

Transitioning To Seam

Brian Leonard
Sun Microsystems, Inc.

Agenda

- (Brief) Introduction to Java EE 5
 - JSF 1.2
 - EJB 3.0
 - JPA
 - Interceptors
- Introduction to Seam
- Refactor to use the Seam Framework
- Seam Portability
- Summary and Resources



Agenda

- (Brief) Introduction to Java EE
 - JSF 1.2
 - EJB 3.0
 - JPA
 - Interceptors
- Introduction to Seam
- Refactor to use the Seam Framework
- Seam Portability
- Summary and Resources



Java EE 5 Goal

Make it easier to develop Java EE applications

Especially when first getting started with Java EE



How Was It Made Easier?

Then

- deployment descriptors
- required container interfaces
- JNDI Lookups
- configuration files and command line options
- no supported UI framework



Now

- Java language annotations
- Plain Old Java Objects (POJOs)
- dependency injection
- more and better defaults
- Java Server Faces (JSF)

Are Java Server Faces, which are now part of the Java EE specification, EJB aware?

How Much Easier Is It?

RosterApp¹

	J2EE 1.4	Java EE 5
Number of Java files	17	7
Lines of code (Java)	987	716
Number of XML Files	9	1
Lines of code (XML)	792	5

[1] Source: Raghu Kodali,
Oracle

Annotations Everywhere in Java EE 5

- For defining and using web services
 - @WebService
- To greatly simplify EJB development
 - @Stateless, @Stateful, @MessageDriven
- To map Java classes to databases
 - @Entity, @Table, @Column
- To specify external dependencies
 - @Resource
- Reduces need for deployment descriptors and JNDI lookups

For Example - J2EE 1.4 Web Service

```

package endpoint;

import java.rmi.*;

public class HelloServiceImpl
    implements HelloServiceSEI {

    public String sayHello(String param)
        throws java.rmi.RemoteException {
        return "Hello " + param;
    }
}

package endpoint;

import java.rmi.*;

public interface HelloServiceSEI
    extends java.rmi.Remote {
    public String sayHello(String param)
        throws java.rmi.RemoteException;
}

```

```

<?xml version='1.0' encoding='UTF-8' ?>
<webservices xmlns='http://java.sun.com/xml/ns/j2ee'
version='1.1'>
  <webservice-description>
    <webservice-description-name>
      HelloService</webservice-description-name>
    <wsdl-file>
      WEB-INF/wsdl/HelloService.wsdl</wsdl-file>
    <jaxrpc-mapping-file>
      WEB-INF/HelloService-mapping.xml
    </jaxrpc-mapping-file>
    <port-component xmlns:wsdl-
port_ns='urn:HelloService/wsdl'>
      <port-component-name>HelloService</port-component-name>
      <wsdl-port>wsdl-port_ns:HelloServiceSEIPort</wsdl-port>
      <service-endpoint-interface>
        endpoint.HelloServiceSEI</service-endpoint-interface>
      <service-impl-bean>
        <servlet-link>WSServlet_HelloService</servlet-link>
      </service-impl-bean>
    </port-component>
  </webservice-description>
</webservices>

```

```

<?xml version='1.0' encoding='UTF-8' ?>
<configuration
xmlns='http://java.sun.com/xml/ns/jax-rpc/ri/config'>
  <service name='HelloService'
    targetNamespace='urn:HelloService/wsdl'
    typeNamespace='urn:HelloService/types'
    packageName='endpoint'>
    <interface name='endpoint.HelloServiceSEI'
      servantName='endpoint.HelloServiceImpl'>
    </interface>
  </service>
</configuration>

```

Java EE 5 Web Service

```

package endpoint;

import javax.jws.WebService;

@WebService
public class Hello {

    public String sayHello(String param) {
        return "Hello " + param;
    }
}

