

XP Lab 2007, Nanjing, China

JBoss Seam

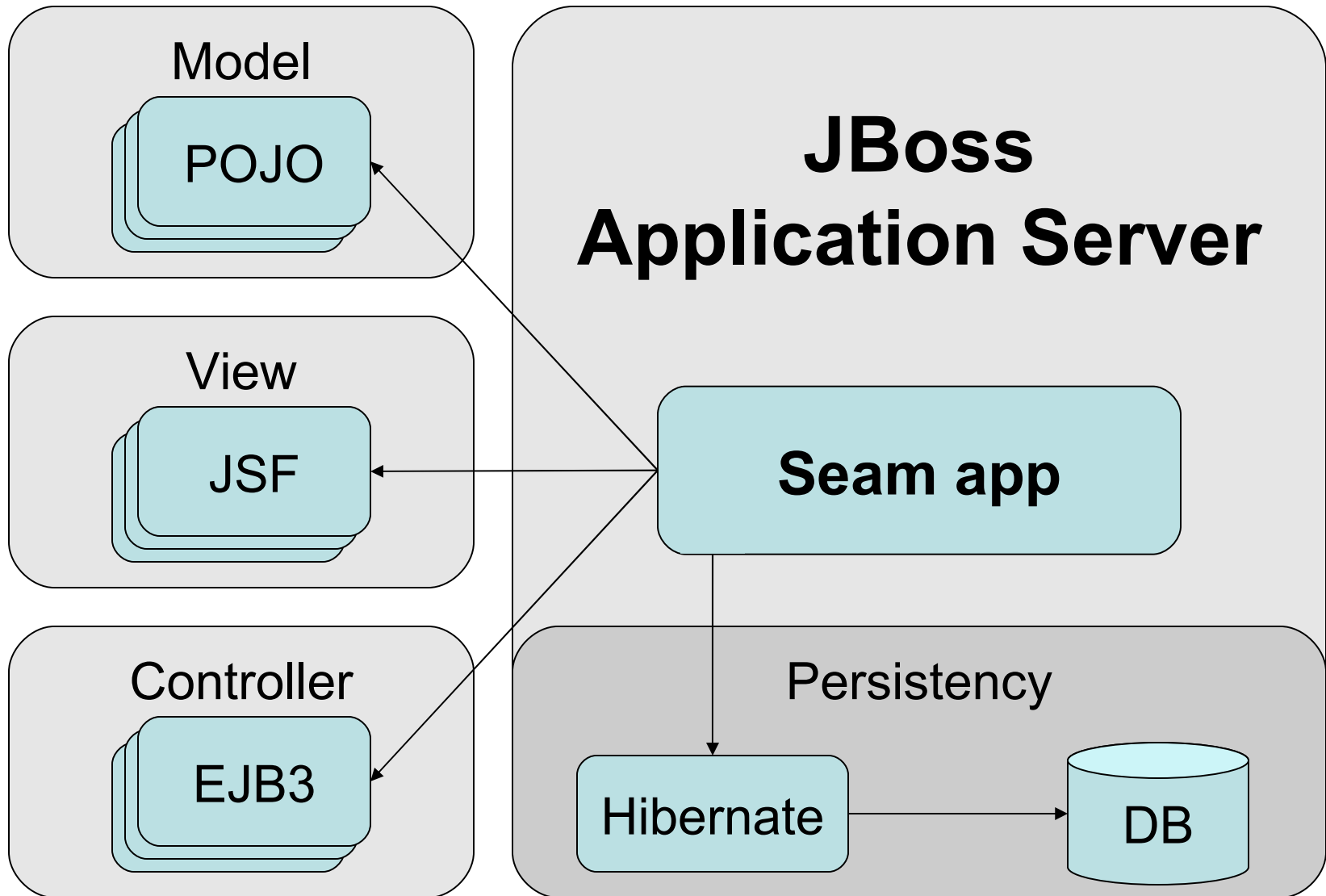
A quick introduction

Mark Schmatz

JBoss Seam - some key facts

- Web application framework
- Predecessor of Struts
- Combines JavaServer Faces (JSF) and Enterprise Java Beans 3.0 (EJB3)
- Page-based (Struts is action-based)
- Almost no need to edit config files
 - Configuration is done by Java annotations
 - “Configuration by exception”
- Applications run in JBoss Application Server (AS)
 - Struts apps are deployed as web archives (WAR files)
 - Seam apps with EJB support are deployed as Enterprise archives (EAR files)
- Database schema is automatically built from the Model
 - Complete abstraction from database

