT MANCHURIAN	✓
57		English	BASIC EASY	✓
58		English	BEACH HEARTBREAKERS	✓
59		English	BEAR GRACELAND	✓
60		English	BEAST HUNCHBACK	✓

Row: 63 / 63

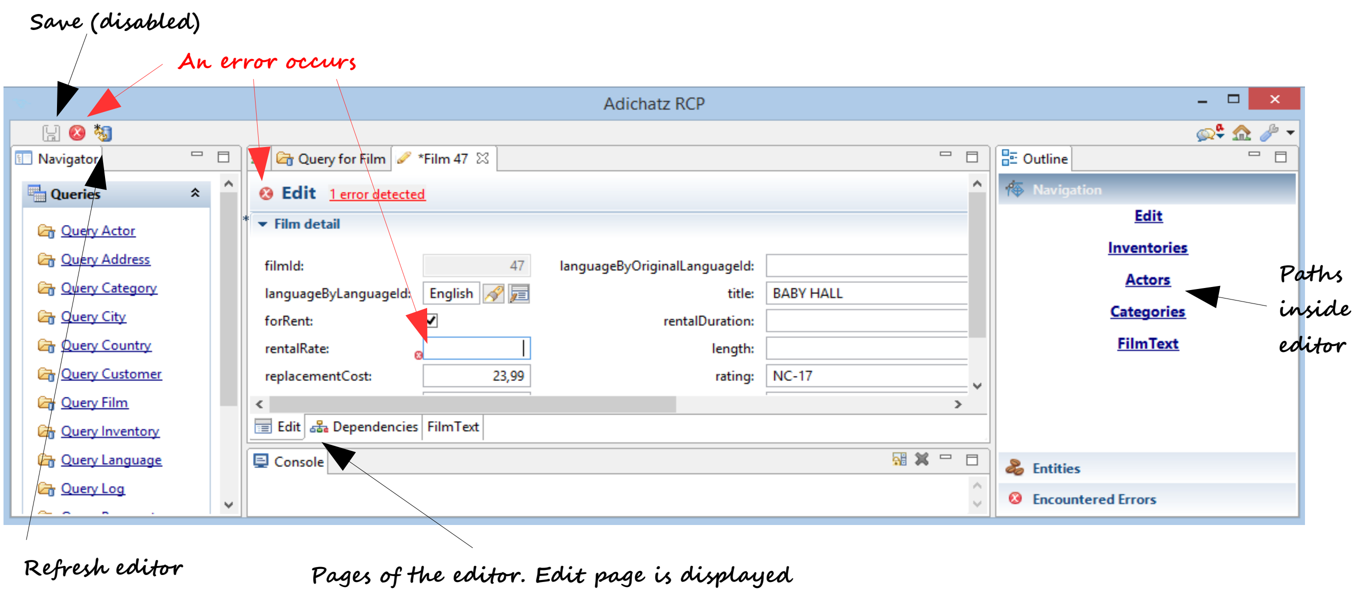
valid	prompt	column
✗	description	f.descriptio
✗	filmId	f.filmId
✗	forRent	f.forRent
✗	imageUrl	f.imageUrl
✓	languageByLanguageId	f.languageI
✗	languageByOriginalLanguageId	f.languageI
✗	lastUpdate	f.lastUpdat
✗	length	f.length
✗	rating	f.rating
✗	rentalDuration	f.rentalDura
✗	rentalRate	f.rentalRate
✗	replacementCost	f.replaceme
✗	specialFeatures	f.specialFea
✓	title	f.title

- Define pagination.
- Define criteria, e.g. as here:
  - Title start by "B"
  - Language is english
- Set filters
- Visualize JPQL order.

These above data determine query preferences which can be stored and restored.

2.4.4 Start a Detail Editor

Double-click on the first row (filmId=47), you obtain a new interesting editor with three pages named Edit, Dependencies and FilmText. Three tabs at the bottom/left of the editor allow you to switch from one page to another. Clear the field "**Title**" then an error occurs.

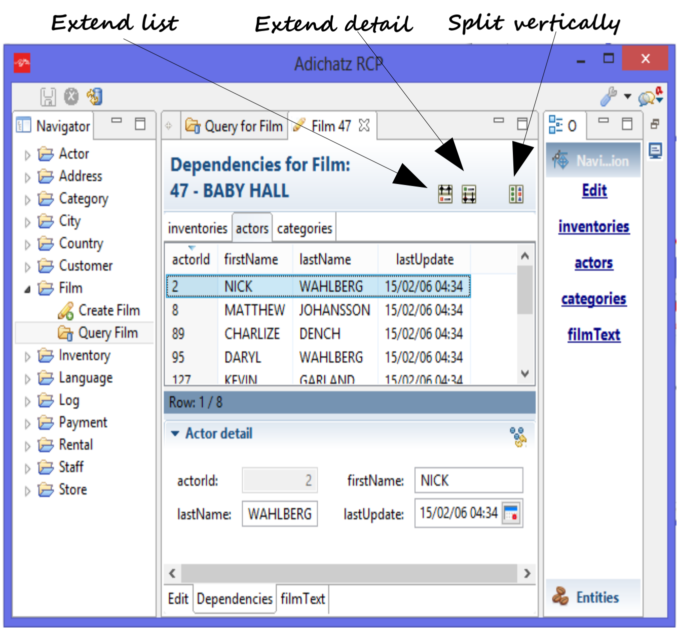


Edit 1 error detected

Film detail

filmId:	47	languageByOriginalLanguageId:	
languageByLanguageId:	English	title:	BABY HALL
forRent:	✓	rentalDuration:	
rentalRate:		length:	
replacementCost:	23,99	rating:	NC-17

- Click on path actors in the outline panel (page Dependencies is open, tab item actors is selected).
- Select actor “2, NICK WAHLBERG” in the list: Detail of the actor is displayed.



Dependencies for Film:
47 - BABY HALL

inventories | actors | categories

actorId	firstName	lastName	lastUpdate
2	NICK	WAHLBERG	15/02/06 04:34
8	MATTHEW	JOHANSSON	15/02/06 04:34
89	CHARLIZE	DENCH	15/02/06 04:34
95	DARYL	WAHLBERG	15/02/06 04:34
127	KEVIN	GARLAND	15/02/06 04:34

Row: 1 / 8

Actor detail

actorId: 2
firstName: NICK
lastName: WAHLBERG
lastUpdate: 15/02/06 04:34

2.4.5 Manipulating Detail Editor

Editors provide the abilities to manage standard operations on database objects.

