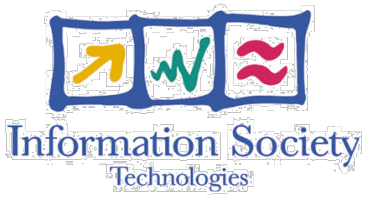
# Hands on Portlets


## *Writing a Portlet*

Albert Einstein Institute

# Hands on Session!

## Goals of this session

Write a sophisticated portlet using the GridSphere portlet model based upon JSR 168 Portlet API

Learn the UI tag and bean model for rapid portlet development

You will have a running portal on your machine after this tutorial!

It’s not hard! :-)

# Required software

- External software:
  - Java 1.4.2 (from http://java.sun.com)
  - Jakarta Tomcat 4 or 5 (from http://www.apache.org/dist/jakarta/tomcat)
  - Jakarta Ant >= 1.6 (from http://www.apache.org/dist/ant/binaries)
- GridSphere from CVS

```
cvs -d :pserver:anonymous@portal.aei.mpg.de:/home/repository login
cvs -d :pserver:anonymous@portal.aei.mpg.de:/home/repository  co gridsphere
```

- jsrtutorial from cvs

```
cd gridsphere/; mkdir projects; cd projects
cvs -d :pserver:anonymous@portal.aei.mpg.de:/home/repository  co jsrtutorial
```

# Install and startup

- Make sure $CATALINA_HOME environment variable is set to the rootdir of the tomcat installation
- Install gridsphere and tutorial web applications

```
cd gridsphere; ant install; cd projects/jsrtutorial/; ant install
```

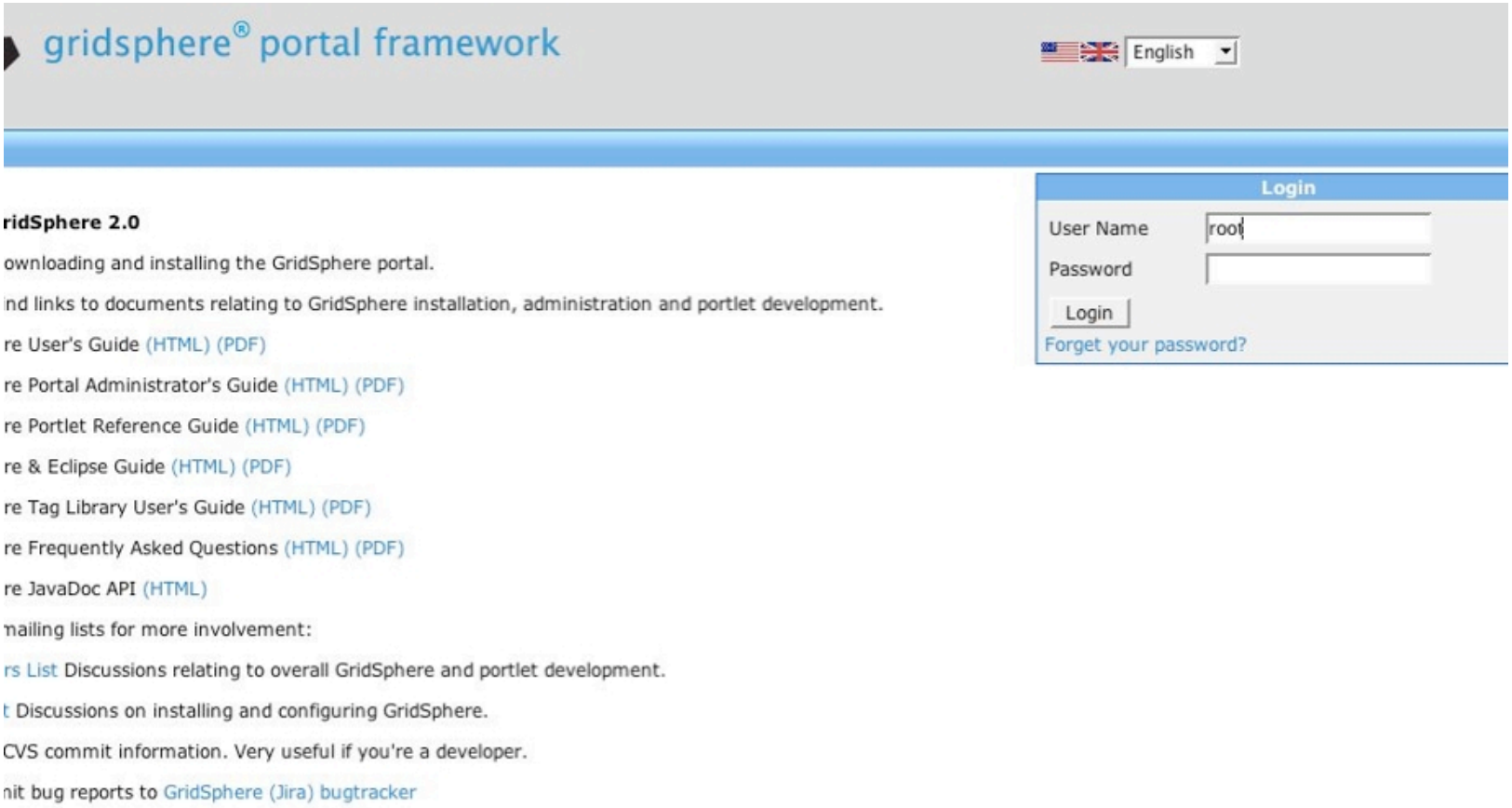
- Startup tomcat

```
$CATALINA_HOME/bin/startup.sh
```

