



Lilya Platform Tutorial

Authors : Mohammed LOUKILI EL FAHSSI & Abdelghani BOUASSOULE
Company : Q-LOGIC
Version : 1.0 Beta 1
Revision : 1
Last Modified : 2007-06-08 10:30:24



Table Of Contents

1. Preface.....	4
2. Introduction to Lilya.....	5
3. Requirements.....	6
4. Compatibility.....	7
5. Running Lilya Demo.....	8
5.1. Requirements.....	9
5.2. Download/Checkout Lilya Demo.....	10
5.3. Load demo data.....	11
5.4. Configure TomCat.....	12
5.5. Run the Demo.....	13
6. Start Using Lilya.....	14
6.1. Using Lilya Web Application Template.....	15
6.2. Create from scratch.....	17
7. Quick Start.....	20
7.1. Creating a Dialog.....	21
7.2. Creating a Toolbar Menu.....	23
7.3. Creating a Flat Menu (XP style).....	26
7.4. Creating a Layout.....	28
7.5. Creating a Form.....	29
7.6. Creating a Dynamic Form.....	30
8. Architecture.....	31
8.1. Overview.....	32
8.2. Ajax Capabilities.....	33
9. Working With Data.....	34
9.1. DataStore/DataModel and Server Implementation.....	35
9.2. Configure.....	36
9.3. Grid Example.....	37
9.4. Form Example.....	38
10. Writing Events.....	39
10.1. Client Events.....	40
10.2. Server Events.....	41

10.3. Decide What to update.....	42
10.4. Understanding the difference between reRendering and Updating.....	43
10.5. Form Example.....	44
11. Upgrading to a new version of ExtJs	45
12. Extending Lilya	
12.1. Adding new Platform Definitions.....	46
12.2. Adding new Platform Editors.....	47
12.3. Creating your own components.....	48



1. Preface

Working with Jsf, ExtJs, Ajax4Jsf and others to create user friendly applications can be cumbersome and time consuming in today's enterprise environments.

Lilya is an Open Source platform for J2EE environments, it uses ExtJs, WZ Grapher widgets and event model.

Lilya DO NOT override or replace ExtJs api, and writing Javascript code still possible.

2. Introduction to Lilya Platform

This chapter is an introductory tutorial for new users of Lilya. This quick start will give you an idea about Lilya's concept.

This tutorial is intended for new users of Lilya but requires Jsf knowledge.

The source code for this tutorial is included in the distribution of Lilya under demo directory.



3. Requirements

Lilya requires :

```
JDK 1.5 or higher  
Jsp/Servlet container like TomCat or Jetty or an application server J2EE 1.4  
certified
```

4. Compatibility

Lilya is tested under JSF 1_1_01 and JSF RI.



5. Running Lilya Demo

To be able to run lilya demo, follow those instructions :

- 5.1.Requirements
- 5.2.Download/Checkout Lilya Demo
- 5.3.Load demo data
- 5.4.Configure TomCat
- 5.5.Run the Demo

5.1. Requirments

In order to run Lilya Demo, you need a small Mysql server, Jdk 1.5.x and TomCat 5.x



5.2. Download/Checkout Lilya Demo

Download Lilya Demo release file `lilya-demo-x.y.z.zip` from <http://sharesource.org/project/lilya> or

Checkout Lilya Demo using svn command line utility or by using an eclipse svn plugin:

```
svn co http://svn.sharesource.org/svn/lilya/lilya-demo [your_lilya_demo_location]
```

5.3. Load demo data

This demo requires a small mysql database.

Run demo scripts located under the 'db' directory:

```
mysql -u yourDbUser -pyourDbPassword
source /[your_lilya_demo_location]/db/demo-script.sql
```

This script will create some US cities to be used in the demo examples (DataGrid, Combos,...).

Link the demo datasource to your database:

Open the file [your_lilya_demo_location]/META-INF/kernel.xml and search/replace urlProvider, user and password with your database configuration.

kernel.xml looks like:

```
<string name="com.qlogic.products.qlor.urlProvider"
        value="jdbc:mysql://localhost:3306/lilya_demo" />
<string name="user" value="yourDbUser" />
<string name="password" value="yourDbPassword" />
```



5.4. Configure TomCat

Configure Tomcat to load the demo application by adding a new context in [TOMCAT_HOME]/conf/server.xml:

```
<Context path="/lilya" reloadable="true"  
        docBase="[your_lilya_demo_location]"  
        workDir="[your_lilya_demo_location]/work"/>
```

5.5. Run the Demo

Start TomCat, and enter `http://localhost:8080/lilya/init.jsf` in your browser.

have fun.