```

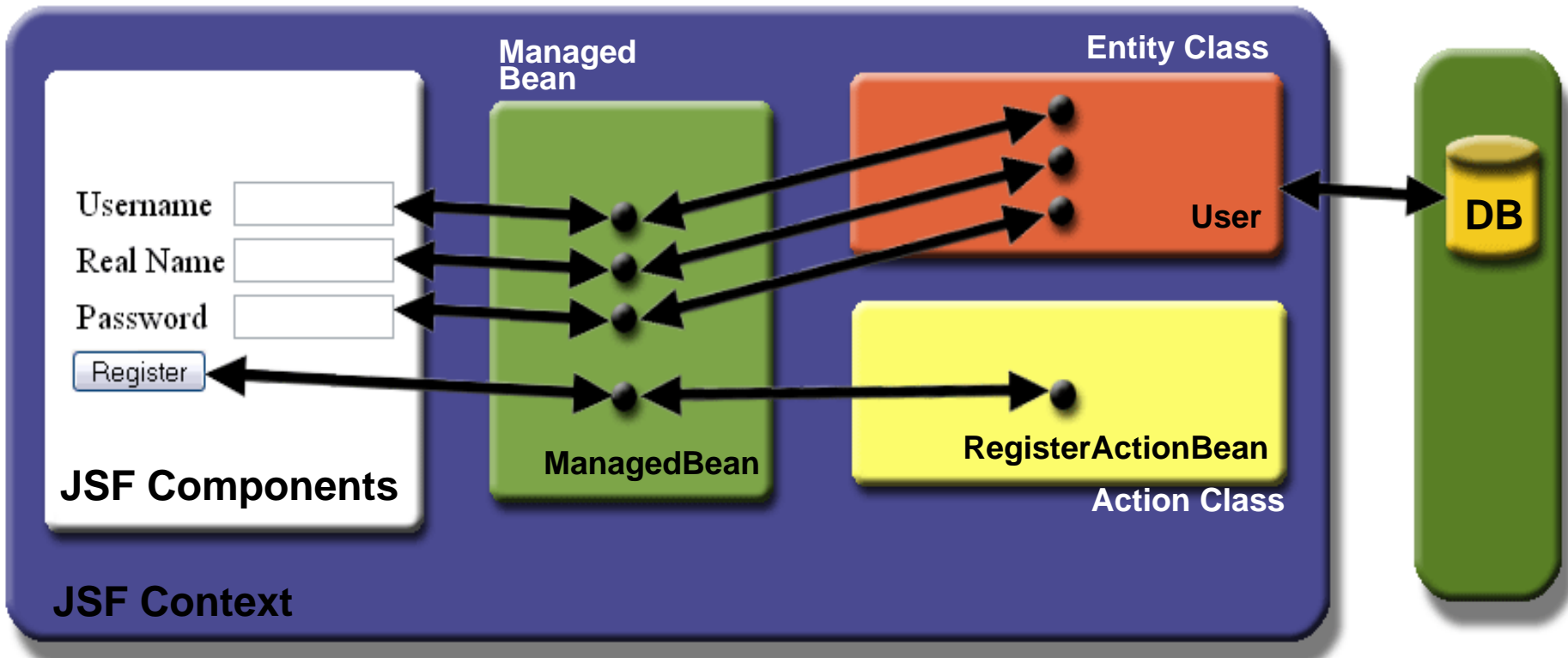


Java EE 5 Programming Model

The screenshot shows a Mozilla Firefox browser window titled "Register New User - Mozilla Firefox". The address bar displays the URL "http://localhost:8080/Registration-WebModule/faces/index.jsp". The form contains the following fields and a button:

Username	<input type="text" value="bleonard"/>
Real Name	<input type="text" value="Brian Leonard"/>
Password	<input type="password" value="*****"/>
<input type="button" value="Register"/>	

Java EE 5 Programming Model





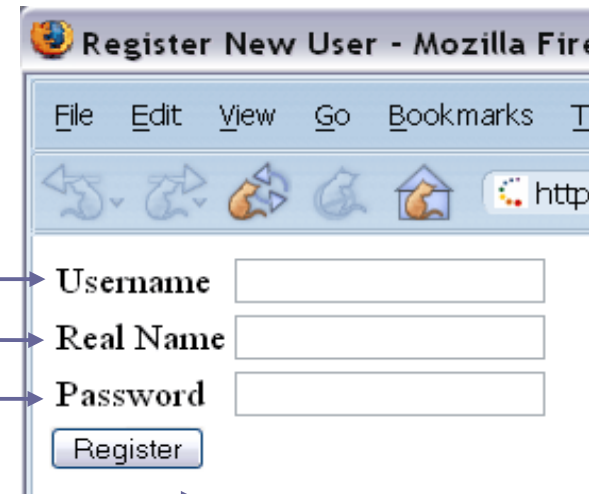
register.jsp: JSF In Action

```
<td>Username</td>
<td><h:inputText id="userName"
                value="#{user.username}"
                required="true">
    <f:validateLength minimum="5"
                    maximum="15" />
```

```
<td>Real Name</td>
<td><h:inputText id="name"
                value="#{user.name}"
                required="true">
```

```
<td>Password</td>
<td><h:inputSecret id="password"
                  value="#{user.password}"
                  required="true">
    <f:validateLength minimum="5"
                    maximum="15" />
```

```
<h:commandButton id="registerCommand"
                 type="submit" value="Register"
                 action="#{user.register}" />
```