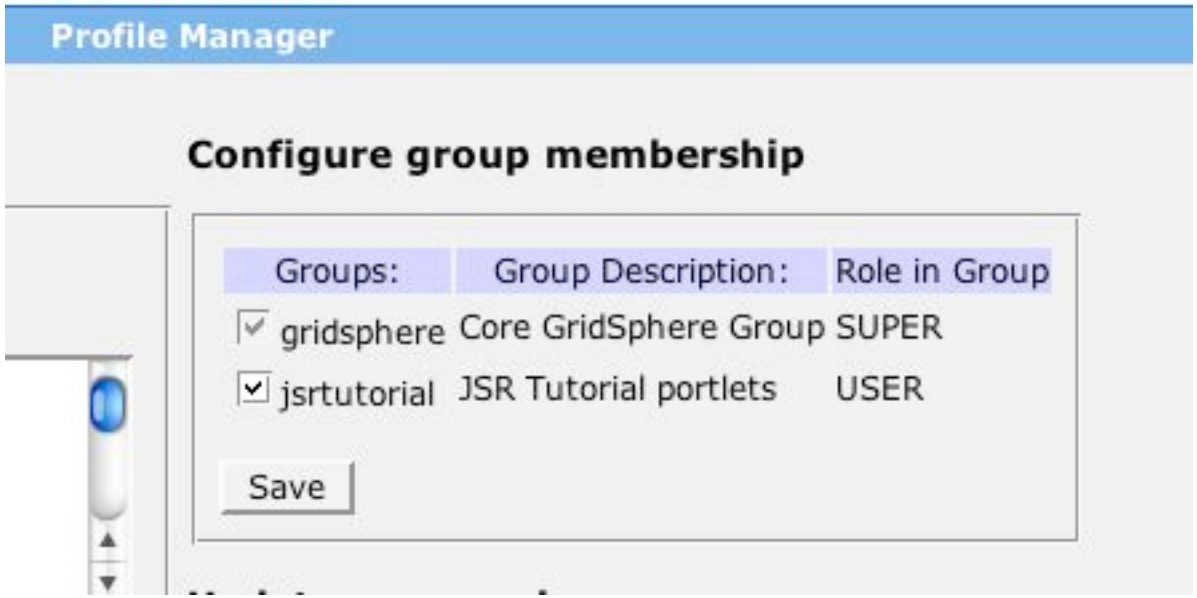
- Point your browser to `http://localhost:8080/gridsphere/gridsphere`

# Running Portal

- Log in as root with no password

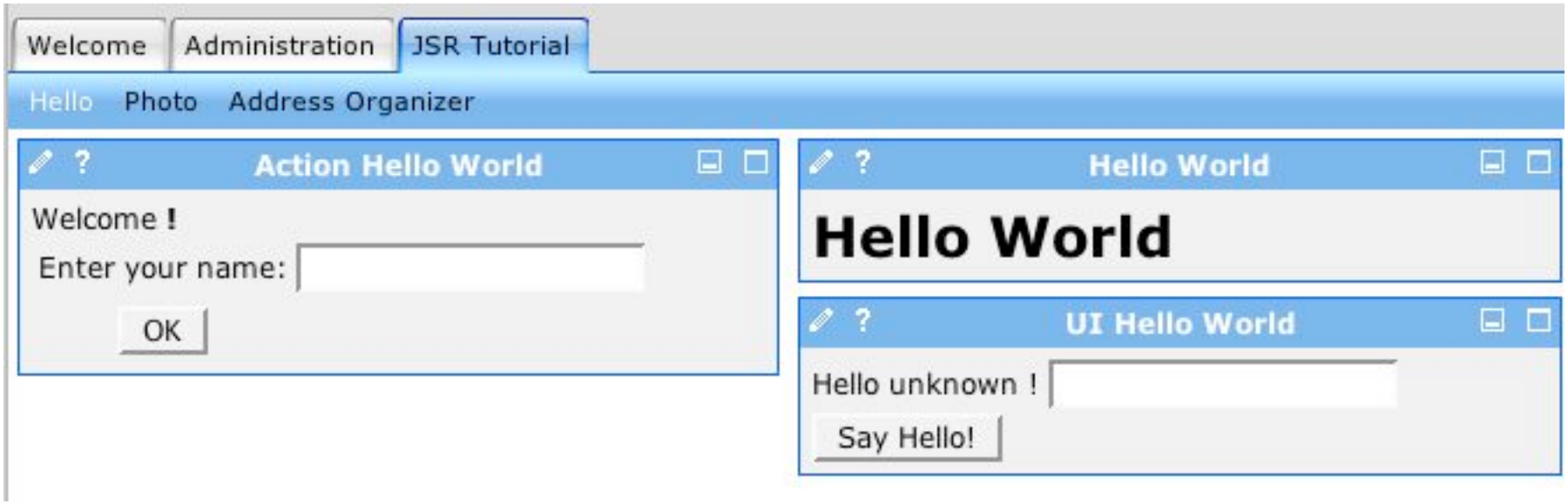


# Subscribe to jsrtutorial group



# Hello World!

- Tutorial webapp is loaded as well
- HelloWorld and ActionHelloWorld are on the ‘Tutorial’ Tab