6. Start Using Lilya

To start using Lilya Platform, there is two options:

- 1 - Using Lilya Web Application Template (quick mode)
- 2 - Create from scrach (experienced users)

6.1.Using Lilya Web Application Template

6.2.Create from scrach

6.1. Using Lilya Web Application Template

Create a project directory [your_new_project_dir].

```
mkdir [your_projects_directory]/[your_new_project]
```

Download Lilya application template release file lilya-app-x.y.z.zip from <http://sharesource.org/project/lilya> or Checkout it using svn command line utility or by using an eclipse svn plugin:

```
svn co http://svn.sharesource.org/svn/lilya/lilya-template [your_new_project]
```

This template application contains all necessary jar/tld files (including jsf/j2eeall/lilya apis) and a preconfigured web.xml file

Modify project properties by editing .project and .tomcatPlugin files:

```
<name>lilya-template</name> to <name>[your_new_project]</name>
```

Import your project using Eclipse-->Import.

Register your project in TomCat:

```
<Context path="/[your_new_project]" reloadable="true"
  docBase="[your_projects_directory]/[your_new_project]"
  workDir="[your_projects_directory]/[your_new_project]/work"/>
```

Start TomCat and run

```
http://localhost:8080/your_new_project/init.jsf
```

You will see a page showing a waiting message.

Notes:

- 1 - Lilya Application Template is a blank jsf project containing one jsf page and one Baking Bean and a configured WEB-INF.
- 2 - In order to add new pages you put them in project root or create a new directory to hold your new pages
- 3 - You can delete the existing Baking Bean and create your company java packages.
- 4 - Don't forget to configure your backing beans in WEB-INF/faces-config.xml like you did in JSF
- 5 - Lilya does not



override JSF 6 - Lilya Template Web Application directory structure:

- META-INF (empty)
 - + ui (contains all css, js required by Lilya Jsf components)
 - WEB-INF
 - lib
 - asm-commons.jar
 - asm-util.jar
 - asm.jar
 - cglib-2.2_beta1.jar
 - bsh-2.0b4.jar

Byte Code and bsh shells interpreter
 - jsf-api.jar
 - jsf-impl.jar
 - jstl.jar
 - standard.jar
 - commons-beanutils.jar
 - commons-collections.jar
 - commons-digester.jar

JSF and Sun RI
 - j2ee4all.jar
 - j2ee4all-jsf.jar
 - ajax4jsf.jar
 - oscache-2.3.2.jar
 - lilya.jar
- Lilya and ajax apis
- src (contains dummy Backing bean, create yours)
- tld
 - c.tld
 - jsf_core.tld
 - html_basic.tld

Jsf tld files
- j2ee4all-jsf.tld
- lilya.tld
- J2ee4All Jsf (standard html components like div, pre, h...)
Lilya Platform components
- faces-config.xml
- web.xml
- contains your backing beans config
Jsf and Lilya configuration

6.2. Create from scratch

In order to create your project from scratch:

- 1 - Copy all needed jars and tld files from lilya dist or lilya application template to your project.
- 2 - Create an empty faces-config.xml file under WEB-INF

```
<?xml version="1.0"?>
<!DOCTYPE faces-config
  PUBLIC "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN"
  "http://java.sun.com/dtd/web-facesconfig_1_1.dtd">
<faces-config xmlns="http://java.sun.com/JSF/Configuration">
</faces-config>
```

