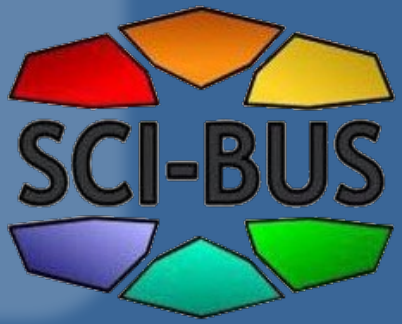
WS-PGRADE Portal Introduction

WS-PGRADE Customization: Portlets

Eduardo Lostal
Zaragoza, 17th May 2012

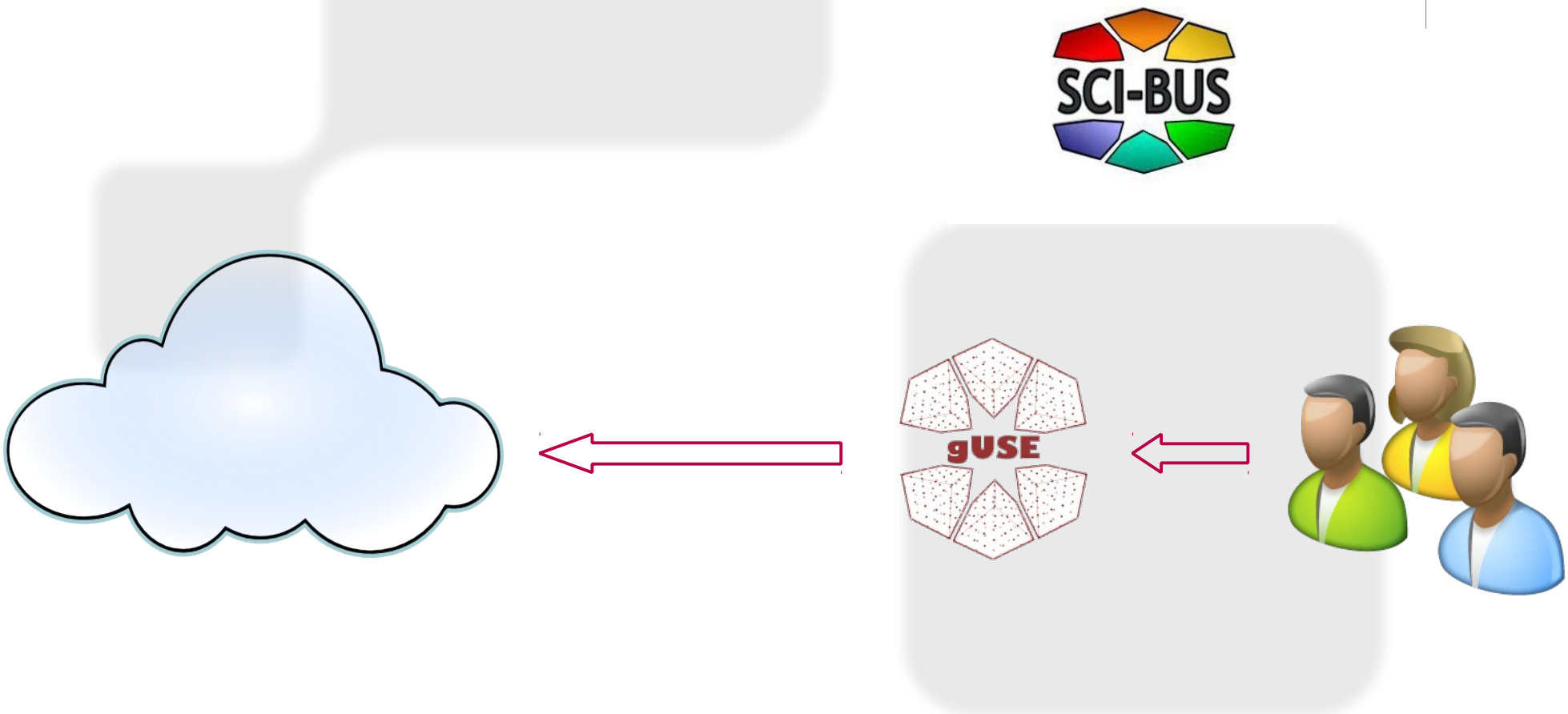


Instituto Universitario de Investigación
Biocomputación y Física de Sistemas Complejos
Universidad Zaragoza