# Simple Hello World Portlet!

- Look at following portlet in src/org/gridlab/gridsphere/jsrtutorial/portlets/helloworld/HelloWorld.java

```
package org.gridlab.gridsphere.jsrtutorial.portlets.helloworld;


import javax.portlet.GenericPortlet;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;
import javax.portlet.PortletException;
import java.io.PrintWriter;
import java.io.IOException;

/**
 * a simple HelloWorld Portlet
 */
public class HelloWorld extends GenericPortlet {

    public void doView(RenderRequest request, RenderResponse response)
            throws PortletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<h1>Hello World</h1>");
    }

}
```

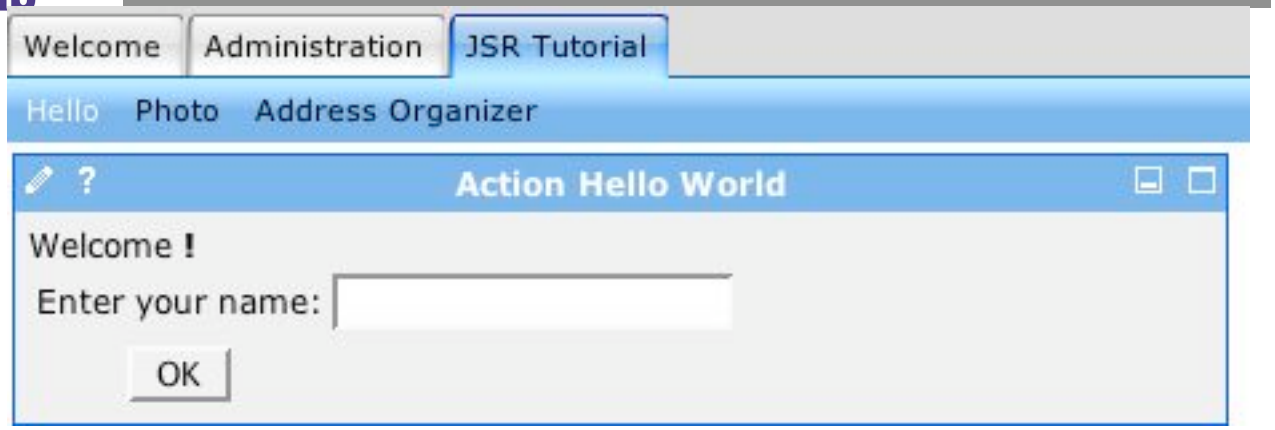
# Action Hello World Portlet

- Look at following portlet in src/org/gridlab/gridsphere/jsrtutorial/portlets/helloworld/ActionHelloWorld.java

```
public void doView(RenderRequest request, RenderResponse response) throws PortletException,
IOException {

        // retrieve the name
        String name = request.getParameter("name");
        if (name != null) request.setAttribute("name", name);

        // create action url
        PortletURL url = response.createActionURL();
        request.setAttribute("url", url.toString());
        getPortletConfig().getPortletContext().getRequestDispatcher("/jsp/helloworld/
actionhello.jsp").include(request, response);
    }
```

- if name exists store in request
- create action url used to invoke portlet processAction
- includes a JSP for rendering

# Action Hello World Portlet




```
<jsp:useBean id="url" class="java.lang.String" scope="request"/>
<jsp:useBean id="name" class="java.lang.String" scope="request"/>

Welcome <b><%= name %>!</b>

<form method="POST" action="<%= url %>">
    <table>
        <tr>
            <td>Enter your name:</td>
            <td><input type="text" name="name" size="20" maxlength="20"/></td>
        </tr>
        <tr>
            <td align="center"><input type="submit" value="OK"/></td>
        </tr>
    </table>
</form>
```

# Action Hello World Portlet

- Look at processAction

```
public void processAction(ActionRequest req, ActionResponse res) throws PortletException,
IOException {

        String name = req.getParameter("name");

        // must be passed into render parameters
        if (name != null) res.setRenderParameter("name", name);
    }
```

- name is retrieved from request
- Must be placed as a render parameter for view method