- 3 - Modify your web.xml web descriptor to enable jsf and Lilya

```
<context-param>
  <param-name>javax.faces.application.CONFIG_FILES</param-name>
  <param-value>/WEB-INF/faces-config.xml</param-value>
</context-param>
<context-param>
  <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
  <param-value>server</param-value>
</context-param>
<context-param>
  <param-name>com.sun.faces.validateXml</param-name>
  <param-value>>false</param-value>
</context-param>

<filter>
  <display-name>Ajax4jsf Filter</display-name>
  <filter-name>ajax4jsf</filter-name>
  <filter-class>org.ajax4jsf.Filter</filter-class>
</filter>
<filter-mapping>
  <filter-name>ajax4jsf</filter-name>
  <servlet-name>FacesServlet</servlet-name>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
  <dispatcher>INCLUDE</dispatcher>
</filter-mapping>

<!-- Faces Servlet -->
<servlet>
  <servlet-name>FacesServlet</servlet-name>
  <servlet-class>com.qlogic.products.lilya.JsfUIPlatformServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>FacesServlet</servlet-name>
  <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</jsp-config>
```



```
<taglib>
  <taglib-uri>http://java.sun.com/jsp/jstl/core</taglib-uri>
  <taglib-location>/WEB-INF/tld/c.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>http://java.sun.com/jsf/html</taglib-uri>
  <taglib-location>/WEB-INF/tld/html_basic.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>http://java.sun.com/jsf/core</taglib-uri>
  <taglib-location>/WEB-INF/tld/jsf_core.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>http://qlogic.net/web/jsf</taglib-uri>
  <taglib-location>/WEB-INF/tld/j2ee4all-jsf.tld</taglib-location>
</taglib>
<taglib>
  <taglib-uri>http://www.qlogic.ma/lilya</taglib-uri>
  <taglib-location>/WEB-INF/tld/lilya.tld</taglib-location>
</taglib>
</jsp-config>
```

- 4- Create a 'ui' directory under your project directory.
- 5- Copy lilyajax.js, lilya.css and lilya.js and 'images' from lilya 'ui' distribution to your 'ui' directory.
- 6 - Copy tld files from lilya 'tlds' distribution to WEB-INF/tld (c.tld, jsf_core.tld, html_basic.tld, j2ee4all-jsf.tld, lilya.tld).
- 7 - Download ExtJs Api from ExtJs web site <http://extjs.com/download>.
- 8 - Extract the zip file and copy 'adapter', 'resources' directories and ext-all.js, ext-all-debug.js files to 'ui' directory.
- 9 - Add a test page blank.jsp

```
<?xml version="1.0"?>
<?xml version="1.0" encoding="UTF-8"?>
<jsp:root version="1.2"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:ql="http://qlogic.net/web/jsf"
  xmlns:ui="http://www.qlogic.ma/lilya"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
<jsp:directive.page
  contentType="text/html;charset=UTF-8" pageEncoding="UTF-8"/>
<f:view>
<html>
  <head>
    <title>Blank Page Test</title>
    <ui:platform uipath="ui" />
  </head>
  <body>
    <ui:mask message="Loading Blank Page ..." />
  </body>
</html>
```

```
</f:view>
</jsp:root>
```

10 - Setup TomCat or your preferred application server to run your application

Edit [TOMCAT_HOME]/conf/server.xml by adding this line

```
<Context path="/[yourProject]" reloadable="true"
docBase="/[projects_dir]/[yourProject]"
workDir="/[projects_dir]/[yourProject]/work" />
```

Where /[projects_dir]/[yourProject] is your project directory.

Start TomCat (Using Eclipse or standalone) and access your blank page from <http://localhost:8080/yourProject/blank.jsf>

Some explanations :

- Note the use of `<ui:platform uipath="ui" />` to locate ExtJs ui directory (created during previous steps).
- We added `ui:mask` to see if everything is ok. This component waits for a page loading.

If your page runs without problems, you can continue with this tutorial, otherwise verify previous steps or submit a help request.



7. Quick Start

In coming examples, we'll use ExtJs's `examples.css`, `examples.js` and `menu/*.gif` files. Copy those files and copy them to your project directory.

In this chapter, we'll demonstrate how to use basic Lilya features.

- 7.1. Creating a Dialog
- 7.2. Creating a Toolbar Menu
- 7.3. Creating a Flat Menu (XP style)
- 7.4. Creating a Layout
- 7.5. Creating a Form
- 7.6. Creating a Dynamic Form