Structure of the presentation

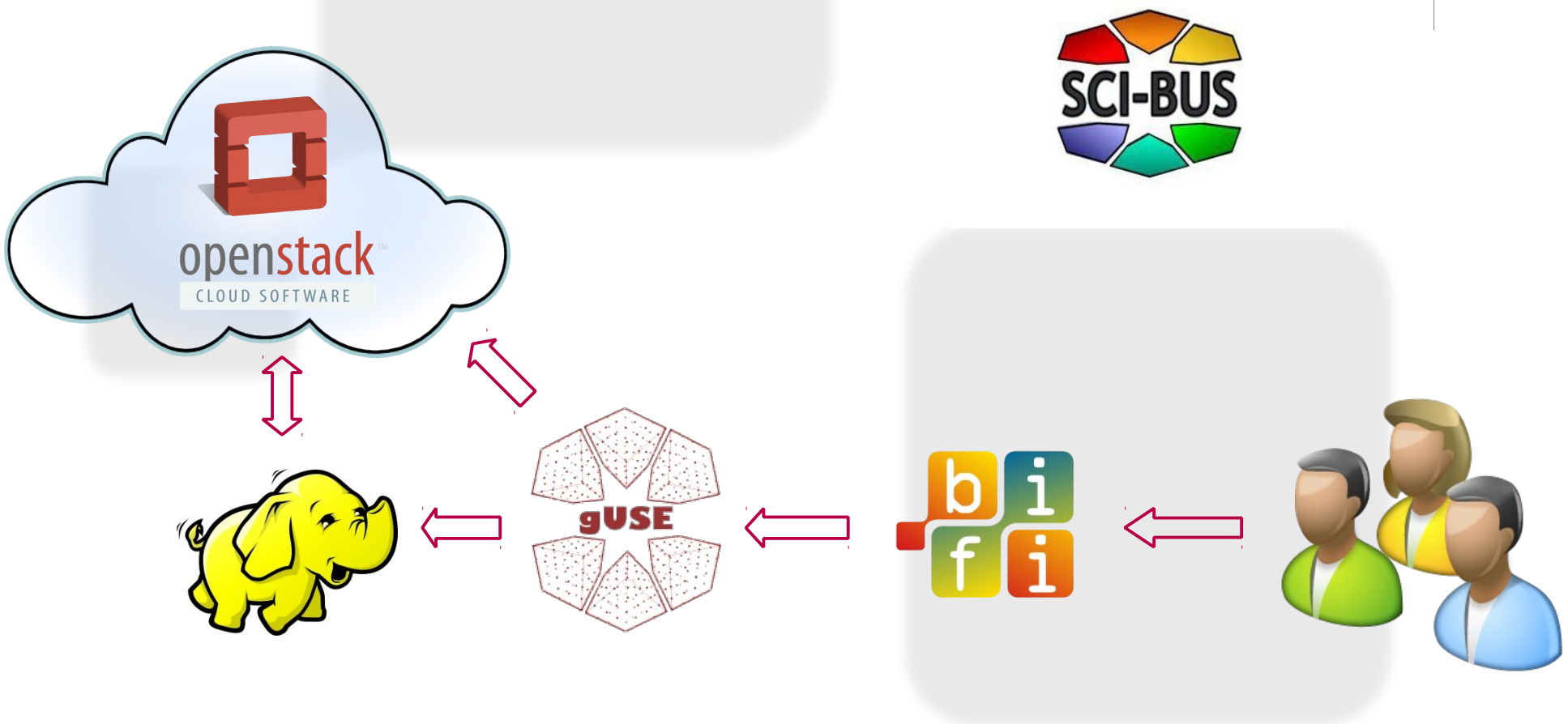
- SCI-BUS Development
- Liferay Portal
- Portlets
- Conclusion