# UiHelloWorld.java

src/org/gridlab/gridsphere/jsrtutorial/portlets/helloworld/

- GridSphere provides an easy and unified way to handle HTML forms and elements

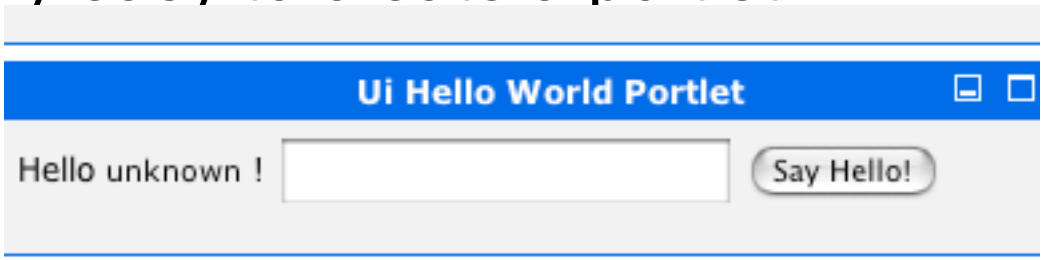
```
public class UiHelloWorld extends ActionPortlet {

    private static final String DISPLAY_PAGE = "helloworld/uihelloworld.jsp";

    public void init(PortletConfig config) throws PortletException {
        super.init(config);
        DEFAULT_VIEW_PAGE = "prepare";
    }

    public void prepare(RenderFormEvent event) throws PortletException {
        TextBean helloname = event.getTextBean("helloname");
        if (helloname.getValue()==null) {
            helloname.setValue("unknown");
        }
        setNextState(event.getRenderRequest(), DISPLAY_PAGE);
    }

    public void showName(ActionFormEvent event) throws PortletException {
        TextFieldBean name = event.getTextFieldBean("name");
        TextBean helloname = event.getTextBean("helloname");
        helloname.setValue(name.getValue());
        setNextState(event.getPortletRequest(), DISPLAY_PAGE);
    }
}
```

# uihelloworld.jsp

- First three lines load and init the jsp Gridsphere JSP Taglibs

```
<%@ taglib uri="/portletUI" prefix="ui" %>
<%@ taglib uri="http://java.sun.com/portlet" prefix="portlet" %>
<portlet:defineObjects/>

<ui:form>
    Hello <ui:text beanId="helloname"/> !
    <ui:textfield size="20" beanId="name"/>
    <ui:actionsubmit action="showName" value="Say Hello!"/>
</ui:form>
```

- Very easy to create a portlet



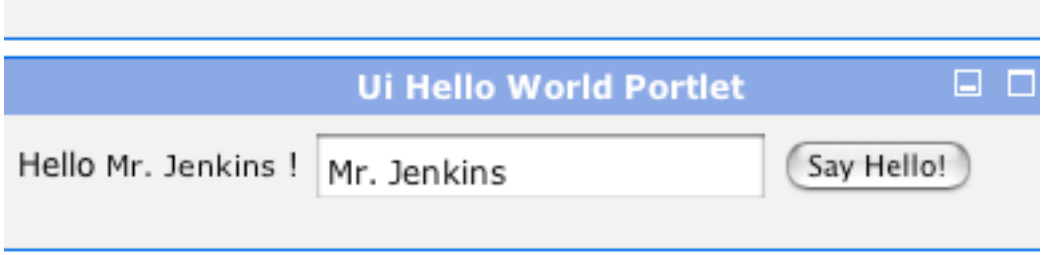
# Handling an Action

- How is the button click handled?

```
public class UiHelloWorld extends ActionPortlet {

    private static final String DISPLAY_PAGE = "helloworld/uihelloworld.jsp";

    public void init(PortletConfig config) throws PortletException {
        super.init(config);
        DEFAULT_VIEW_PAGE = "prepare";
    }

    public void prepare(RenderFormEvent event) throws PortletException {
        TextBean helloname = event.getTextBean("helloname");
        if (helloname.getValue()==null) {
            helloname.setValue("unknown");
        }
        setNextState(event.getRenderRequest(), DISPLAY_PAGE);
    }
   public void showName(ActionFormEvent event) throws PortletException {
        TextFieldBean name = event.getTextFieldBean("name");
        TextBean helloname = event.getTextBean("helloname");
        helloname.setValue(name.getValue());
        setNextState(event.getActionRequest(), DISPLAY_PAGE);
    }}
```

# How does it work?

- All beans are constructed from FormEvent
- getXXXBean returns the object from the request or creates a new empty one
- `TextFieldBean name = event.getTextFieldBean("name");` returns a TextField containing the value for that field and represents that as a bean
- `helloname.setValue(name.getValue());` sets the value of the Text
- `setNextState(event.getPortletRequest(), this.show);` sets the next display state



# Enhance UiHelloWorld

webapp/jsp/helloworld/uihelloworld.jsp

- This will make it possible to let the name be in bold font
- We choose a checkbox to do that

```
<%@ taglib uri="/portletUI" prefix="ui" %>
<%@ taglib uri="/portletAPI" prefix="portletAPI" %>
<portletAPI:init/>

<ui:form>
    Hello <ui:text beanId="helloname"/> !
    <ui:textfield size="20" beanId="name"/>

     Bold : <ui:checkbox beanId="bold"/>

    <ui:actionsubmit action="showName" value="Say Hello!"/>
</ui:form>
```