7.1. Creating a Dialog

Create a jsp page hello-dialog.jsp and copy/paste this code

```
<?xml version="1.0"?>
<?xml version="1.0" encoding="UTF-8"?>
<jsp:root version="1.2"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:jsp="http://java.sun.com/JSP/Page"
    xmlns:ql="http://qlogic.net/web/jsf"
    xmlns:ui="http://www.qlogic.ma/lilya"
    xmlns:c="http://java.sun.com/jsp/jstl/core">
<jsp:directive.page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"/>
<f:view>
  <html>
    <head>
      <title>Hello Dialog</title>
      <ui:platform uipath="ui" />
      <link rel="stylesheet" type="text/css" href="../examples.css" />
    </head>
    <body>
      <h1>Hello World Dialog</h1>
      <p>This example shows how to create a very simple modal Dialog with "autoTabs".</p>

      <input type="button" id="show-dialog-btn" value="Hello World" /><br /><br />

      <ui:dialog id="dialog" title="Hello Dialog" autoTabs="true" width="400"
        height="300" modal="true" minWidth="300" minHeight="250"
        proxyDrag="true" invokers="show-dialog-btn" >

        <ui:dialogBody>
          <!-- Auto create tab 1 -->
          <ui:dialogTab title="Hello World 1">
            <ql:plainText value="Put Html, Jsf markup here" />
          </ui:dialogTab>
          <!-- Auto create tab 2 -->
          <ui:dialogTab title="Hello World 2">
            <h:inputText value="... World!" />
          </ui:dialogTab>
        </ui:dialogBody>

        <ui:button id="submitDlg" text="Submit">
          <ui:invoke clientAction="dialog.hide" scope="dialog" inline="true"/>
        </ui:button>

        <ui:button id="closeDlg" text="Close">
          <ui:invoke clientAction="dialog.hide();" scope="dialog" />
        </ui:button>

      </ui:dialog>

    </body>
  </html>
</f:view>
</jsp:root>
```

Note that, invokers specify Dialog callers. Invokers syntax is : [event0/invokerId0,event1/invokerId1,...], for example :



```
invokers="mousemove/show-dialog-btn,anotherId"
```

If the event is not specified, default to click.

For more details about component's properties, see developer guide.