What is being developed



WS-PGRADE Customization: Portlets

What is being developed

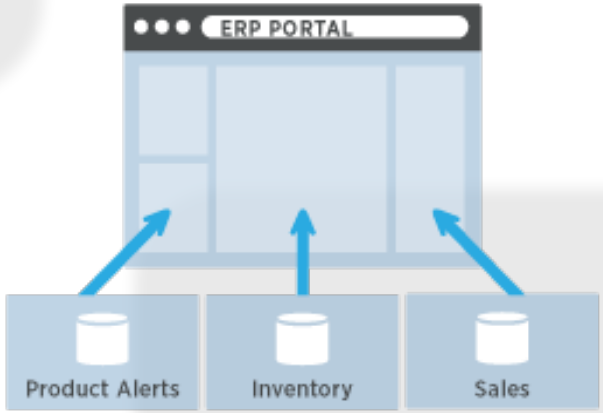


WS-PGRADE Customization: Portlets

- Free, open source enterprise portal

- Java-Based

- Fundamentally constructed of functional units → Portlets



- Support for:
  - Identity Management
  - Content Management System – Web Publishing and Shared Workspaces
  - Role-Based Content Delivery and Role-Based Approvals for Content Update
  - Multiple Languages and Platforms
  - Enterprise Collaboration, Social Networking and Mashups
  - SOA Framework

- Sophisticated API for Developers vs Simple Basic Website Installation and Administration for Common users (no programming skills are required at all)

WS-PGRADE Customization: Portlets

- Customize how the default features work or look
- Deployed as .war files
- Types
  - Portlet (JSR-286)
    - Open Social Gadgets (same as portlets, but no mandatory back-end tech, social applications)
  - Theme → Look
  - Layout → Portlet Arrangement
  - Hook → Customize functionality
  - Ext → Larger flexibility for customization