- Create, retrieve, update and delete (CRUD) objects.
- Add and remove relationships between objects.
- Performs validators before saving changes.
- Performs Adichatz listeners on events on life cycle of controllers or entities.

By default, only mandatory field validators are created when needed. Name for film and first and last name for actor are mandatory. If you erase one of these fields, an error message will appear on the page. Because editor is composed of several pages and several objects could be changed on one page, the error button give the ability to retrieve all errors encountered by validators on the current editor.

2.4.6 Entity and Application data cache

Entity

For Adichatz, an entity is a composition, wrapping the bean (or POJO) received from the JPA framework. Beans compose input of queries but as soon as the editor needs the bean to be managed, it is transformed in an entity.

Application data cache

Moreover, an Application Data Cache, storing entities, supplies central features:

- **Unicity** of entity: An entity is identified by a Primary Key. When looking for an entity, Adichatz asks the cache before requesting the Application Server or the database.
- **Consistency**: an entity could be updated only in one editor at a moment.
- **Databinding:** any change on a field is reflected throughout the application, including process on persistent sets (ManyToMany and OneToMany relationships).
- **Listeners** allows you to configure buttons, actions and other controls according to the status of the entity (Created, Updated, Deleted, Read).
- Adichatz provides in standard a plain solution for managing **locks** in JEE context. An entity could be updated by only one session.
- By default, every relationship are “**lazy loaded**”. However, a «decision tree» of all needed lazy fetches of the entity is managed to optimize access to the server.
- Some query results could be pooled.

2.4.7 Databinding service

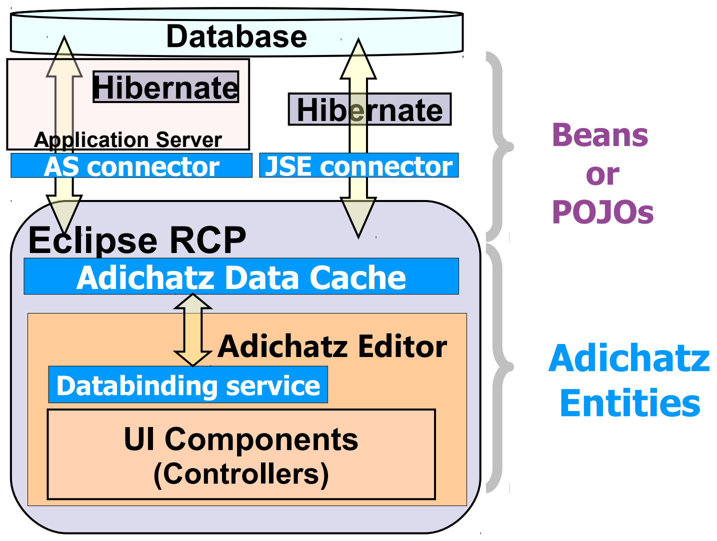
Adichatz databinding service is the environment where data are manipulated. By default, one databinding service is created when opening an editor. This service ensures data consistency of the entities in connection with the application cache and editor:

- Change are broadcast to other editors (*including changes on relationship between objects*).
- Dirty management (*asterisk* *) of editor, section and table is provided.
- Lazy loading: If you decide to display for example the name of the country of the city you do not have to program any query. The service manages all that is necessary.

*The framework provides a default databinding service for JPA environment (data coming from a SQL database). Internally, Adichatz uses an other implementation of databinding to manage XML files using all Adichatz components (see **axml** editor and Scenario editor). In the future, we hope others implementations to be provided.*

2.4.8 Running Architecture

Beans are provided by Hibernate, wrapped in Data Cache to become an Entity capable of managing Lazy loading, Locks, unicity thru Application data cache and Editor databinding service.



2.4.9 Conclusion

In a few minutes, you can obtain editors to query the database and to edit objects of the database. For each object, you can manage CRUD (Create, Read, Update, Delete) operations and also manage dependencies (ManyToOne, OneToMany or ManyToMany relationships).

3 *Manipulating generated components*

3.1 Generated Files

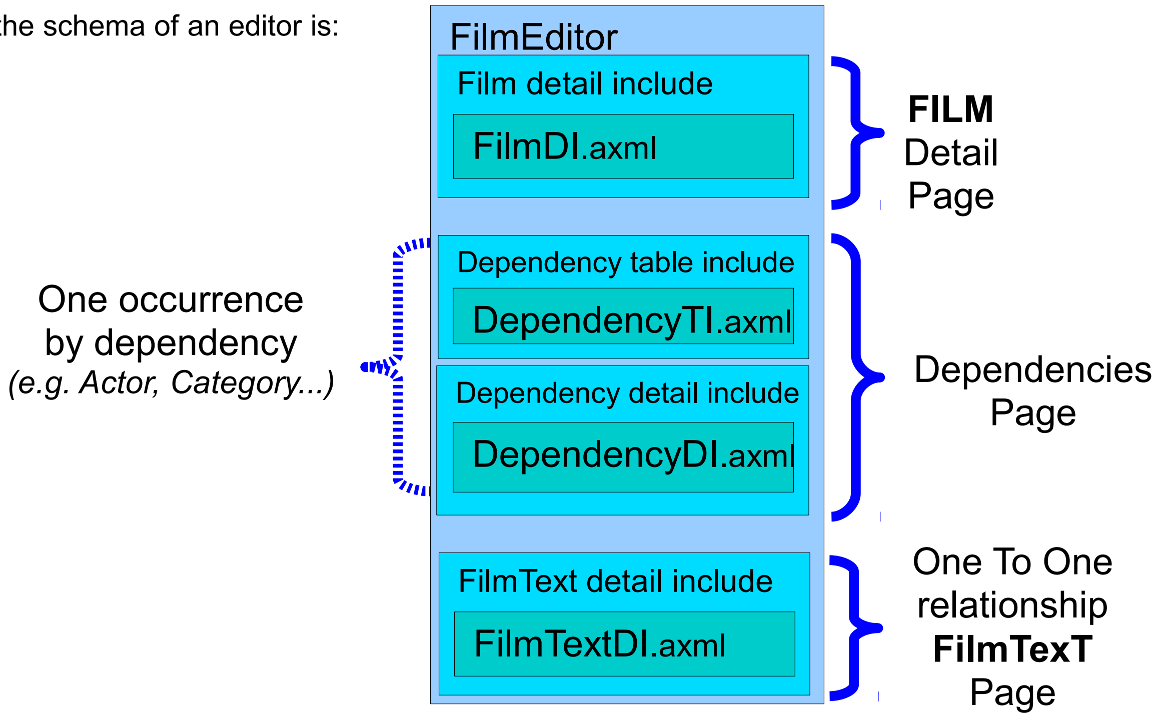
Adichatz generates XML files with **axml** extension from which UI Java classes are generated.