Bird's eye view



Seam beans

- Every class can be a Seam bean
- Just annotate the class with `@Name`
- Seam beans are accessible
 - from JSF pages via EL expressions
(EL = Expression language)
 - from other classes by injection

→ Example

Seam bean example

```
@Name("testBean")
public class MyTest {

    private String name = "Mark";

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

JSF page accessing our Seam bean

```
<!-- ... -->
```

```
<s:div>
```

```
Hello, my name is:
```

```
<h:outputText value="#{testBean.name}" />
```

```
This is a static output:
```

```
<h:outputText value="42" />
```

```
</s:div>
```

```
<!-- ... -->
```

EL expression

Bean property

Bean name

JavaServer Faces - Some basic UI components

- Input text field:

```
<h:inputText  
    id="myId"      // always give a unique id  
    required="true" // if true then the field must not be empty  
    value="#{testBean.name}" // the entered text  
    rendered="true" /> // if false it will not be rendered
```
- Output text:

```
<h:outputText value="#{testBean.name}" />
```
- Text area:

```
<h:inputTextarea cols="..." rows="..." ... />
```
- Secret input field:

```
<h:inputSecret ... />
```

JavaServer Faces - Some basic UI components

- Selectbox with only one selectable element:

```
<h:selectOneListbox value="#{myBean.selectedUser}">  
    <f:selectItems value="#{myBean.users}" />  
</h:selectOneListbox>
```
- Selectbox with only many selectable element:

```
<h:selectManyListbox value="#{myBean.selectedUsers}">  
    <f:selectItems value="#{myBean.users}" />  
</h:selectManyListbox>
```

Note:

instead of `<f:selectItems ... />` you can also use multiple `<f:selectItem itemLabel="..." itemValue="..." />`

JavaServer Faces - Some basic UI components

- Hyperlink with request parameter:

```
<s:link id="userLink"
  value="#{user.name}" // Rendered text
  view="/user.xhtml"> // Destination view

// submit the user id as request parameter
// which is named 'userId'

<f:param name="userId"
  value="#{user.id}"/>

</s:link>
```

JavaServer Faces - Some basic UI components

- Data table (renders a table populated with the elements of e.g. a Java collection):

```
<h:dataTable id="userList" var="user"
  value="#{userList.resultList}"
  rendered="#{not empty userList.resultList}">

  <h:column>
    <f:facet name="header">Id</f:facet>
    #{user.id}
  </h:column>
  <h:column>
    <f:facet name="header">Name</f:facet>
    #{user.name}
  </h:column>