# Enhance UiHelloWorld (cont)

src/org/gridlab/gridsphere/tutorial/portlets/helloworld/UiHelloWorld.java

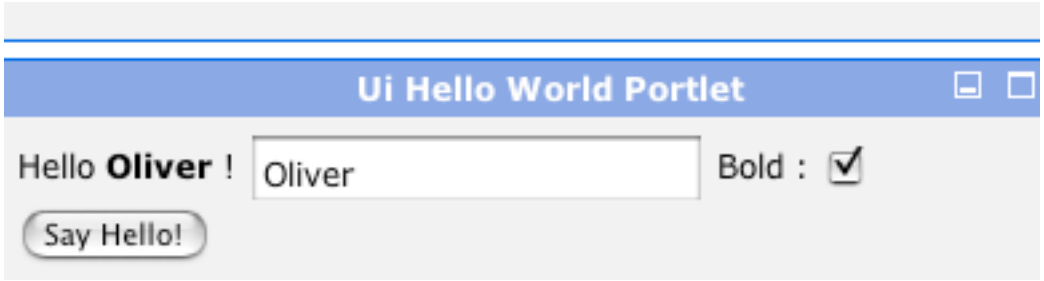
- to check if checkbox is checked insert the following code in the portlet

```
public class UiHelloWorld extends ActionPortlet {

    private String show = "helloworld/uihelloworld.jsp";

    public void init(PortletConfig config) throws UnavailableException {
        super.init(config);
        DEFAULT_VIEW_PAGE = "prepare";
    }

...

    public void showName(FormEvent event) throws PortletException {
        TextFieldBean name = event.getTextFieldBean("name");
        TextBean helloname = event.getTextBean("helloname");
        helloname.setValue(name.getValue());
        CheckBoxBean bold = event.getCheckBoxBean("bold");
        if (bold.isSelected()) {
            helloname.setStyle(MessageStyle.MSG_BOLD);
        }

        setNextState(event.getPortletRequest(), this.show);
    }
}
```

# Review UiHelloWorld

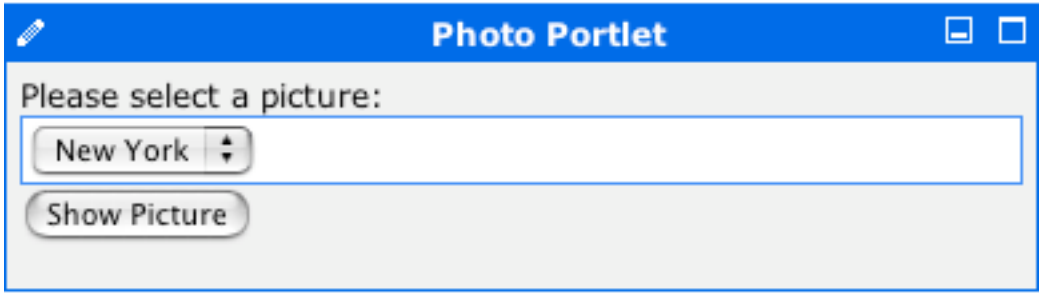
- redeploy the tutorial webapp  restart tomcat or use portlet manager

```
$CATALINA_HOME/bin/shutdown.sh
ant deploy
$CATALINA_HOME/bin/startup.sh
```



# PhotoPortlet

- PhotoPortlet is a simple Portlet which shows a fixed number of pictures from an url. Pictures are local on tomcat and came with tutorial webapp.





- We are going to change that!
- project dir: gridsphere/project/tutorial

# Sourcecode-PhotoPortlet.java

projects/tutorial/src/org/gridlab/gridsphere/portlets/tutorial/photo

```
1   class PhotoPortlet extends ActionPortlet {
2   public void init(PortletConfig config) throws UnavailableException {
3        super.init(config);
4        DEFAULT_VIEW_PAGE = "showSelection";
5        list.add(new PhotoURL(pic1, desc1));
6        list.add(new PhotoURL(pic2, desc2));
7        list.add(new PhotoURL(pic3, desc3));
8   }
9   public void showSelection(ActionFormEvent event) throws PortletException {
10       setNextState(event.getActionRequest(), DEFAULT_VIEW_PAGE);
11  }
12
13  protected void setListBox(FormEvent event) {
14       ListBoxBean lb = event.getListBoxBean("photolist");
15       lb.setSize(1);
16       lb.setMultipleSelection(false);
17       for (int i = 0; i < list.size(); i++) {
18           PhotoURL photo = (PhotoURL) list.get(i);
19           lb.addBean(makeItem(photo.getDesc(), photo.getUrl()));
20       }
21  }
```

# Sourcecode (cont)

- view.jsp

```
1 <ui:listbox beanId="photolist"/>
2 <ui:actionsubmit action="showPicture" value="Show Picture"/>
```

- PhotoPortlet.java

```
1    public void showPicture(ActionFormEvent event) throws PortletException {
2        ActionRequest request = event.getActionRequest();
3        ListBoxBean lb = event.getListBoxBean("photolist");
4        String url = lb.getSelectedValue();
5        ImageBean image = event.getImageBean("urlphoto");
6        image.setSrc(url);
7        setNextState(request, "photo/picture.jsp");
8    }
```