The complete syntax of XML files will be presented later.

The main XML file for the editor is named «Object»Editor, in our case FilmEditor (complete path is $projectDirectory/resources/xml/editor/FilmEditorGENERATED.axml).

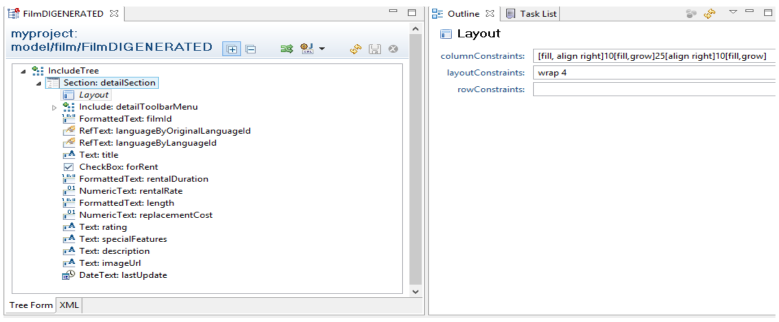
An **axml** file can call others **axml** files by the use of the clause <include>.

So the schema of an editor is:



3.2 Customizing FilmDI

In the Package Explorer, open resource "resources/xml/model/film/FilmDIGENERATED.axml".


FilmDIGENERATED

myproject: model/film/FilmDIGENERATED

```
IncludeTree
  Section: detailSection
    Layout
    Include: detailToolbarMenu
    FormattedText: filmId
    RefText: languageByOriginalLanguageId
    RefText: languageByLanguageId
    Text: title
    CheckBox: forRent
    FormattedText: rentalDuration
    NumericText: rentalRate
    FormattedText: length
    NumericText: replacementCost
    Text: rating
    Text: specialFeatures
    Text: description
    Text: imageUrl
    DateText: lastUpdate
```

Tree Form | XML

Outline | Task List

Layout

columnConstraints: [fill, align right]10[fill,grow]25[align right]10[fill,grow]
layoutConstraints: wrap 4
rowConstraints:

You obtain an editor which contains 2 pages:

- FormPage: Propose a tree and an outline view to select and edit element of the **axml** file.
- XML: Propose to edit xml source.

We want to set different changes on the layout of the detail of a film:

- the description field must fill 3 cells.
- the imageURL field makes a reference to an image which much take half of the width of the editor.
- The other half must contains following fields.
- Display of 3 fields depends on forRent value (dynamic block).

Select XML page and change elements:

```
<formattedText editPattern="######" format="Short" property="filmId" enabled="false" id="filmId"/>
<refText property="languageByOriginalLanguageId" style="SWT.BORDER | AdiSWT.FIND_BUTTON | AdiSWT.DELETE_BUTTON" id="languageByOriginalLanguageId">
    <convertModelToTarget>return null==value ? &quot;&quot; : #FV().name;</convertModelToTarget>
 </refText>
```

```
<refText property="languageByLanguageId" mandatory="true" style="SWT.BORDER | AdiSWT.FIND_BUTTON" id="languageByLanguageId">
    <convertModelToTarget>return null==value ? &quot;&quot; : #FV().name;</convertModelToTarget>
</refText>
<text textLimit="255" property="title" mandatory="true" id="title"/>
<formattedText editPattern="######" format="Short" property="rentalDuration" mandatory="true" id="rentalDuration"/>
<numericText pattern="##.##" property="rentalRate" mandatory="true" style="SWT.BORDER | SWT.RIGHT" id="rentalRate"/>
<formattedText editPattern="######" format="Short" property="length" id="length"/>
<numericText pattern="###.##" property="replacementCost" mandatory="true" style="SWT.BORDER | SWT.RIGHT" id="replacementCost"/>
<text textLimit="5" property="rating" id="rating"/>
<text textLimit="54" property="specialFeatures" id="specialFeatures"/>
<dateText property="lastUpdate" mandatory="true" style="SWT.BORDER | SWT.TIME" id="lastUpdate"/>
<text textLimit="255" property="description" id="description"/>
<text textLimit="255" property="imageUrl" id="imageUrl"/>
```