A JSF Validator



register.jsp & BackingBean

Register New User - Mozilla Firefox

File Edit View Go Bookmarks

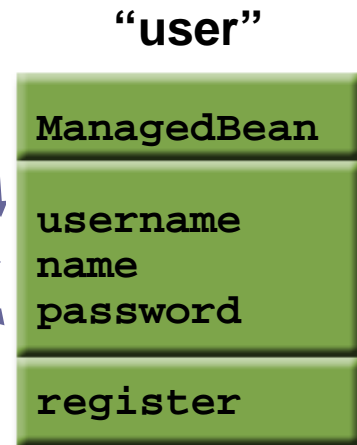
Username

Real Name

Password

```

1. ...
2. <td>Username</td>
3. <td><h:inputText
4.     id="userName"
5.     value="#{user.username}"
6. ...
7. <td>Real Name</td>
8. <td><h:inputText
9.     id="name"
10.    value="#{user.name}"
11. ...
12. <td>Password</td>
13. <td><h:inputSecret
14.     id="password"
15.     value="#{user.password}"
16. ...
17. <h:commandButton
18.     id="registerCommand"
19.     type="submit" value="Register"
20.     action="#{user.register}" />
21. ...
    
```





Managed Beans Configuration

faces-config.xml

1. ...
2. <managed-bean>
3. <managed-bean-name>user</managed-bean-name>
4. <managed-bean-class>
5. org.examples.jsf.ManagedBean
6. </managed-bean-class>
7. <managed-bean-scope>request</managed-bean-scope>
8. </managed-bean>
9. ...

The name of the contextual variable we will refer to in the JSF EL

The component scope



Managed Bean

```
public class ManagedBean {
```

```
    private String username;
    private String name;
    private String password;
```

Properties bound to controls via the value bindings

```
    public String getUsername() { return username; }
    public String setUsername(String username) { this.username=username; }
    ...
```

```
    private RegisterActionLocal registerActionBean;
    private InitialContext ctx;
    {
```

```
        try {
            ctx = new InitialContext();
            registerActionBean = (RegisterActionLocal)
ctx.lookup("registration/RegisterActionBean/local");
        } catch (NamingException ex) {
            ex.printStackTrace();
        }
    }
```

JNDI Lookup of Session Bean

register action listener method bound to control via the JSF method bindings

```
    public String register() {
        return registerActionBean.register(username, name, password);
    }
}
```

EJB 3.0

- Dramatic simplification of all bean types
- Regular Java classes (POJO)
 - `@Stateless`, `@Stateful`, `@MessageDriven` annotations
 - Use standard interface inheritance
- Dependency injection
 - Instead of using JNDI API to location components and resources, let the container fetch them for you.
- Interceptors
- Entity Beans (CMP) replaced with JPA

Java Persistence API (JPA)

- Single persistence API for Java EE **AND** Java SE
- Much simpler than EJB CMP
- At least three implementations (all open source):
 - Oracle – GlassFish/TopLink Essentials
 - JBoss – Hibernate
 - BEA – Kodo/OpenJPA
- Configured via persistence.xml
- Feedback is overwhelmingly positive



JPA – Object Relational Mapping

- Developer works with objects
 - Database queries return objects
 - Object changes persist to the database
- Data transformation is handled by the persistence provider (TopLink, Hibernate, etc.)
- Annotations define how to map objects to tables
 - **@Entity** marks a regular Java class as an entity.
 - Class attributes map to table columns. Can be customized with **@Column**.
 - Manage relationships: **@OneToMany**, ...

jug JPA – Entity Manager

- EntityManager stores/retrieves data
- Inject EntityManager:
 - `@PersistenceContext private EntityManager em;`
- Create an instance of the entity:
 - `User u = new User(params);`
- Use EntityManager methods to persist data
 - `em.persist(u); em.merge(u); em.delete(u);`
- Query using EJB QL or SQL
 - `User u = em.find(User.class, param);`