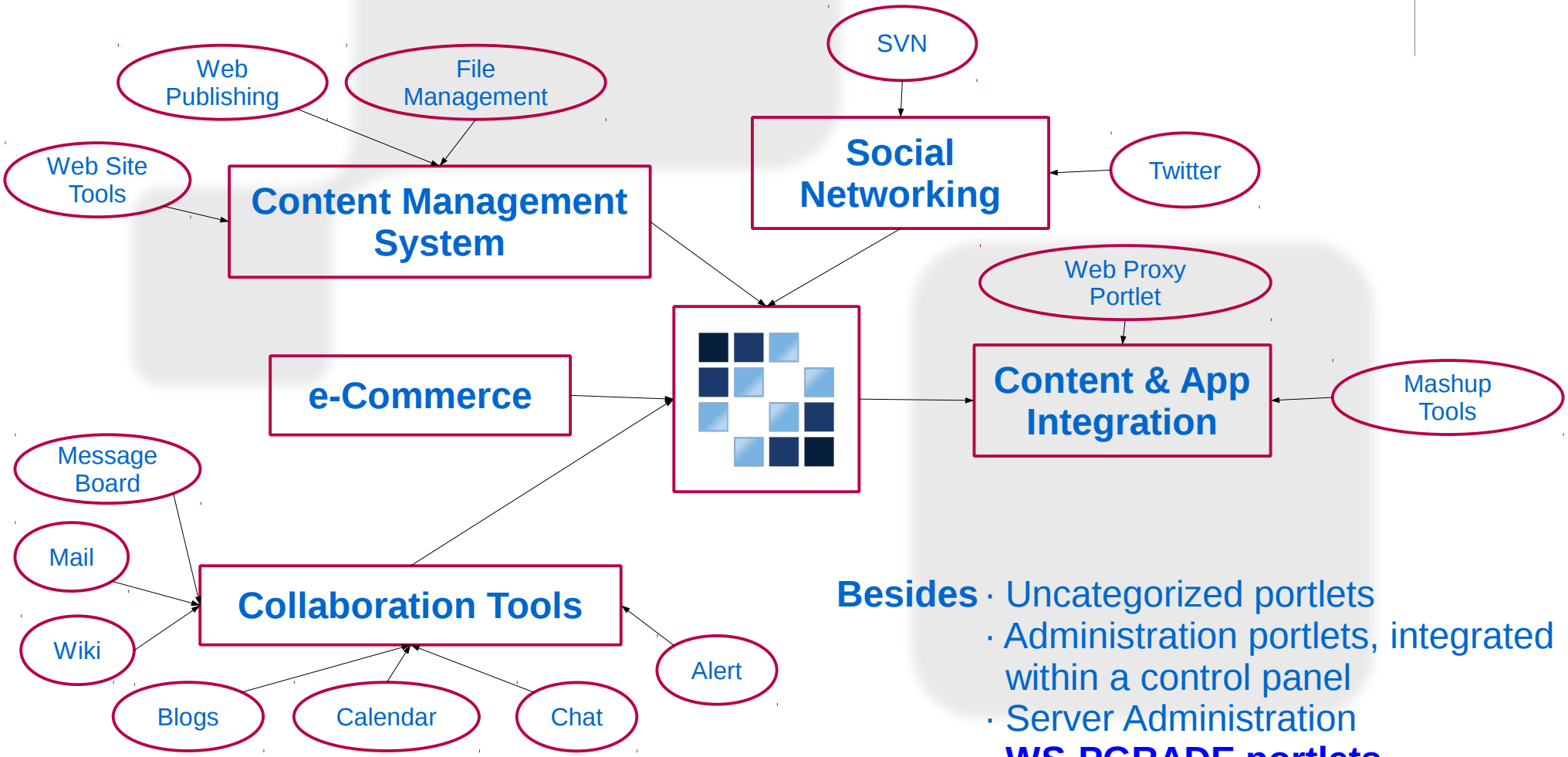
Hot-Deployable

Out of the Box Features

- Reusability

- Public Repository
  - Developer put the ready to run apps in the Repository
  - Common user can get them out from it

WS-PGRADE Customization: Portlets



Besides
- Uncategorized portlets
- Administration portlets, integrated within a control panel
- Server Administration
- **WS-PGRADE portlets**

```
/PORTLET-NAME
    build.xml
    /docroot
        icon.png
        view.jsp
        /css
        /js
        /META-INF
        /WEB-INF
            liferay-display.xml
                Category in Add Menu
            liferay-plugin-package-properties
                To hot deployer
            liferay-portlet.xml
                Liferay enhancements for JSR-286
            portlet.xml    Configuration file for JSR-286
            web.xml    Deployment descriptor file
            /src    Java Source
```

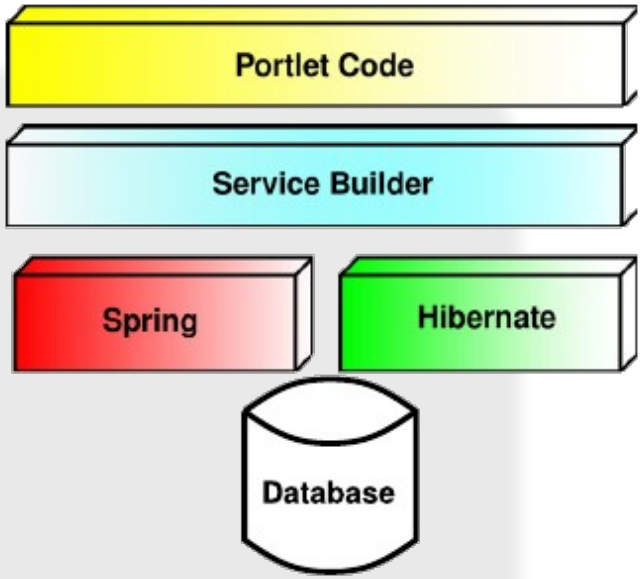
Client Side Files

Liferay-Specific Configuration Files

- Portlets owns just a piece of the page
- Portal must generate the page
  - Do that, rendering the whole page
- Two phases
  - Action Phase
  - Render Phase
- URLs might be generated
  - renderURL
  - actionURL
  - resourceURL → Retrieve other resources
    - AJAX requests