- picture.jsp

```
1 <ui:image beanId="urlphoto"/>
2 <ui:actionsubmit action="showSelection" value="Show Selection"/>
```

# Modifying the portlet

- to make the pictures available to other portlets we need to use a portletservice
- 3 Steps to create the service
  - create an interface
  - create an implementation
  - create a descriptor for the service
- to make it a dynamic list we will create the edit mode to add new pictures

# Adding the service

- services for portlets have to be created in the services subdirectory of the portletsource
- the interface looks like

```
1 public interface PhotoService extends PortletService {
2   public void add(String url, String desc);
3    public List getPictures(); }
```

- the implementation

```
 1 public class PhotoServiceImpl implements PortletServiceProvider, PhotoService {
 2   private List photolist = new ArrayList();
 3    public void init(PortletServiceConfig config) throws
PortletServiceUnavailableException {}
 4    public void destroy() {}
 5    public void add(String url, String desc) {
 6        PhotoURL photo = new PhotoURL(url, desc);photolist.add(photo);}
 7    public List getPictures() {
 8        return photolist; }
 9 }
```

# Adding the service (cont)

- webapps/WEB-INF/PortletServices.xml

```
1 <portlet-services>
2   <service>
3        <name>PhotoService</name>
4        <description>an example photo service</description>
5        <interface>org.gridlab.gridsphere.tutorial.services.photo.PhotoService</
interface>
6
<implementation>org.gridlab.gridsphere.tutorial.services.photo.impl.PhotoServiceImpl</
implementation>
7    </service>
8 </portlet-services>
```

- the services will compiled into separate jar
- everybody who uses this service could now add to and retrieve the pictures

# Adding the service (cont)

- to use the service in the portlet you need to get a handle on it

```
1 public void init(PortletConfig config) throws UnavailableException {
2       super.init(config);
3
4       DEFAULT_VIEW_PAGE = "showSelection";
5   try {
6               photoService = (PhotoService)
7   this.createPortletService(PhotoService.class);
8           } catch (PortletServiceException e) {
9               log.error("Unable to initialize PhotoService", e);
10          }
11          photoService.add(pic1, desc1);
12          photoService.add(pic2, desc2);
13          photoService.add(pic3, desc3);
14 }
```

# Creating Edit Mode

- we want to be able to add new pictures in ‘edit’ mode of the PhotoPortlet

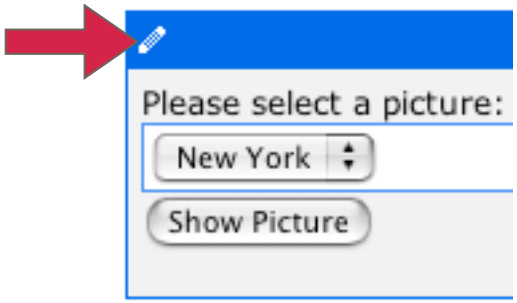
```
1  public void init(PortletConfig config) throws UnavailableException {
2       super.init(config);
3      DEFAULT_EDIT_PAGE = "photo/edit.jsp";
```

- edit.jsp defines the userinterface for adding pictures

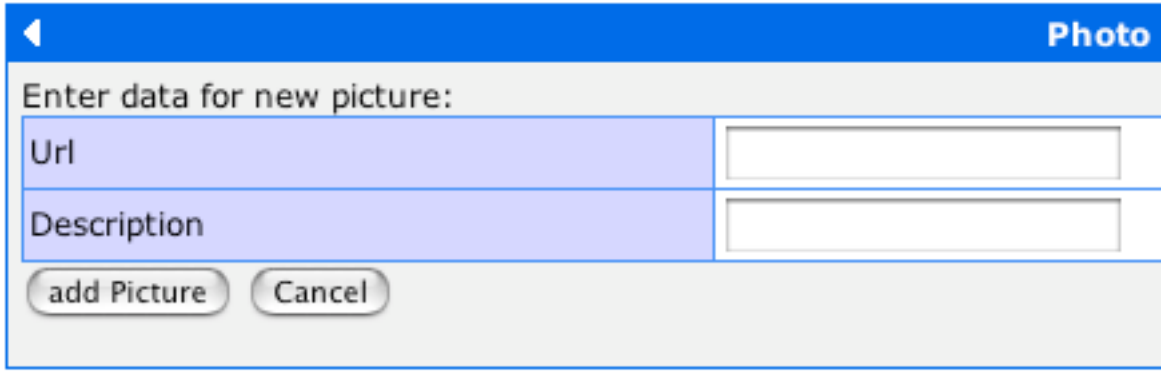
```
1 <ui:text value="Url"/>
2 <ui:textfield beanId="url"/>
3 <ui:text value="Description"/>
4 <ui:textfield beanId="desc"/>
5 <ui:actionsubmit action="addPicture" value="add Picture"/>
6 <ui:actionsubmit action="showSelection" value="Cancel"/>
```