7.2. Creating a Toolbar Menu

Create a jsp page toolbar-menu.jsp and copy/paste this code

```
<?xml version="1.0"?>
<?xml version="1.0" encoding="UTF-8"?>
<jsp:root version="1.2" xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:jsf="http://java.sun.com/JSP/Page"
  xmlns:ql="http://qlogic.net/web/jsf"
  xmlns:ui="http://www.qlogic.ma/lilya"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
<jsp:directive.page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" />
  <f:view>
    <html>
      <head>
        <title>Toolbar with Menus</title>
        <ui:platform uipath="ui" />
        <script type="text/javascript" src="examples.js"></script>

        <!-- Common Styles for the examples -->
        <style type="text/css">
          #container {
            width:600px;
            height:300px;
            border:3px solid #c3daf9;
          }
          .calendar .x-menu-item-icon {
            background-image:url(ui/resources/images/default/shared/calendar.gif);
          }
          .blist .x-btn-text {
            background-image: url(list-items.gif)
          }
          .bmenu .x-btn-text {
            background-image: url(menu-show.gif)
          }
          .menu-title{
            background: #e6eadb url(ui/resources/images/aero/grid/grid-hrow.gif) repeat
            border-bottom:1px solid #99bbe8;
            color:#15428b;
            font:bold 10px tahoma,arial,verdana,sans-serif;
            display:block;
            padding:3px;
          }
        </style>
      </head>
      <body>
        <ql:div id="container" >
          <ui:toolBar>
            <ui:menuButton styleClass="x-btn-text-icon bmenu" text="Button w/ Menu" >
              <ui:menu >
                <ui:checkMenuItem text="I like Ext" checked="true"
                  clientAction="Ext.example.msg('Item Check', 'You {1} the &quot;{0}&quot;');
                  funcParams="item, checked"/>
                <ui:checkMenuItem text="Ext for jQuery" checked="true"
                  clientAction="Ext.example.msg('Item Check', 'You {1} the &quot;{0}&quot;');
                  funcParams="item, checked" />
                <ui:checkMenuItem text="I donated!" checked="false"
                  clientAction="Ext.example.msg('Item Check', 'You {1} the &quot;{0}&quot;');
                  funcParams="item, checked" />
                <ui:menuSeparator />
                <ui:menu text="Radio Options">
```



```
<ui:titleMenuItem text="&lt;b class='menu-title'&gt;Choose a Theme&lt;/b"
<ui:checkMenuItem text="Aero Glass" group="theme" checked="true"
  clientAction="Ext.example.msg('Item Check', 'You {1} the &quot;{0}&quot;."
  funcParams="item, checked" />
<ui:checkMenuItem text="Vista Black" group="theme"
  clientAction="Ext.example.msg('Item Check', 'You {1} the &quot;{0}&quot;."
  funcParams="item, checked" />
<ui:checkMenuItem text="Gray Theme" group="theme"
  clientAction="Ext.example.msg('Item Check', 'You {1} the &quot;{0}&quot;."
  funcParams="item, checked" />
<ui:checkMenuItem text="Default Theme" group="theme"
  clientAction="Ext.example.msg('Item Check', 'You {1} the &quot;{0}&quot;."
  funcParams="item, checked" />
</ui:menu>
<ui:dateMenu text="Choose a Date" styleClass="calendar"
  clientAction="Ext.example.msg('Date Selected', 'You chose {0}.!', date"
  funcParams="dp, date" />
<ui:colorMenu text="Choose a Color"
  clientAction="Ext.example.msg('Color Selected', 'You chose {0}.!', color"
  funcParams="cm, color" />
<ui:menuSeparator />
<ui:textMenuItem text="Dynamically added Item"
  clientAction="Ext.example.msg('Menu Click', 'You clicked the &quot;{0}&quot;."
  funcParams="item" />
<ui:textMenuItem text="Disabled Item" disabled="true"
  clientAction="Ext.example.msg('Menu Click', 'You clicked the &quot;{0}&quot;."
  funcParams="item" />
</ui:menu>
</ui:menuButton>
<ui:toolBarSeparator />
<ui:menuButton styleClass="x-btn-text-icon blist"
  tooltip="&lt;b&gt;Tip Title&lt;/b&gt;&lt;br/&gt;This is a QuickTip with text"
  text="Split Button"
  clientAction="Ext.example.msg('Button Toggled', 'Button &quot;{0}&quot;."
  funcParams="item,pressed">
<ui:menu>
  <ui:textMenuItem text="Bold" textStyle="font-weight: bold; "
    clientAction="Ext.example.msg('Menu Click', 'You clicked the &quot;{0}&quot;."
    funcParams="item" />
  <ui:textMenuItem text="Italic" textStyle="font-style: italic; "
    clientAction="Ext.example.msg('Menu Click', 'You clicked the &quot;{0}&quot;."
    funcParams="item" />
  <ui:textMenuItem text="Underline" textStyle="text-decoration: underline;"
    clientAction="Ext.example.msg('Menu Click', 'You clicked the &quot;{0}&quot;."
    funcParams="item" />
  <ui:menuSeparator />
  <ui:menu text="Pick a Color" id="chooseColor">
    <ui:colorMenuItem
      clientAction="Ext.example.msg('Color Selected', 'You chose {0}.!', color"
      funcParams="cp, color" />
    <ui:menuSeparator />
    <ui:textMenuItem text="More Colors..." clientAction="Ext.example.ms
  </ui:menu>
  <ui:textMenuItem text="Excellent!"
    clientAction="Ext.example.msg('Menu Click', 'You clicked the &quot;{0}&quot;."
    funcParams="item" />
</ui:menu>
</ui:menuButton>
<ui:toolBarSeparator />
<ui:menuButton text="Toggle Me" pressed="true"
  clientAction="Ext.example.msg('Button Toggled', 'Button &quot;{0}&quot;."
  funcParams="item,pressed"/>
<ui:toolBarSeparator />
<ui:menuButton styleClass="x-btn-icon" icon="list-items.gif"
```

```
        tooltip="&lt;b&gt;Quick Tips&lt;/b&gt;&lt;br/&gt;Icon only button with  
    </ui:toolBar>  
  </ql:div>  
</body>  
</html>  
</f:view>  
</jsp:root>
```

In this example we produced ExtJs menu example using Lilya. We added the `clientAction` to handle user interaction.

`Ext.example.msg` function exists in `examples.js` : is a utility function that displays a box showing the event's target and value


In next chapter, we will explain in details event handling and client/server features.