Our Action Bean

```
1. @Stateless
2. public class RegisterAction implements Register{
3.     @PersistenceContext
4.     private EntityManager em;
5.
6.     public String register(String username, String name,
7. String password){
8.         List existing = em.createQuery("select username
9.         from User where username=:username")
10.        .setParameter("username",username)
11.        .getResultList();
12.        if (existing.size()==0){ // Create a new user
13.            User user = new User(username, name, password)
14.            em.persist(user);
15.            return "success";
16.        } else {
17.            FacesContext facesContext =
18.            FacesContext.getCurrentInstance();
19.            FacesMessage message = new
20.            FacesMessage(username + " already exists");
21.            facesContext.addMessage(null, message);
22.            return null;
23.        }
24.    }
25. }
```

Agenda

- (Brief) Introduction to Java EE 5
 - JSF 1.2
 - EJB 3.0
 - JPA
 - Interceptors
- **Introduction to Seam**
- Refactor to use the Seam Framework
- Seam Portability
- Summary and resources

- RAD programming model for data-driven applications without sacrificing the full power of Java EE 5
- Framework for integrating JSF and EJB3 component models.
 - Bridge web-tier and EJB tier session contexts
 - Enable EJB 3.0 components to be used as JSF managed beans.
- Prototype for JSR 299: Web Beans

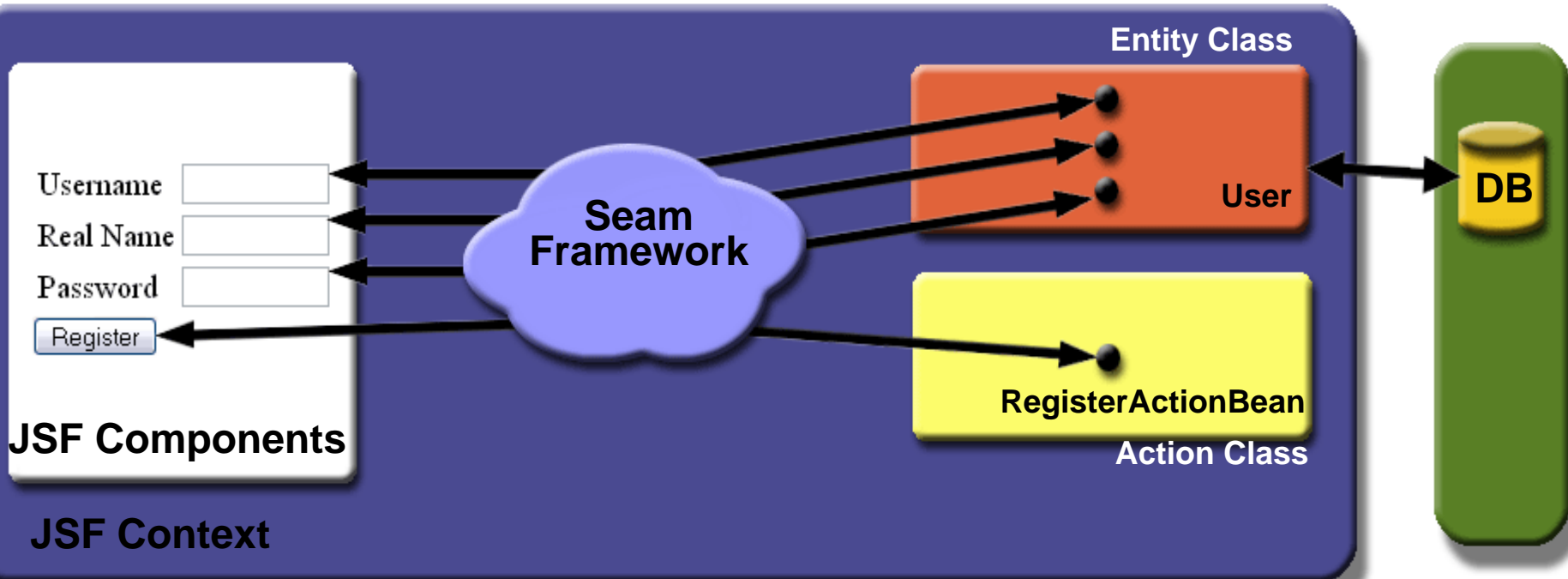
JBoss Seam

- Some Key Concepts:
 - Eliminate the ManagedBean – bind directly to our entity and action classes.
 - Enhanced context model
 - Conversation
 - Business process
 - Depend less on xml (faces-config) – use annotations instead
 - ***Bijection*** – for stateful components - dynamic, contextual, bidirectional
 - Constraints specified on the model, not in the view.