# Using ‘Edit’ Mode

- if you click on the ‘Edit’ Mode icon




- ... the edit mode as just defined will show up



# Using the service

- modify (uncomment) PhotoPortlet.java to add the action needed to add data to the service

```
1   // uncomment this for using the editmode
2   public void addPicture(FormEvent event) throws PortletException {
3       PortletRequest request = event.getPortletRequest();
4       TextFieldBean url = event.getTextFieldBean("url");
5       TextFieldBean desc = event.getTextFieldBean("desc");
6       photoService.add(url.getValue(), desc.getValue());
7       request.setMode(Portlet.Mode.VIEW);
8       setNextState(request, "showSelection");
9   }
```

- one more change in the source to make the service work

# Make the service work

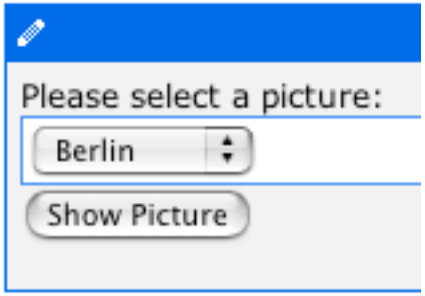
- the listbox in showSelection() should now get the pictures from the service

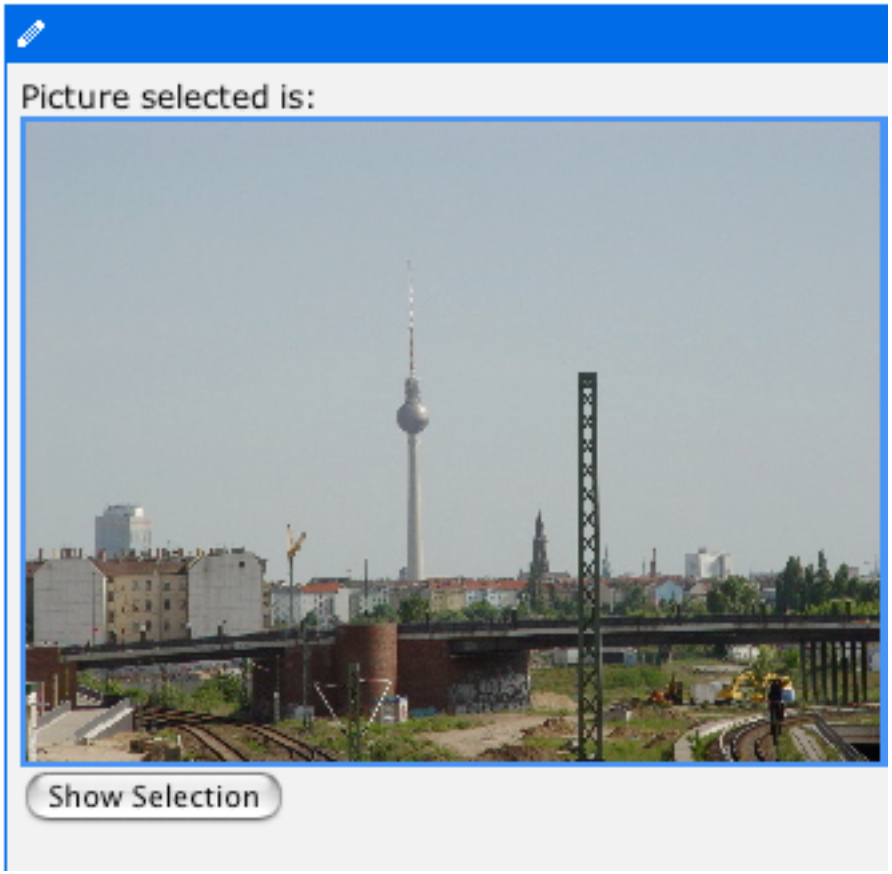
```
1
2 List list = photoService.getPictures();
3
4 for (int i = 0; i < list.size(); i++) {
5            PhotoURL photo = (PhotoURL) list.get(i);
6            lb.addBean(makeItem(photo.getDesc(), photo.getUrl()));
7        }
```

- redeploy the tutorial webapp

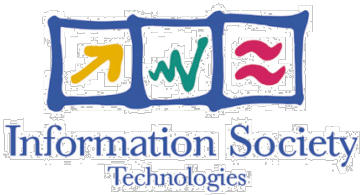
# Using Edit Mode and Service

- In Edit mode add `http://localhost:8080/tutorial/html/images/berlin.jpg` as URL and Berlin as Description


# Making objects persistent




- GridSphere's persistence layer uses Hibernate (http://hibernate.org) for storing data in SQL databases
- During project creation a sample hibernate.properties file is copied to `webapp/WEB-INF/persistence/`
- No modification is necessary if using the build in pure Java HSQL databases fits the developers needs
- Examples of how to connect to MySQL/McKoi are in the properties file as well
- Hibernate can connect to DB2, MySQL, PostgreSQL, Oracle, Sybase, Microsoft SQL Server and more