</h:dataTable>
```

JavaServer Faces - Summary

- JSF offers UI components like in Swing
 - higher abstraction, easier to use
- Names differ from known HTML tags and have namespaces like e.g. **h:** of **f:**
- You can easily read and write properties of Seam beans
 - even complex types and collections

Online documentation:

- JSF Tutorial: <http://www.coreservlets.com/JSF-Tutorial/>
- JSF tags: <http://horstmann.com/corejsf/jsf-tags.html>
- JSF tag library API:
http://java.sun.com/javaee/javaserverfaces/1.1_01/docs/tld/docs/index.html

Enterprise Java Beans (EJB)

- EJBs implement business behavior
- Also accessible from JSF pages and other classes
 - because they are annotated with `@Name` as well
- Great built-in support for persistency and transactions
- EJBs are very performing because of caching and pooling
- Middleware concerns are handled by the app server
 - but you can also do the transaction stuff manually if you want
- There are two kinds of EJBs: **stateless** and **stateful**
 - Just annotate a Java class with either
 - `@Stateless` or
 - `@Stateful`

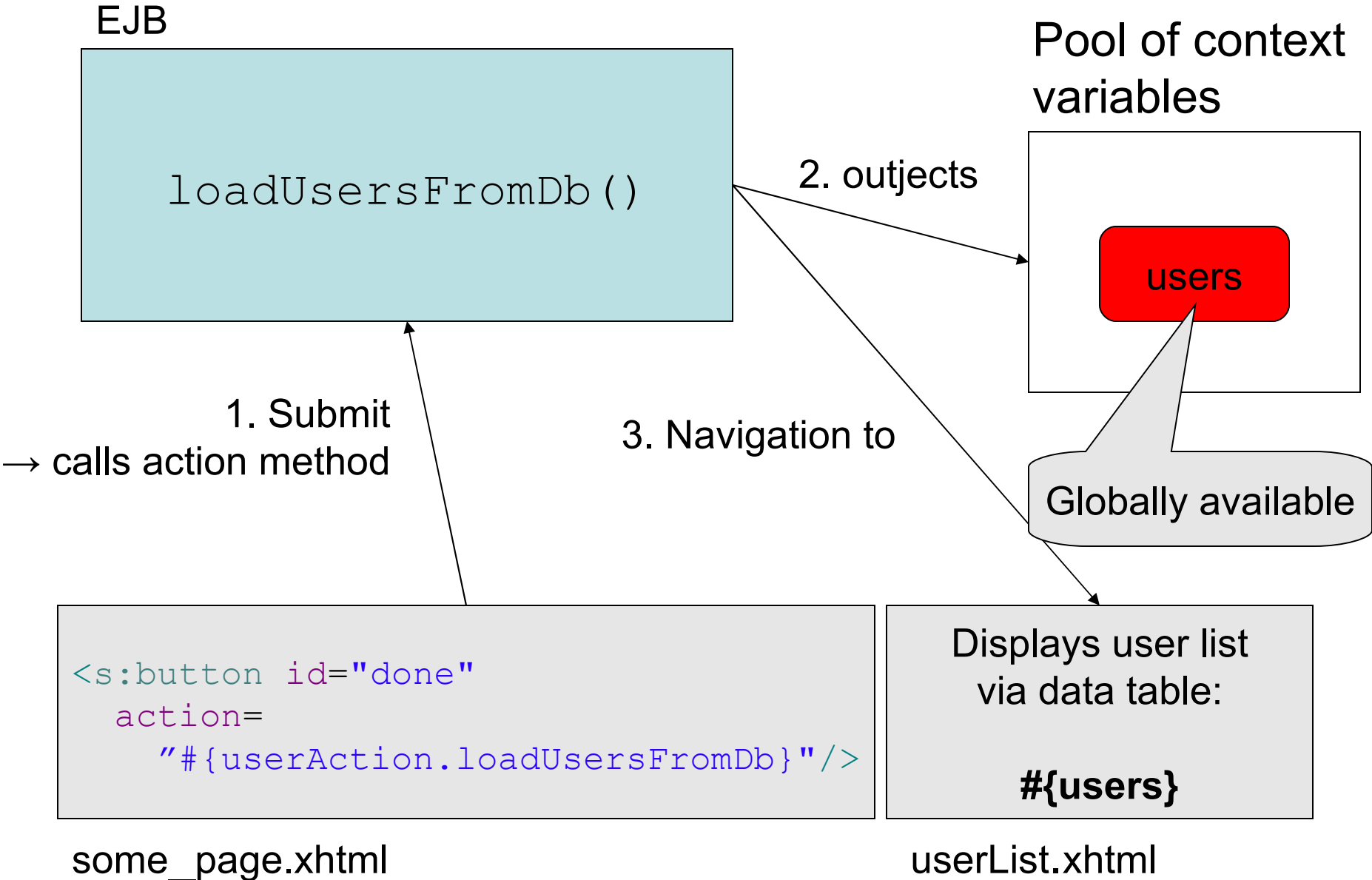
Enterprise Java Beans

```
@Stateful // <- UserAction is a stateful EJB
@Name("userAction") // <- The name of the EJB
public class UserAction {
    @Out
    private List<User> users;
    // via the @Out annotation the users list is
    // read- and writable from pages and other classes

    public List<User> getUsers() { ... }
    public void setUsers(List<User> users) {...}

    // This is an action method.
    // Return value is for page navigation.
    public String loadUsersFromDb() {
        users = em.createQuery("from User").getResultList();
        return "/userList.xhtml";
    }
}
```

Enterprise Java Beans - Action beans navigate to pages



Enterprise Java Beans - Context variables can be injected

Another EJB

```
public class AnotherEJB {  
  
    @In  
    private List<User> users;  
  