Agenda

- (Brief) Introduction to Java EE 5
 - JSF 1.2
 - EJB 3.0
 - JPA
 - Interceptors
- Introduction to Seam
- Refactor to use the Seam Framework
- Seam Portability
- Summary and resources

Seam Registration Application



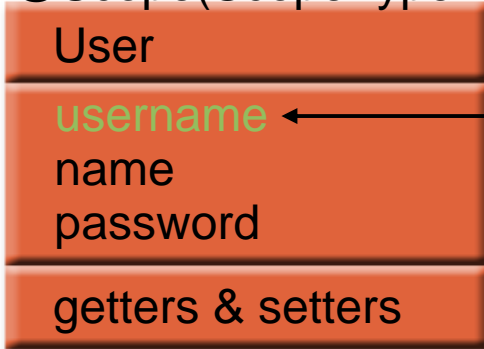
Additions...

- EJB Module (jar)
 - Include jboss-seam.jar
 - seam.properties
- Web Module (war)
 - faces-config.xml
 - SeamPhaseListener
 - web.xml
 - JndiPattern
 - SeamListener



Eliminating Your ManagedBean

1. `@Name("user")`
`@Scope(ScopeType.EVENT)`



2. `@Name("register")`



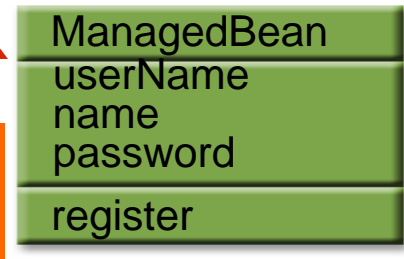
3. register.jsp

```
<h:inputText id="userName" value="#{user.username}" ...
<h:commandButton ... action="#{register.register}"/> ...
```

4. faces-config.xml

```
<managed-bean-name>user</...
<managed-bean-class>org.ex...
```

GONE!!!





Component name for Seam

User.java (1 of 2)

```
1. @Entity
2. @Name("user")
3. @Scope(ScopeType.Event)
4. @Table(name="users")
5. public class User implements Serializable{

1.     private String username;
2.     private String password;
3.     private String name;
4.
5.     public User(String name, String password,
6.                 String username){
7.         this.name = name;
8.         this.password = password;
9.         this.username = username;
10.    }...
```

Component context

User.java (2 of 2)

```
1.     public User() {}
2.
3.     @Length(min=5, max=15)
4.     public String getPassword() {
5.         return password;
6.     }
7.
8.     public String getName() {
9.         return name;
10.    }
11.
12.    @Id @Length(min=5, max=15)
13.    public String getUsername() {
14.        return username;
15.    }
```

Hibernate Validator
Framework

RegisterAction.java

Attribute injected
by Seam

```
1. @Stateless
2. @Name("register")
3. public class RegisterAction implements Register{
4.     @In
5.     private User user;
6.     @PersistenceContext
7.     private EntityManager em;
8.
9.     public String register(){
10.         List existing = em.createQuery("select username
11.             from User where username=:username")
12.             .setParameter("username",user.getUsername())
13.             .getResultList();
14.         if (existing.size()==0){
15.             em.persist(user);
16.             return "success";
17.         }else{
18.             FacesMessages.instance().add("User
19.                 #{user.username} already exists");
20.             return null;
21.         }
    }
```

Agenda

- (Brief) Introduction to Java EE 5
 - JSF 1.2
 - EJB 3.0
 - JPA
 - Interceptors
- Introduction to Seam
- Refactor to use the Seam Framework
- **Seam Portability**
- Summary and Resources



- Add missing JBoss Libraries
- Change Persistence Unit to TopLink
- Update web.xml:
 - JndiPattern: java:comp/env/
 - Delete MyFaces context listener
 - RegisterActionEJB reference

Agenda

- (Brief) Introduction to Java EE 5
 - JSF 1.2
 - EJB 3.0
 - JPA
 - Interceptors
- Introduction to Seam
- Refactor to use the Seam Framework
- Seam Portability
- **Summary and Resources**

Summary

- Hopefully you've learned how to start using the Seam framework in your existing JSF / EJB 3.0 applications.
- There's much more to Seam, I've just touched the surface.



Resources

- Java EE
 - <http://java.sun.com/javaee>
 - <http://glassfish.dev.java.net>
- NetBeans IDE
 - <http://www.netbeans.org>
- Everything presented here today is documented on my blog on java.net
 - <http://weblogs.java.net/blog/bleonard>



Questions & Answers



Brian Leonard
Sun Microsystems, Inc.