- Database persistence code generator

- Built on top of:
  - Hibernate → Object/Relational mapping
    - DB tables mapped to Java objects
    - Takes care of the persistence
    - Easier integration with more DBs
  - Spring → Dependency Injection



- From DB defintion in `service.xml`, it generates:
  - Hibernate and Spring configuration
  - Finder methods
  - Model layer
  - SQL code for leading databases
  - Data Access Object (DAO) and Data Transfer Objects (DTO)

Two layers for persistence

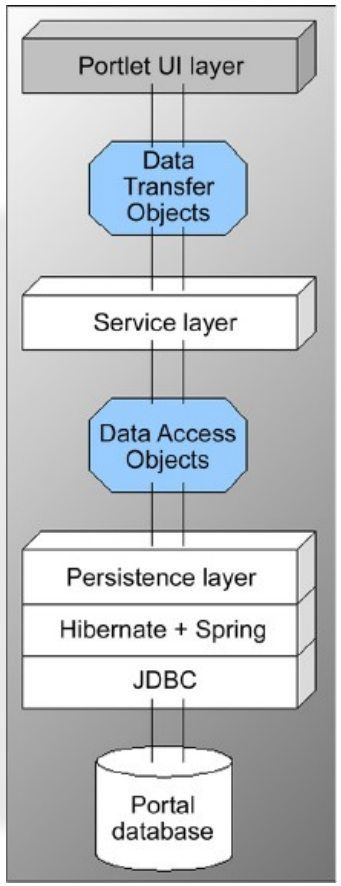
- Separation of concerns

DTO

- Working with objects
  - To be persisted
  - Retrieved from DB
- Buffer between business logic and underlying DB code
- Generated as a stub
- Contains generic methods, not working directly with DB

DAO

- Invoke methods for the actual persisting
- Generated from the `finder` tags in `service.xml`



Events

- General communication mechanism
- Allow portlets to respond on actions not related to a direct user interaction
- Processed at the end of Action Phase
- Then, if an event is returned, portal container calls all portlets listening that event, and the render for the whole portlets afterwards

Public Render Parameters

- Efficient for portlets sharing a set of Render Parameters
  - e.g. Zip code for weather and map portlets

Session Messages (between Action and Render Phase)

How To

- Preparing Action/Render Phase
- Interaction with database
- Using Public Render Parameters
- Using Events

Preparing Action/Render Phase

- Complete the portlet class (Java file)
  - It must extend MVCPortlet
  - Edit portlet-class element in portlet.xml replacing the default one by the name of the package plus the one of the class, e.g. com.bifi.myCoursePortlet
- Complete `view.jsp` which calls the Action Phase
  - Do the call to the Action Phase by means of the portlet element: e.g. `<portlet:actionURL var=”varName” name=”portletMethod” />`
  - Where `portletMethod` is the name given to the method implementing the action which is being called

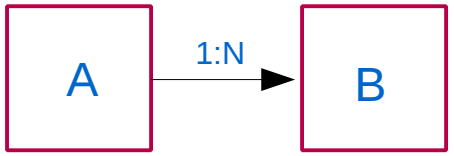
Interaction with Database: Anatomy

```
/PORTLET-NAME
    /docroot
        /META-INF
        /WEB-INF
            service.xml    Database model classes and their attributes
            /service    Interface layer for database management
            /sql    SQL source
            /src
                service.properties    SB properties needed at runtime
                /META-INF    Spring and Hibernate configuration
                /packageName
                    Portlet class source code files
                    /model
                        Base model and base model implementation files
                    /service
                        /base    Abstract classes
                        /impl    Contains-LocalServiceImpl.java files, DTO Layer
                        /persistence    DAO Layer
```