by

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<includeTree xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" entityURI="adi://myproject/model.film/FilmMM" generationType="DETAIL"
            xsi:noNamespaceSchemaLocation="http://viatz.org/xsd/v0.8.8/generator/includeTree.xsd">
  <section text="#MSG(film, detailSectionText)" style="Section.TWISTIE | Section.TITLE_BAR | Section.EXPANDED" id="detailSection">
    <layout layoutConstraints="wrap 4" columnConstraints="[fill, align right]10[fill,grow]25[align right]10[fill,grow]"/>
    <include adiResourceURI="#PARAM(TOOL_BAR)" id="detailToolbarMenu">
      <params>
        <param id="CONTROLLER" value="#CONTROLLER(detailSection)"/>
      </params>
    </include>
    <formattedText editPattern="######" format="Short" property="filmId" enabled="false" id="filmId"/>
    <text textLimit="255" property="title" mandatory="true" id="title"/>
    <text textLimit="255" property="description" layoutData="span 3" id="description"/>
    <composite layoutData="newline, grow, push, span 4" id="bottomComposite">
      <layout layoutConstraints="wrap 2" columnConstraints="[fill, grow][fill, grow]" rowConstraints="[fill, grow, al top]"/>
      <imageViewer toolBarStyle="AdiSWT.DELETE_BUTTON  | AdiSWT.EXPANDABLE | AdiSWT.EDITABLE" fitCanvas="true" imageType="Url"
              property="imageUrl" noLabel="true" layoutData="hmax 300" id="imageUrl"/>
      <composite id="fieldComposite">
        <layout layoutConstraints="wrap 2" columnConstraints="[align right]10[fill,grow]"/>
        <refText property="languageByOriginalLanguageId" style="SWT.BORDER | AdiSWT.FIND_BUTTON | AdiSWT.DELETE_BUTTON" id="languageByOriginalLanguageId">
          <convertModelToTarget>return null==value ? &quot;&quot; : #FV().name;</convertModelToTarget>
        </refText>
        <refText property="languageByLanguageId" mandatory="true" style="SWT.BORDER | AdiSWT.FIND_BUTTON" id="languageByLanguageId">
          <convertModelToTarget>return null==value ? &quot;&quot; : #FV().name;</convertModelToTarget>
        </refText>
        <combo values="G, PG, PG-13, R, NC-17" property="rating" id="rating"/>
        <checkBox text="#MSG(film,forRent)" property="forRent" noLabel="true" style="SWT.CHECK" id="forRent"/>
        <composite id="forRentComposite">
          <layout layoutConstraints="wrap 1, ins 0, hidemode 3" columnConstraints="[fill, grow]"/>
          <pgroup text="#MSG(Film,forRent)" id="forRentGroup">
              <dynamicClause listenedFieldId="forRent" listenedContainerId="detailSection">
                       <conditionCode>null!= #BEAN().getForRent() &amp;&amp; #BEAN().getForRent()</conditionCode>
                       <postCode>#CONTROLLER(fieldComposite).getComposite().layout();
#CONTROLLER(detailSection).reflow();</postCode>
              </dynamicClause>
              <layout layoutConstraints="wrap 2" columnConstraints="[align right]10[fill,grow]"/>
              <formattedText editPattern="######" format="Short" property="rentalDuration" mandatory="true" id="rentalDuration"/>
              <numericText pattern="##.##" property="rentalRate" mandatory="true" style="SWT.BORDER | SWT.RIGHT" id="rentalRate"/>
              <numericText pattern="###.##" property="replacementCost" mandatory="true" style="SWT.BORDER | SWT.RIGHT" id="replacementCost"/>
          </pgroup>
        </composite>
        <formattedText editPattern="######" format="Short" property="length" id="length"/>
        <text textLimit="54" property="specialFeatures" id="specialFeatures"/>
        <dateText property="lastUpdate" enabled="false" style="SWT.BORDER | SWT.TIME" id="lastUpdate"/>
      </composite>
    </composite>
  </section>
</includeTree>
```

When saving file, changes are reflected to Tree Form page and java classes are automatically regenerated.
If you close and reopen editor Film 1 , the layout for detail will change like this:

Adichatz uses Miglayout as main Java Layout manager. Miglayout greatly simplifies layout management and concentrates in only one layout definition most of all the features provides in standard by multiple SWT layouts. Syntax is quite fine explained on the website.

3.3 Customization element

We have seen that an XML file could call another XML file which could call an other XML file... For that feature, <include> tag is used.

To be efficient, the call of a « sub XML file» must be accompanied with the possibility to drive the content of the «sub XML file» or the <sub sub XML file»...

In Package Explorer, open resource “resources/xml/model/film/FilmDIGENERATED.axml”. Then click on icon to open the XML editor.

To manage these demands of customization from the parent XML file, use the `<customization>` tag in the `<config>` paragraph.

The syntax is the same as for other **axml** file with the exception of the attribute `id`.

```
<customizations>
    <tableColumn valid="false" id="actorsTSI:filmsTC"/>
    <table id="actorsTSI:table">
       <listeners>
          <listener type="onDoubleClick" id="onDoubleClick">
             <code>import org.adichatz.engine.action.EditEntityAction
EditEntityAction.openForm(getFromRegister(&quot;actorsTSI:table&quot;));
             </code>
          </listener>
       </listeners>
    </table>
</customizations>
```

The expression ***id="actorsTSI:filmsTC"*** means the controller identified by ***filmsTC*** of the include ***actorsTSI*** (here a table column controller).

The expression ***id="actorsTSI:table"*** means the controller identified by ***table*** of the include ***actorsTSI***(here a table column controller).

So the tableColumn for the property “film” will no be added to the table of actors and when doubleclicking on a row of this table, a new editor opens.

If you want to disable the action “Add Link” of the context menu of the table controller for Actors, use the following syntax:

```
<action id="actorTSI:tableContextMenu:addRelationshipAction" enabled="false"/>
```

3.4 Message Resource Bundles and images.

3.4.1 Message Resource Bundle

Message Resource Bundle is a way for internationalizing wordings.

To access to the property actors of the message resource bundle file $project/resources/bunldes/film.properties, use the syntax ***`#MSG(film, actors)`***.

For the property `date` of the message resource bundle file $project/resources/bunldes/adichatzEngine.properties of plugin org.adichatz.engine, use the syntax ***`#MSG(adi://org.adichatz.engine/./adichatzEngine, date)`***.

3.4.2 Images

To access to the image actor.png in the folder $project/resources/icons, use the syntax ***`#IMG(actor.png)`***.

Image `IMG_JQL.png` in the folder $project/resources/icons, of the plugin org.adichatz.jpa, could be accessed using use the syntax ***`#IMG(adi:/org.adichatz.jpa/./IMG_JQL.png)`***.

To access to image descriptor use the syntax ***`#IMGDESC(...)`***.

3.5 Validation Process

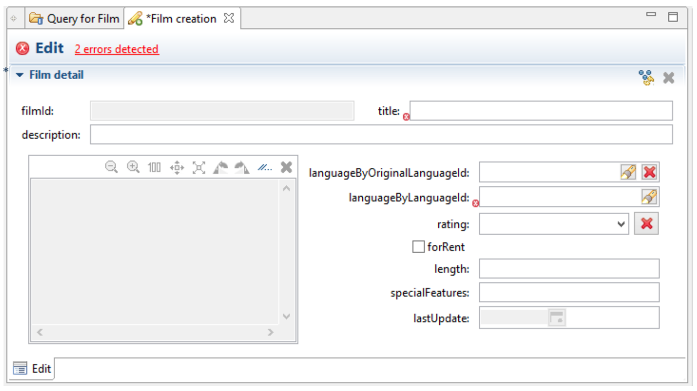
3.5.1 Principles

The validation process is totally integrated to the editor.

- ➢ You cannot save an editor when a validation error is encountered..
- ➢ When displaying, modifying, saving data, the validation process is triggered.

To illustrate this issue, choose in the menu the option `Film` / `Create Film`:

You can see that you obtain five errors corresponding to five mandatory fields. Each time you give a value in a field, an error is removed.



In fact, attributes `mandatory="true"` of fields `title` and `languageByLanguageId`, … are transformed as validators during the generation process of UI classes.

So, 2 errors occur when asking for a new `Film` record and you must fulfill these fields.

3.5.2 Validators

The validation process consists to trigger validators following on three steps:

- ➢ When field value is changing due to user action or databinding propagation.
- ➢ When data are refreshed.
- ➢ When a new entity is injected.

3.5.3 Validator Types

Validators is a way to control data when there are displayed or seized.

There are two kinds of Validator:

- ➢ XML written Validators. All the code needed for the validator is written in the XML file.
- ➢ Validators calling a validator class: Create a Java class which extends class org.adichatz.engine.validation.AValidator and reference it in the XML file.

3.5.4 Creating a validator in XML File

In editor `Film` we opened (see #3.5.Customizing FilmDI), change element:

```
<text layoutData="span 3" property="description" id="description"/>
```

by:

```
<text layoutData="span 3" property="description" id="description">
   <validators>
      <validator key="titleLength" warningMessage="Description generally has more than 10 characters."
            errorMessage="Description should not be null">
         <warningWhen>return ((String) getValue()).length() &lt; 11;</warningWhen>
         <errorWhen>return null == getValue() || 0 == ((String) getValue()).length();</errorWhen>
      </validator>
   </validators>
</text>
```

You have created a new Validator that:

- triggers an error if the description field contents only blank characters or is a null string.
- triggers a warning (not blocking) when description text content less than 11 characters.

3.5.5 Using a pre-defined validator class

Some validator could be complex or could be used in several xml files. For that, a better way is to use a predefined validator class. Syntax is:

```
<text layoutData="span 3" property="description" id="description">
    <validators>
            <validator key="titleLength"
                    warningWhen="return ((String) getValue()).length() &lt; 11;"
                    warningMessage="Description generally has more than 10 characters."
                    errorWhen="return null == getValue() || 0 == ((String) getValue()).length();"
                    errorMessage="Description should not be null"/>
        <validator key="alphaNumeric" validatorClassName="org.mycompany.myproject.gencode.custom.TitleValidator"/>
   </validators>
</text>
```

The **TitleValidator** class must extends the class org.adichatz.engine.validation.AValidator.

If you create the class in a subpackage of org.adichatz.gencode in source folder resources/gencode/src, you will not have to close and relaunch the application because these classes are dynamically loaded. To create the class, use the Eclipse IDE from which you launched the application.

Code could be like this:

```java
package org.mycompany.myproject.gencode.custom;
import java.util.regex.Pattern;
import org.adichatz.engine.controller.IValidableController;
import org.adichatz.engine.controller.field.TextController;
import org.adichatz.engine.validation.AValidator;
import org.eclipse.jface.dialogs.IMessageProvider;
public class TitleValidator extends AValidator {
	public TitleValidator(IValidableController triggeringController, String key) {
		super(triggeringController, key);
	}
	@Override
	public void validate() {
		String title = (String) ((TextController) triggeringController).getValue();
		Pattern pattern = Pattern.compile("(\\w| )+");
		if (null != title && !pattern.matcher(title).matches()) {
			setLevel(IMessageProvider.ERROR, "Only alphanumeric characters are accepted.");
		} else
			setLevel(IMessageProvider.NONE, null);
	}
}
```

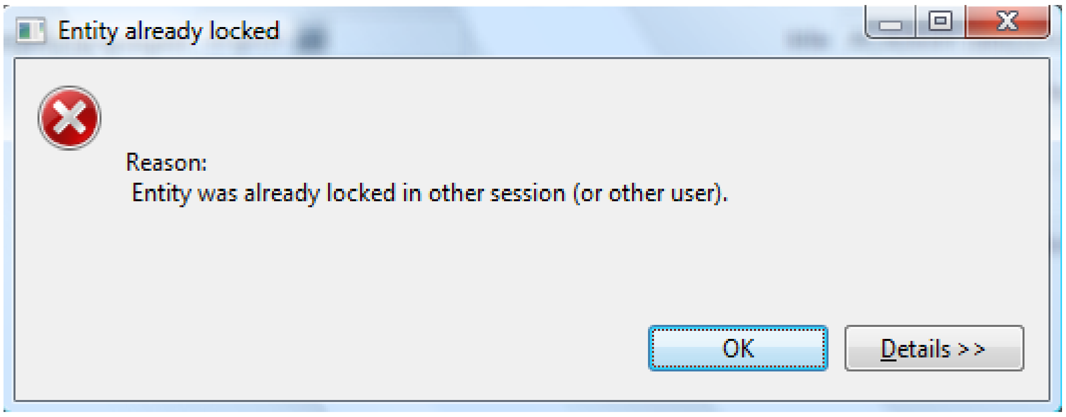
You have just created a validator. The validator triggers an error where the title don't have an alphanumeric value.

4 Other Features

4.1 Lock management in Application server context

Open a new application (see #Run Eclipse Application) and update «Film 1» object.

Go to the first application and try to update «Film 1» object. The following error message occurs and update is cancelled:



Reason:
Entity was already locked in other session (or other user).

4.2 Composite key strategy

Consider that you have an object with a composite key as following:

- Parent key
- Increment

Adichatz proposes to manage directly the incrementation of this kind of key.

4.3 Security

Adichatz provides a simple way to manage authorizations, roles and privileges.

4.4 Callbacks

You can easily add EJB callback to complete process on the server-side.

You can add kind of callback on client side called callfore.

5 Component Architecture

5.1 Controller layer

Adichatz defines a new layer composed of controllers.

A controller is a composition of Rich UI components, as a ManagedForm or simple SWT components as Text.

The controller manages links between UI components and entities to ensure the connection with locks, dirty and lazy fetches management.

5.2 Controller features

Each controller has a parent controller except the Editor controller.

Each collection controller contents a set of child controllers.

With that you can define an «tree of controllers».

5.3 Collection controllers

A collection controller is a controller which can host other controllers.

There are two types of collection controller:

5.3.1 Controllers hosting entity

That means that they can host their own entity which can be different from the entity hosted by the parent controller. This allows an editor to manage several entities:

ArgPShelfController	Composition around a PShelf (see nebula project). Children should be PShelftemControllers and are defined by a List of values (like a combo) and so is not determined like for PShelfController.
ArgTabFolderController	Composition around a CTabFolder. Children should be CtabItemControllers and are defined by a List of values (like a combo) and so is not determined like for CtabFolderController.
CTabItemController	Composition around a CTabItem. Children available are the same as those of SectionController.
CompositeController	Composition around a Composite. Children available are the same as those of SectionController.
CompositeBagController	Composition around a specific control called CompositeBag: A bag of composites. Only one composite is displayed.
FormPageController	Composition around a ManagedForm for simulating a page of an editor.
GroupController	Composition around a Group. Children available are the same as those of SectionController.
IncludeController	Specific controller to manage include process (see **axml** Files)
PGroupController	Composition around a PGroup (nebula project).
PShelfItemController	Composition around a PShelfItem (nebula project).
SashFormController	Composition around a SashForm.
ScrolledCompositeController	Composition around a SharedScrolledComposite.
ScrolledFormController	Composition around a ScrolledForm (Assumes scrolling and message zone).
ScrolledPGroupController	Composition around a PGroup (nebula project). Content composite is a scrolled composite.
SectionController	Composition around a Section.

To know the exhaustive and up to date list of child controller types, consult the xsd files.

5.3.2 Other collection controllers

ButtonBarController	Composition around a bar of buttonControllers and separators. You can specify an orientation.
CTabFolderController	Composition around a CTabFolder. Children should be CTabItemControllers.
GridController	Composition around nebula components Grid and GridViewer (nebula project).
ManagedToolBarController	Specific ToolBar used in section. Children should be ActionControllers.
PShelfController	Composition around nebula component PShelf. Children should be PShelfItemControllers.
TableController	Composition around a Table and a TableViewer.
ToolBarController	Composition around a ToolBar. Children should be ToolItemControllers.
TreeController	Composition around a Tree and a TreeViewer.

5.4 Field controllers

Field controllers are controllers relating to fields of the bean.

CComboController	Composition of a CCombo. Style AdiSWT.FIELD_ASSIST was added when Field Assist is demanded.
CheckBoxController	Composition of a Button with style SWT.CHECK. Corresponding type in the database is boolean.
ComboController	Composition of a Combo. Style AdiSWT.FIELD_ASSIST was added when Field Assist is demanded.
DateTextController	Composition of a DateText. DateText is an UI Adichatz component which allows to manage date in several format (Date, DateTime...).
ExtraTextController	Composition of a ExtraText. An extra text is a special widget that allows you to manage the bold, italic, underline or URLs in the text.
FileController	Composition of a FileText: a specific control which manage file path.
FormTextController	Composition of a FormText. An easy way to manage URL tag and image in your application.
GMapController	Composition of a GMap control. GMap is a composite control which contains a browser. A http request is send to google and result is displayed as a mapor a satellite image.
HyperlinkController	Composition of a Hyperlink.
ImageViewerController	Receive an URL or file path or data and display an image.
LabelController	Composition of a Label.
NumericTextController	Composition of a NumericText. NumericText is a Adichatz UI component. Corresponding types in the database are all numeric types. Format could be specified.
RadioGroupController	Composition of a list of Buttons with style SWT.RADIO.
RefRadioGroupController	Similar to a referenced RadioGroupController, but list of Buttons is determined by a persistent set or a query.
RefTextController	Composition of a RefText. RefText (referenced text) is an UI Adichatz component which allows to manage Many-To-One relationship. Styles AdiSWT.FIND_BUTTON and AdiSWT.DELETE_BUTTON are added when a find button and a delete button (for clearing value) are demanded.
RichTextController	Composition of a RichText. a rich text is a special widget that allows you to manage the bold, italic, underline or URLs in the text.
TextController	Composition of a Text. Corresponding type in the database is varchar.

5.5 Others controllers

ActionController	Child controller of ManagedToolBarController or component of a ButtonController.

ButtonController	Composition of a Button. It could contain, an Action Controller or define its own action.
FormattedTextController	Composition around a FormattedText (nebula project).
GridColumnController	Child controller of GridController (nebula project) corresponding to a column.
GridColumnGroupController	Child controller of GridController (nebula project) corresponding to a group of columns.
PGroupMenuController	Menu of a PGroup (nebula project).
PGroupToolItemController	Tool Item of a PGroup (nebula project).
RichTextController	Composition around a RichText (nebula project).
TableColumnController	Child controller of TableController, composition of a TableViewerColumn.
ToolItemController	Child controller of ToolItemController, composition of a ToolItem.

6 Extending Adichatz

There is different ways to extend Adichatz features. We will explore three of them.

Suppose that you want to change the foreground color of title field when length of the text is greater than 40.

At least four possibilities are provided by Adichatz:

- Add code thru a listener:
  - directly in the XML FilmDIGENERATED.axml file.
  - Using a listener class to
- Create a Green40TextController which extends TextController.
- Create a new Key word Red40Text in the XML syntax.

6.1 Listeners

Listeners provides a very effective and simple way to add behavior to your application.

Mostly two categories of listeners are used: Life cycle listeners and control listeners,

Life cycle listeners: fired at specific times of the life cycle of the controller.

AFTER_INITIALIZE	Just after the instantiation of the controller.
BEFORE_CREATE_CONTROL	Before control is created (see method AComponentController#createControl()).
AFTER_CREATE_CONTROL	After control is created.
BEFORE_SYNCRHONIZE	Before entity is bound to the controller.
AFTER_SYNCRHONIZE	After all controls of the generated UI class and all controls of children controllers have been created.
BEFORE_END_LIFE_CYCLE	Before End of life cycle of the controller is launched. (see method org.adichatz.engine.controller.AWidgetController.endLifeCycle()).
AFTER_END_LIFE_CYCLE	After End of life cycle of the controller is launched.
BEFORE_DISPOSE	Before disposing controller.
AFTER_DISPOSE	After entity is bound.

Control listeners: fired when value changes due to user actions

MODIFY_TEXT	Works with all Text controllers and ImageViewerController
WIDGET_SELECTED	Works with controllers having selection action ad Combo, button, group. Controllers.
SELECTION_CHANGED	Works with set controllers (Tree and table controllers).
DOUBLE_CLICK	Works with set controllers (Tree and table controllers).
REFRESH	After data of a set controller (Table, grid or tree) is refreshed
BEFORE_FIELD_CHANGE	When controller hosts a data field *(controller extends AFieldController and controller is bound to data)*, before data is set in the controller.
AFTER_FIELD_CHANGE	After data is set in the controller.

Entity listeners: fired on event of the life cycle of the entity (composition of a POJO bean).

CHANGE_STATUS	Status of the entity change: e.g. RETRIEVE (read) to MERGE (updated)
LOCK_ENTITY	Entity is locked by user action.
BEFORE_PROPERTY_CHANGE	Before a change was sent to a property of the Entity by the databinding process
WHEN_PROPERTY_CHANGE	During a change was sent to a property of the Entity by the databinding process
AFTER_PROPERTY_CHANGE	After a change was sent to a property of the Entity by the databinding process
PRE_SAVE	Before save action was sent to server.
POST_SAVE	After save action was sent to server.
PRE_REFRESH	Before refresh (cancel updates, actualize entity) demand was sent to server.
POST_REFRESH	After refresh (cancel updates, actualize entity) demand was sent to server.
LOAD_ENTITY	After synchronizing entity data in UI components.

Others listeners exists and will be discussed later.

6.1.1 Create a listener adding code in xml file

Listener is a way to add code to generated UI classes by specifying it in the XML file.

Open the XML file FilmDIGENERATED.axml in a XML editor.

Change element:

```
<text layoutData="span 3" property="title" id="title">
```

by:

```
<text layoutData="span 3" property="title" id="title">
   <listeners>
     <listener type="onModifyText">
       <code>import org.eclipse.swt.SWT
if (40 &gt; ((String) getValue()).length())
    getControl().setForeground(#COLOR(SWT.COLOR_RED));
else
    getControl().setForeground(#COLOR(SWT.COLOR_BLUE));
       </code>
     </listener>
   </listeners>
</text>
```

The listener onModifyText will add the code in a ModifyListener added to the text instance included in the TextController.

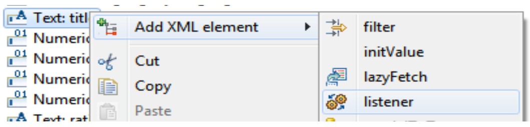
The key word `#COLOR` means the system color red.

Make a test then remove the change you made previously before continuing on next choice.

6.1.2 Create a listener using a class

Instead of add the code in the XML file, you can instantiate a new class listener.

Open file “resources/xml/model/film/FilmDIGENERATED.axml” using adichatz editor,



- ➢ Right click on element “Text: title”
- ➢ Choose “Add XML element / Listener”
- ➢ In the outline view
  - ➢ Select MODIFY_TEXT as listener type.
  - ➢ Click on hyperlink: listenerClassName.
  - ➢ A “New Java class” widow is open with a predefined package.
- ➢ Type the class name “TitleListener” and valid.

The java class editor is open for TitleListener.java. Add modifyText method:

```
package org.mycompany.myproject.gencode.custom;

import org.adichatz.engine.controller.AWidgetController;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events. ModifyEvent;
import org.eclipse.swt.events. ModifyListener;
import org.eclipse.swt.widgets.Text;


public class TitleListener implements ModifyListener {

	AWidgetController controller;

	public TitleListener(AWidgetController controller) {
		this.controller = controller;
	}

	@Override
	public void modifyText(ModifyEvent e) {
		Text title = (Text) controller.getControl();
		title.setForeground(//
		title.getText().length() > 40 ? //
		title.getDisplay().getSystemColor(SWT.COLOR_RED)
				: title.getDisplay().getSystemColor(SWT.COLOR_BLUE));
	}
}
```

In the runtime application Close an reopen Film 1 editor.

It is important to create this class in directory $projectDirectory/resources/gencode/src, because classes in this directory are dynamically loaded. So you do not have to relaunch the application.

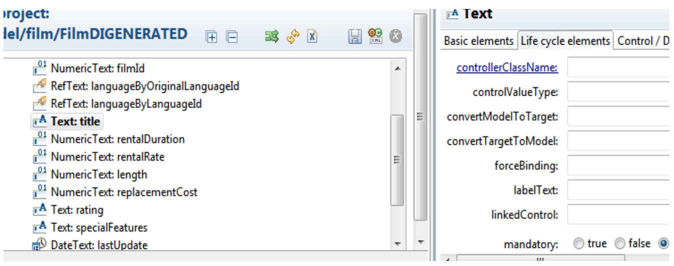
By this way, you can enrich your editor with complex code.

Make a test then, remove the change you made previously (add a combined class) before continuing on next choice.

6.2 Controller extension

By extending class org.adichatz.implementation.controller.field.TextController, you can reach the same result.

Open FilmDIGENERATED.axml file and remove listeners on element "Text: Title" you created in previous chapter. Then in the outline view select "Life cycle elements" tab item.



roject:
lel/film/FilmDIGENERATED

NumericText: filmId
RefText: languageByOriginalLanguageId
RefText: languageByLanguageId
Text: title
NumericText: rentalDuration
NumericText: rentalRate
NumericText: length
NumericText: replacementCost
Text: rating
Text: specialFeatures
DateText: lastUpdate

Text

Basic elements | Life cycle elements | Control / D

controllerClassName:
controlValueType:
convertModelToTarget:
convertTargetToModel:
forceBinding:
labelText:
linkedControl:
mandatory: true false [✓]

- Click on the "Text: Title" element of the tree.
- In the outline view select "Life cycle elements" tab item.
- In the outline view, click on hyperlink field "controllerClassName": a new "New Java class" window is open with a predefined package and super class. Give a name (e.g. Green40TextController) to the class.

The java class editor is open for FilmDICC.java. Add method afterCreateControl:

```
package org.mycompany.myproject.gencode.custom;

import org.adichatz.engine.controller.IContainerController;
import org.adichatz.engine.controller.field.TextController;
import org.adichatz.engine.core.ControllerCore;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.ModifyEvent;
import org.eclipse.swt.events.ModifyListener;
import org.eclipse.swt.graphics.Color;


public class Green40TextController extends TextController {

	public Green40TextController(String id, IContainerController parentController, ControllerCore genCode) {
		super(id, parentController, genCode);
	}