# Saving objects

- Object to be saved should have getter/setter methods on all required fields
- A unique object identifier is needed (with getter/setter methods as well)

```
public class Address {
   private String oid = null;
   private String firstname = new String();
   private String lastname = new String();
   public String getOid() {return oid;}
   public void setOid(String oid) {this.oid = oid; }
   public String getFirstname() {return firstname; }
   public void setFirstname(String fn) {this.firstname = fn;}
   public String getLastname() {return lastname;}
   public void setLastname(String ln) {this.lastname = ln;}
}
```

# Object Mapping File

- Hibernate requires each persistent object to have a mapping file with the description of the datastructure of the object
- Place one or more mapping files in `webapp/WEB-INF/persistence` describing one or more objects

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 1.1//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-2.0.dtd">
<hibernate-mapping>
 <class name="org.gridlab.gridsphere.tutorial.services.address.Address"
      table="tutorial_address">
        <id name="oid" column="oid" type="java.lang.String" length="32">
            <generator class="uuid.hex"/>
        </id>
        <property name="Lastname" type="string" column="lastname"/>
        <property name="Firstname" type="string" column="firstname"/>
    </class>

</hibernate-mapping>
```

class and databasetable definition

objectid and fields definition

# Saving Objects

- Objects are saved from services to make them reusable components
- The persistent object needs to be placed in the `services` directory
- In the service implementation the gridsphere persistence manager singleton offers methods to **c**reate/**r**ead/**u**pdate/**d**elete objects
- Each 3rd party web application has it's own persistence manager and can only access it's own data

# Persistence Manager

- In the init method of the service the creation of the persistence manager (PM) is suggested

```
this.pm = PersistenceManagerFactory.
    createPersistenceManagerRdbms("tutorial");
```

- The PM is initialized with the name of the web application, this will load all needed database connection/objectmapping settings
- The destroy() method frees all resources with:

```
pm.destroy()
```

# CRUD objects

- Create an object in the database the PM provides the `create(Object o)` method a `update(Object o)` `to` update an object
- Hibernate will take care of the the extra oid field which is needed
- Once an object is stored hibernate will set oid to be an unique id; you can use it later to refer to the object
- To retrieve objects two methods are provided `restore(String query)` and `restoreList(String query) ;` query is a standard Hibernate ObjectQueryLanguage query

# CRUD objects II

- Retrieve all address objects by lastname:

```
result = pm.restoreList("from " +
Address.class.getName() + " as address where
address.lastname='" + lastname + "'");
```

- `retrieve(String query)` returns the first element of a `retrieveList(String query)`; e.g. when querying for the unique oid
- To delete an object `delete(Object o)` is used; this will delete the object from the database but not from memory

# More information

- Any hibernate supported data structure (like lists, sets) can be used in the objects
- Please refer to the hibernate documentation for OQL and hibernate mapping files (http://www.hibernate.org/hib_docs/reference/html/)

# Configuration files

projects/tutorial/webapp/WEB-INF/

- portlet.xml -- contains

```
<portlet>
   <description xml:lang="en">Provides a persistent, searchable address book</description>
   <description xml:lang="de"></description>
   <portlet-name>AddressPortlet</portlet-name>
   <display-name xml:lang="en">Address Book Portlet</display-name>
   <display-name xml:lang="de">AddressPortlet</display-name>
   <portlet-class>org.gridlab.gridsphere.jsrtutorial.portlets.address.AddressPortlet</portlet-cla
   <expiration-cache>60</expiration-cache>
   <supports>
           <mime-type>text/html</mime-type>
           <portlet-mode>edit</portlet-mode>
           <portlet-mode>help</portlet-mode>
    </supports>
    <supported-locale>en</supported-locale>
    <portlet-info>
       <title>Address Book</title>
       <short-title>Address Book</short-title>
       <keywords>Address, persistence</keywords>
   </portlet-info>
</portlet>
```

# Configuration files

projects/tutorial/webapp/WEB-INF/group.xml

- group.xml -- contains

```
<portlet-group>
    <group-name>JSRTutorial</group-name>
    <group-description>JSR Tutorial portlets</group-description>
    <group-visibility>PUBLIC</group-visibility>
    <portlet-role-info>
        <portlet-class>
              org.gridlab.gridsphere.jsrtutorial.portlets.helloworld.ActionHelloWorld
       </portlet-class>
        <required-role>USER</required-role>
    </portlet-role-info>

     • • •
</portlet-group>
```

- name and description
- visibility - public == anyone can subscribe
- required role per portlet

# Configuration files

projects/tutorial/webapp/WEB-INF/layout.xml

- layout.xml -- contains

```
<portlet-tabbed-pane>
    <portlet-tab>
        <title lang="en">JSR Tutorial</title>
        <portlet-tabbed-pane style="sub-menu">
            <portlet-tab label="helloportlet">
                <title lang="en">Hello</title>
                <table-layout>
                    <row-layout>
                        <column-layout width="50%">
                            <portlet-frame>
                                <portlet-class>
org.gridlab.gridsphere.jsrtutorial.portlets.helloworld.ActionHelloWorld
                                </portlet-class>
                            </portlet-frame>
                            ...
```

- Consult reference guide for more details


# Questions ?