Interaction with Database: Model Definition

- Define model classes in `service.xml`
  - A namespace set the tables off from the rest
  - Entity definition might split Service layer into two:
    - local-service
    - remote-service: Security checks, allow invocations from anywhere
  - Finder methods
    - Finding table records by a field distinct to primary key
    - SB will generate the methods to retrieve that objects

Interaction with Database: Model Definition



- Defining relationships (foreign keys)
  - A adds a `column` element for B, declaring A primary key as the `mapping-key`
  - Add a `column` element in B with A primary key, that is the foreign key needed
  - That does:
    - SB creates Spring configuration injecting B persistence objects in to A DTO classes
    - SB generates a `getB()` in A objects allowing to get all the B records belonging to A

Interaction with Database: Updating Tables

- Consider: Insertion, deletion
  - Update → Get and setters methods
- Corresponding file in `/src/packageName/service/impl` folder
  - Insertion
    - Prepare the object to be inserted (aux)
    - Returns `tableNamePersistence.update(aux, false)`
  - Deletion
    - Method using the object
      - `ResourceLocalService.deleteResource(aux.getCompanyId(), tableName.class.getName(), ResourseConstants.SCOPE_INDIVIDUAL, aux.getPrimayKey())`
      - `tableNamePersistence.remove(aux)`
    - Method using primary key
      - Retrieves the object by the key: `aux = tableNamePersistence.findByPrimaryKey(auxId)`
      - Delete the object: `deleteTableName(aux)`

Interaction with Database: Finders and Queries

- Finders
  - Retrieve records from a table
  - Not able to do joins
- Custom queries
  - Add folder `custom-sql` to `src` and placed inside the file `default.xml`
  - Write your SQL query
  - Provide an identifier for the query in a `sql` element within `default.xml`
  - Wrap the query in the `sql` element
- Create the finder
  - Place the new file in `/src/packageName/service/persistence`
  - Instantiate the id of the query
  - Implement the method for the SQL lookup (further info, look for Java Hibernate)
  - Add a method to `tableNameLocalServiceImpl.java` to retrieve the results

Using Public Render Parameters

- Declare Render Parameters to be shared in `portlet.xml`

  `public-render-parameter`
- Specify Render Parameters a portlet would like to share within `<portlet>` element

  `supported-public-render-parameter`
- Set Render Parameters in `processAction` method