7.3. Creating a Flat Menu (XP style)

Create a jsp page flat-menu.jsp and copy/paste this code

```
<?xml version="1.0"?>
<?xml version="1.0" encoding="UTF-8"?>
<jsp:root version="1.2" xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:ql="http://qlogic.net/web/jsf"
  xmlns:ui="http://www.qlogic.ma/lilya"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
  <jsp:directive.page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" />
  <f:view>
    <html>
      <head>
        <title>Flat Menu</title>
        <ui:platform uipath="ui" />
        <style type="text/css">
          .calendar .x-menu-item-icon {
            background-image: url(ui/resources/images/default/shared/calendar.gif);
          }
        </style>
      </head>
      <body>
        <ui:menu flatStyle="true">
          <ui:menu text="Menu One">
            <ui:checkMenuItem text="I like Ext" checked="true" />
            <ui:checkMenuItem text="Ext for jQuery" checked="true" />
            <ui:checkMenuItem text="I donated!" checked="false" />
            <ui:menuSeparator />
            <ui:menu text="Radio Options">
              <ui:titleMenuItem text="&lt;b class='menu-title'&gt;Choose a Theme&lt;/b" />
              <ui:checkMenuItem text="Aero Glass" group="theme" checked="true" />
              <ui:checkMenuItem text="Vista Black" group="theme" />
              <ui:checkMenuItem text="Gray Theme" group="theme" />
              <ui:checkMenuItem text="Default Theme" group="theme" />
            </ui:menu>
            <ui:dateMenu text="Choose a Date" styleClass="calendar" />
            <ui:colorMenu text="Choose a Color" />
            <ui:menuSeparator />
            <ui:textMenuItem text="Dynamically added Item" />
            <ui:textMenuItem text="Disabled Item" disabled="true" />
          </ui:menu>
          <ui:menu text="Menu Two">
            <ui:textMenuItem text="Bold" textStyle="font-weight: bold; " />
            <ui:textMenuItem text="Italic" textStyle="font-style: italic; " />
            <ui:textMenuItem text="Underline" textStyle="text-decoration: underline;" />
            <ui:menuSeparator />
            <ui:menu text="Pick a Color" id="chooseColor">
              <ui:colorMenuItem />
              <ui:menuSeparator />
              <ui:textMenuItem text="More Colors..." />
            </ui:menu>
            <ui:textMenuItem text="Excellent!" />
          </ui:menu>
        </ui:menu>
      </ql:div>
    </body>
  </html>
</f:view>
</jsp:root>
```



In this example we produced a flat menu without a toolbar.
We removed clientAction(s) to simplify the example. you can add them and you will get the same behaviour like a toolbar menu.



7.4. Creating a Layout

Create a jsp page flat-menu.jsp and copy/paste this code

```
<?xml version="1.0"?>
<?xml version="1.0" encoding="UTF-8"?>
<jsp:root version="1.2" xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:ql="http://qlogic.net/web/jsf"
  xmlns:ui="http://www.qlogic.ma/lilya"
  xmlns:c="http://java.sun.com/jsp/jstl/core">
<jsp:directive.page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" />
  <f:view>
    <html>
      <head>
        <title>Layout</title>
        <ui:platform uipath="ui" />
      </head>
      <body>
        <ui:layout>
          <ui:region geography="west" split="true" initialSize="150" minSize="100" maxSize="200"
            titlebar="true" collapsible="true" animate="true" >
            <ui:contentPanel title="West" />
          </ui:region>
          <ui:region geography="center" autoScroll="true" tabPosition="top" closeOnTabChange="true" >
            <ui:contentPanel title="The First Tab"/>
            <ui:contentPanel autoCreate="true" title="Another Tab" background="true">
              <ql:plainText value="put any jsf component here..." />
            </ui:contentPanel>
            <ui:contentPanel autoCreate="true" title="Third Tab" closable="true" background="true">
            </ui:contentPanel>
          </ui:region>
        </ui:layout>
      </body>
    </html>
  </f:view>
</jsp:root>
```

7.5. Creating a Form



7.6. Creating a Dynamic From

8. Architecture

8.1.Overview

8.2.Ajax Capabilities



8.1. Overview

8.2. Ajax Capabilities



9. Working With Data

- 9.1.DataStore/DataModel and Server Implementation
- 9.2.Configure
- 9.3.Grid Example
- 9.4.From Example

9.1. DataStore/DataModel and Server Implementation



9.2. Configure

9.3. Grid Example



9.4. From Example

10. Writing Events

10.1.Client Events

10.2.Server Events

10.3.Decide What to update

10.4.Understanding the difference between reRendering and Updating

10.5.Form Example



10.1. Client Events

10.2. Server Events



10.3. Decide What to update

10.4. Understanding the difference between reRendering and Updating



10.5. Form Example

11. Extending Lilya



12.1. Adding new Platform Definitions

12.2. Adding new Platform Editors



12.3. Creating your own components