	@Override
	public void createControl() {
		super.createControl();
		getControl().addModifyListener(new ModifyListener() {
			@Override
			public void modifyText(ModifyEvent e) {
				Color color;
				if (getControl().getText().length() > 40)
					color = getControl().getDisplay().getSystemColor(SWT.COLOR_GREEN);
				else
					color = getControl().getDisplay().getSystemColor(SWT.COLOR_BLUE);
				getControl().setForeground(color);
			}
		});
	}
}
```

Close Adichatz RCP application and relaunch it, title field will have a green foreground color when text length is greater than 40.

6.3 Create a new element in the XML syntax

Another possibility would be to create a new element which extend basicText element. That is easily possible but you have to made changes in Adichatz plugins. This issue will be explained later.

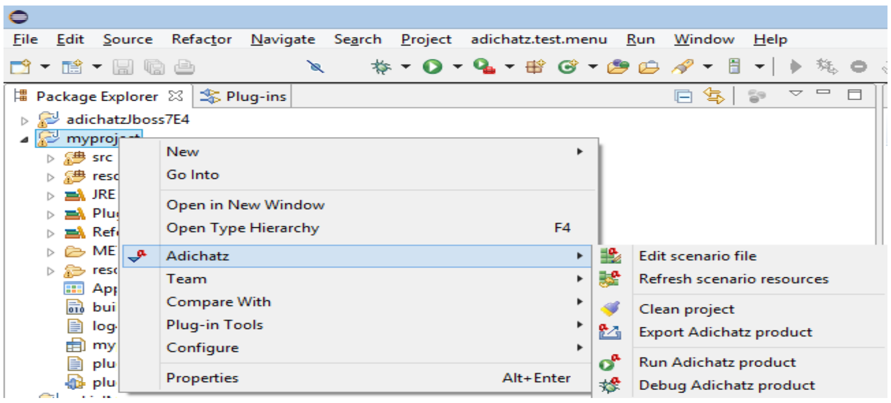
7 Others Features

7.1 Workbench features list

In the Package Explorer, a new option is given when pointing on Adichatz resources, a new options are available in content menu.

7.1.1 Features on Adichatz project:

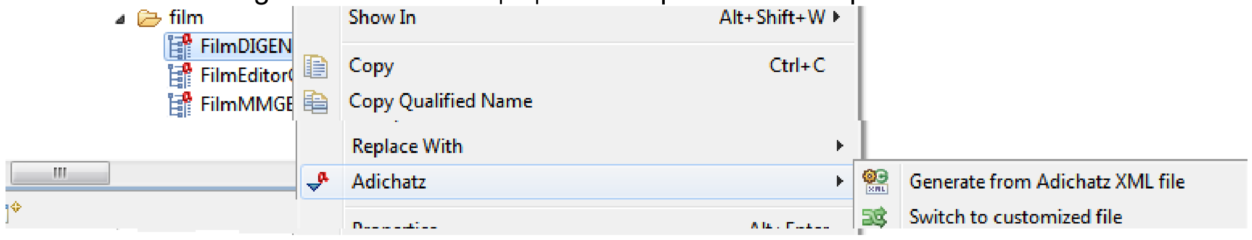
Context menu on an Adichatz project provides this options:



Edit scenario file	Edit the file “resources/xml/scenario.xml” which drive the generation process.
Refresh scenario resources	Refresh project and Scenario resources of the project.
Clean project	Generate needed Java classes, delete obsolete class java and check for error.
Export Adichatz product	Remove org.adichatz.tool and org.adichatz.generator from item Require-Bundle of Manifest.MF file. Export project. Get the project back in running in order to continue developments.
Run Adichatz product	Run Adichatz product.
Debug Adichatz product	Debug Adichatz product.

7.1.2 Generated .axml and .properties files:

Context menu on an generated .axml or .properties file provides this options:



Switch to customized file	Switch generated resources in a new customized file which will not be impacted by new generation process.

7.2 Recursive generation

All generation process can be controlled from “resources/xml/scenario.xml” file.

7.2.1 Contents

One of the main features of Adichatz is the ability to drive generation process at any time.

Scenario.xml file drives the generation process and is composed of 5 sections.

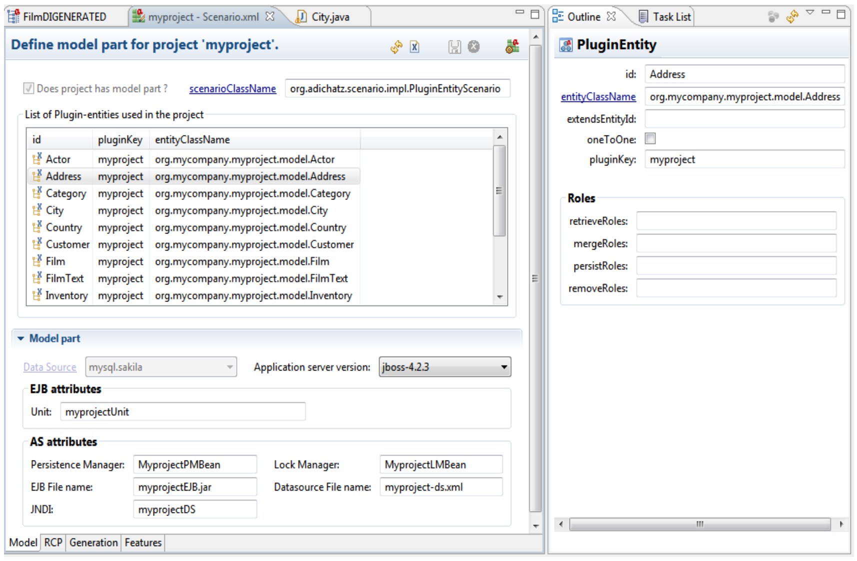
`<params>`	Contains general parameters
`<generatorPaths>`	Paths needed for generation process (plugins, libraries...)
`<generators>`	Mapping between generator and type of XML tree.
`<controllers>`	Mapping between xml elements and Adichatz controllers.
`<generationScenarios>`	In the this main section, the entire generation process is specified: • Do you have a Model Part (EJB) in the project? • Do you have a RCP Part (UI) in the project? • What entities (Pojo) are treated, What Model Plugins hosts these entities (*the same application can handle entities provided by different Model Plugins*).

Have a look on the special editor for file scenario.xml.

One editor containing four pages (*Model, RCP, Generation, Features*) provides a simple way to manage all the generation process.

In changing features, developer can change completely aspects of the generated application.

Note that this editor was built on Adichatz framework. In this case, the datasource is a XML file.



Define model part for project 'myproject'.

☑ Does project has model part ? scenarioClassName: org.adichatz.scenario.impl.PluginEntityScenario

List of Plugin-entities used in the project

id	pluginKey	entityClassName
Actor	myproject	org.mycompany.myproject.model.Actor
Address	myproject	org.mycompany.myproject.model.Address
Category	myproject	org.mycompany.myproject.model.Category
City	myproject	org.mycompany.myproject.model.City
Country	myproject	org.mycompany.myproject.model.Country
Customer	myproject	org.mycompany.myproject.model.Customer
Film	myproject	org.mycompany.myproject.model.Film
FilmText	myproject	org.mycompany.myproject.model.FilmText
Inventory	myproject	org.mycompany.myproject.model.Inventory

Model part

Data Source: mysql.sakila — Application server version: jboss-4.2.3

EJB attributes

Unit: myprojectUnit

AS attributes

Persistence Manager: MyprojectPMBean — Lock Manager: MyprojectLMBean

EJB File name: myprojectEJB.jar — Datasource File name: myproject-ds.xml

JNDI: myprojectDS

Model | RCP | Generation | Features

PluginEntity

id: Address

entityClassName: org.mycompany.myproject.model.Address

extendsEntityId:

oneToOne: ☐

pluginKey: myproject

Roles

retrieveRoles:

mergeRoles:

persistRoles:

removeRoles:

7.2.2 Flexibility

Everyone can create his own generators or controllers or scenarios, maps them in the scenario file.

Moreover, any component proposed by Adichatz, is extensible.

Application can be split in modules following OSGI architecture.

AN example of OSGI example can be seen on youtube.