  `actionResponse.setRenderParameters`
- Get Render Parameters in Render Phase

  `renderRequest.getParameter`

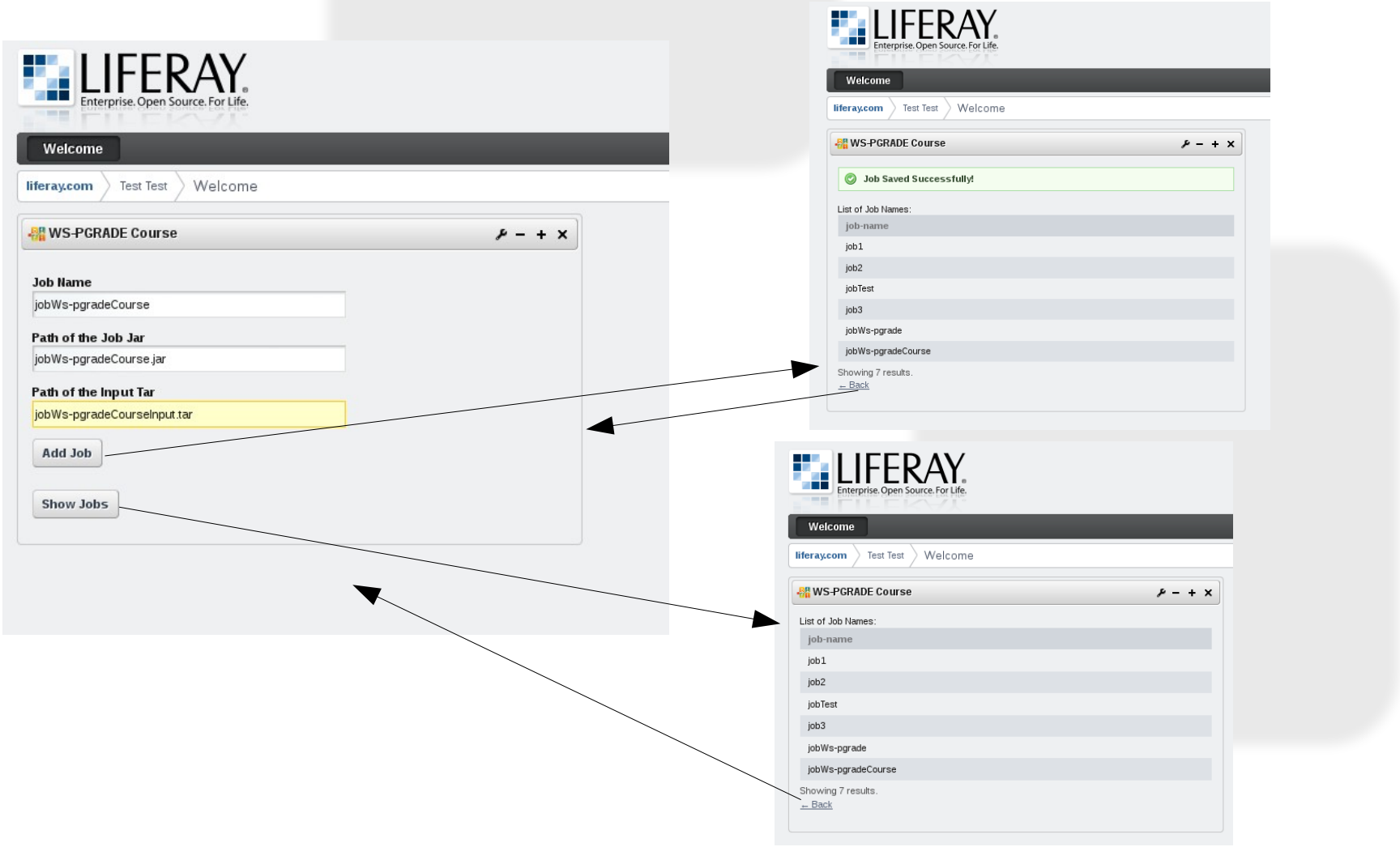
Using Events

- Declare events to be published/processed in `portlet.xml` of the corresponding portlet

  `supporting-publishing-event`

  `supporting-processing-event`
- Publishing portlet issues an event through `setEvent`
  - Event will be processed after Action Phase
- Processing portlet must implement `processEvent`

WS-PGRADE Customization: Portlets



WS-PGRADE Customization: Portlets

view.jsp

```
<portlet:actionURL name="addJob" var="addJobURL"/>

  <aui:form name="aj" action="<%= addJobURL.toString() %>"
method="post">

    <aui:fieldset>

      <aui:input name="jobName" label="Job Name" size="45" />
      <aui:input name="jobJar" label="Path of the Job Jar" size="45" />
      <aui:input name="inputTar" label="Path of the Input Tar" size="45" />

      <aui:button-row>
        <aui:button type="submit" value="Add Job"/>
      </aui:button-row>

    </aui:fieldset>

  </aui:form>

<portlet:renderURL var="showJobsURL">
            <portlet:param name="jspPage" value="/list.jsp" />
</portlet:renderURL>

<aui:form name="sj" action="<%= showJobsURL.toString() %>"
method="post">
            <aui:button-row>
            <aui:button type="submit" value="Show Jobs"/>
            </aui:button-row>
</aui:form>
```