    // ...  
}
```

Pool of context variables

injects

users



Bijection

- Injection and Outjection (short Bijection)
 - is a core feature of Seam
 - is done **before every** method call
 - Bijection can be used within EJBs and Seam beans

- Injection assigns the reference to the resp. bean into the current bean
 - You don't need to get it yourself
 - Also known as Inversion of Control (IoC)

Scopes - How long does a bean live?

- Seam beans and EJBs live in a scope
- When the scope is left then the bean will be destroyed
- Scope types are:
 - **Event** - similar to the request scope known from HTTP
 - **Page** - like Event but the bean remains available if the requested page is the same (Callback)
 - **Session** - known from HTTP
 - **Application** - known from HTTP
 - **Conversation** - Scope is explicitly set
 - A bean will be created when one of its `@Begin`-annotated methods is called
 - A call to a `@End`-annotated method destroys the enclosing bean

Entities

- Java beans annotated with `@Entity` are entities
- They don't have a `@Name` annotation
- They represent the data to be persisted
- Seam cares about the creation of tables in the database!
 - You only need to create the database and a user/role
- Relations between Entities are defined via annotations
 - `@OneToOne`, `@OneToMany`, `@ManyToOne`, `@ManyToMany`
- Always annotate the getters!
 - you also can annotate fields but this isn't useful in most cases

Entity example

```
@Entity
@Table(name="_user") // with '_' because 'user' is reserved
public class User {
    private Long id; // Mandatory! Primary key for DB
    private String name;
    private Date birthday;
    private List<Role> roles;

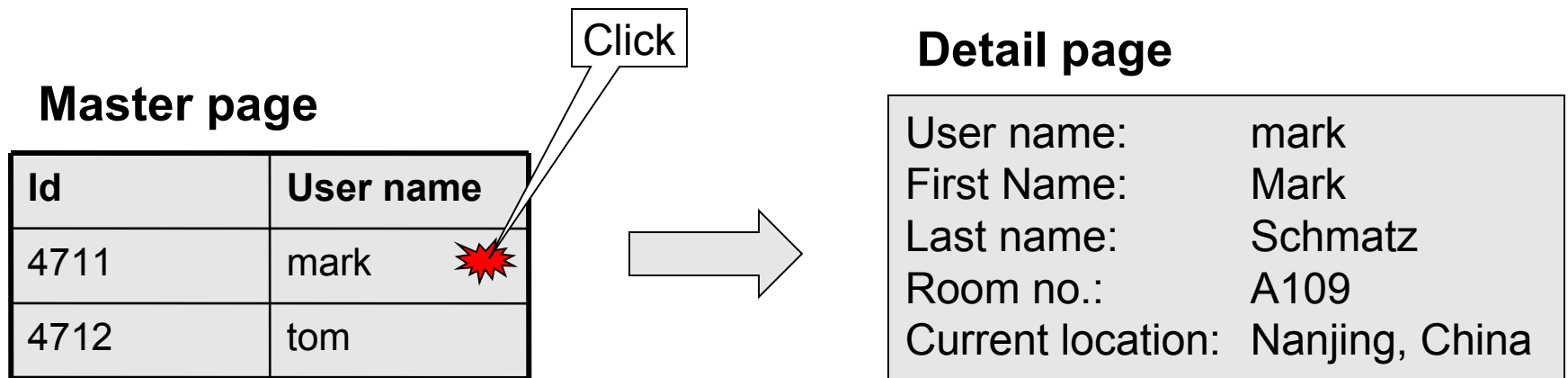
    @Id @GeneratedValue
    public Long getId() { ... }
    public void setId(Long id) { ... }

    @ManyToMany
    @JoinTable(
        name="t_user_role",
        joinColumns={@JoinColumn(name="fk_user")},
        inverseJoinColumns={@JoinColumn(name="fk_role")}
    )
    public List<Role> getRoles() { ... }
    public void setRoles(List<Role>) { ... }

    // The other getters and setters...
}
```

QueryObject / HomeObject

- QueryObject
 - provides the means to query the database to get a list of entities
 - used by the **master** page
- HomeObject
 - provides a wrapper for an entity which enables CRUD
 - used by the **detail** page



A last word

- This was a very incomplete overview of Seam
- Please refer to the reference manual for detailed information!
- At least you must have a deeper look at:
 - JSF
 - Persistence / Transactions (Seam managed / manually)
 - Hibernate annotations
 - Hibernate EntityManager
 - (Form) validation
 - Conversations
 - but you probably don't need them at the beginning
 - Expression language (the `#{...}` expressions)
 - Home objects / Query objects

Seam reference documentation

Your main source of information:

- docs.jboss.com/seam/1.2.1.GA/reference/en/html_single/

All docs are available at your FTP server!