list.jsp

```
<portlet:renderURL var="viewURL">
            <portlet:param name="jspPage" value="/view.jsp" />
</portlet:renderURL>

<liferay-ui:success key="job-saved-successfully" message="Job Saved Successfully!" />

List of Job Names:

<liferay-ui:search-container delta="10" emptyResultsMessage="no-jobs-were-found">
            <liferay-ui:search-container-results
                        results="<%= ActionUtil.getJobs(renderRequest) %>"
                        total="<%= JobLocalServiceUtil.getJobsCount() %>"
            />

            <liferay-ui:search-container-row
                        className="com.dbuse.model.Job"
                        keyProperty="jobId"
                        modelVar="job"
            >
                        <liferay-ui:search-container-column-text
                                    name="job-name"
                                    property="jobName"
                        />
            </liferay-ui:search-container-row>

            <liferay-ui:search-iterator />

</liferay-ui:search-container>

<p><a href="<%= viewURL %>">&larr; Back</a></p>
```

WS-PGRADE Customization: Portlets

JobAdminPortlet.java

```
public class JobAdminPortlet extends MVCPortlet {

    public void addJob(ActionRequest request, ActionResponse response)
      throws Exception {

      ThemeDisplay themeDisplay =
         (ThemeDisplay) request.getAttribute(WebKeys.THEME_DISPLAY);
      Job job = ActionUtil.jobFromRequest(request);

      JobLocalServiceUtil.addJob(
            job, themeDisplay.getUserId());
      SessionMessages.add(request, "job-saved-successfully");

      response.setRenderParameter("jspPage", "/list.jsp");
   }
}
```

JobLocalServiceImpl.java

```
public class JobLocalServiceImpl extends JobLocalServiceBaseImpl {

    public Job addJob (Job newJob, long userId) throws SystemException,
                    PortalException {

            long jobId =
                    counterLocalService.increment(Job.class.getName());

            Job job = jobPersistence.create(jobId);

            job.setJobName(newJob.getJobName());
            job.setJobJar(newJob.getJobJar());
            job.setInputTar(newJob.getInputTar());
            job.setCompanyId(newJob.getCompanyId());
            job.setGroupId(newJob.getGroupId());

            return jobPersistence.update(job, false);

    }
}
```

service.xml

```
<service-builder package-path="com.dbuse">
        <author>Eduardo Lostal</author>
        <namespace>dbuse</namespace>

        <entity name="Job" local-service="true" remote-service="false">
                <column name="jobId" type="long" primary="true" />
                <column name="jobName" type="String" />
                <column name="jobJar" type="String" />
                <column name="inputTar" type="String" />
                <column name="companyId" type="long" />
                <column name="groupId" type="long" />

                <order by="asc">
                        <order-column name="jobId" />
                </order>

                <finder name="G_JN" return-type="Collection">
                        <finder-column name="groupId" />
                        <finder-column name="jobName" />
                </finder>

                <finder name="GroupId" return-type="Collection">
                        <finder-column name="groupId" />
                </finder>

                <finder name="CompanyId" return-type="Collection">
                        <finder-column name="companyId" />
                </finder>
        </entity>
</service-builder>
```

WS-PGRADE Customization: Portlets

- Liferay Portal is a very flexible platform
- Portlets are a useful mechanism to add functionality to a page
- WS-PGRADE contains, in the shape of portlets, tools to deal with workflows and middleware
- Use of WS-PGRADE makes easier working with DCIs for the end-user, allowing an easy communication among the applications involved in the process
- That is possible since it is built upon Liferay Technology which eases the communication among the porlets

Thank you for your attention!

References

- MTA SZTAKI LPDS, WS-PGRADE Portal User Manual, Budapest, Hungary, 2012
- Rich Sezov, Liferay In Action: The Official Guide to Liferay Portal Development, Manning Publications, 2011
- Liferay, Inc., Liferay Developer's Guide, 2011
- Stefan Hepper, JSR-286 Porlet Specification 2.0, 2008
- Deepak Gothe, Understanding the Java Portlet Specification 2.0 (JSR 286), 2010
- SCI-BUS, http://www.sci-bus.eu/home
- WS-PGRADE, https://guse.sztaki.hu/liferay-portal-6.0.5/
- GUSE, http://www.guse.hu/
- Liferay, http://www.liferay.com/