



**Virtua™**  
technical excellence. it's that simple.



JavaOne

# Improve and Expand JavaServer Faces Technology with JBoss Seam

Michael Yuan

Product Manager, Red Hat

<http://www.michaelyuan.com/seam/>

Kito D. Mann

Author, JSF in Action

Principal Consultant

Virtua, Inc.



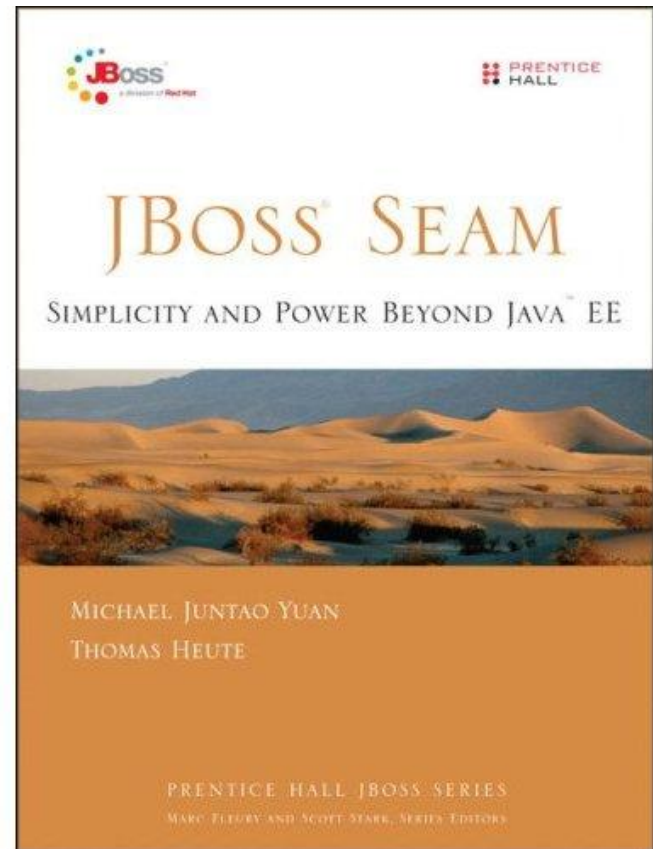
JavaOne

# Improve and Expand JavaServer Faces Technology with JBoss Seam

**Learn how JBoss Seam makes developing JavaServer Faces applications easier.**

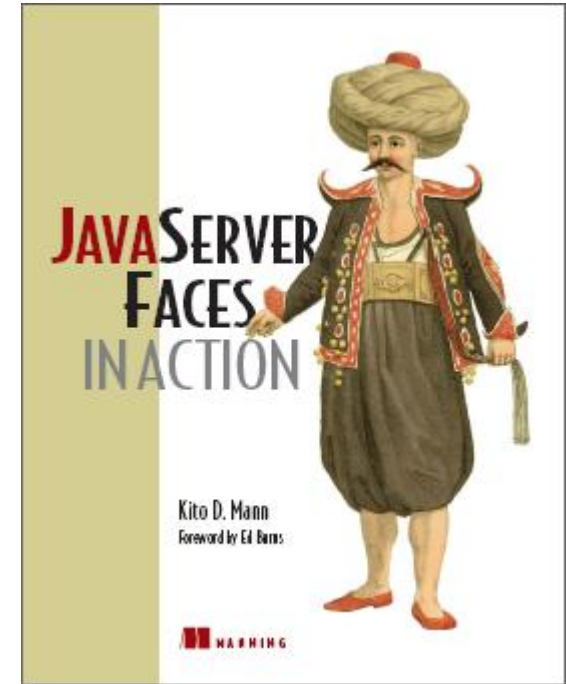
# About Michael Yuan

- Contributor to JBoss Seam
  - Seam on non-JBoss containers
  - Performance analysis
  - Tools integration
  - Samples and demos
- Author of the “JBoss Seam” from Prentice Hall
  - First Seam book written by Seam committers
  - Covers Seam 1.2
  - Available in JavaOne bookstore



# About Kito Mann

- Author, JavaServer Faces in Action
- Independent trainer, consultant, architect, mentor
- Internationally recognized speaker
  - JavaOne<sup>SM</sup> Conference, JavaZone, TSS Symposium, Javapolis, NFJS, AJAX World, etc.
- Founder, JSF Central
  - <http://www.jsfcentral.com>
- Java Community Process<sup>SM</sup> (JCP<sup>SM</sup>) Member
  - JavaServer Faces 1.2, JavaServer Pages<sup>TM</sup> (JSP<sup>TM</sup>) 2.1, Design-Time API for JavaBeans<sup>TM</sup> Architecture, Design-Time Metadata for JavaServer Faces Components, WebBeans, etc.
- Experience with Java<sup>TM</sup> platform since its release in 1995, web development since 1993



# Agenda

- What is Seam?
- Key Seam Features
- Other Goodies
- Application Requirements
- Future Directions
- Summary

# JSF: The Good

- Fully component-based web framework
  - Event-based UI programming model
  - Great for visual tools support
  - Many free and commercial component libraries
- Very rich interaction model
  - Many places for external frameworks to plugin
- Integrated validation / input conversion
- Unified Expression Language (EL)
- Multiple rendering output from same pages
- Non-JSP Template engines available
- Official Java EE standard

# JSF: The Bad

- Does not follow the Java EE 5 simplified programming model
  - No special integration with EJB3
  - Use verbose and repetitive XML instead of annotation
- Not “web friendly”
  - HTTP GET and RESTful URLs are hard
  - Use custom Ajax libraries / frameworks is hard
- Poor exception handling and error page re-direction
- Dependent on the inadequate Java EE security
- Hard to test out of container
- No support for dialogs or conversations

# What is Seam?

- “A lightweight framework for Java EE 5.0”
  - Integrates JSF with EJB3, POJOs, and JPA
- Corrects problems with pure JSF
- Makes heavy use of annotations
  - “Configuration by Exception”
- Developed by JBoss
  - Open source software released LGPL license
- Basis of the “WebBeans” JSR

# Agenda

- What is Seam?
- **Key Seam Features**
- Other Goodies
- Application Requirements
- Future Directions
- Summary

# Key Features: Seam is Made for JSF

- Reduces boilerplate code
- Dependency bijection
- No XML Hell
- Integrated ORM support
- RESTful URLs and page actions
- Extensive use of JSF EL
- Conversations
- Direct JavaScript integration for AJAX
- Elegant input validation

# Key Features: Seam is Made for JSF

- Graceful exception handling
- Fine-grained security
- Business process and rules integration
- Very easy to test
- Seam Gen

# Reduces Boilerplate Code

- No JSF backing beans
  - Use transactional components directly on web pages or in other components
  - EJB3 session beans, entity beans, or POJOs
- Bi-jection of named components
- Simplified navigation rules
  - Navigate based on state not return string
- Configuration by exception

# Dependency “Bijection”

- Dependency Injection
  - The ability of a “container” to *set* object references during (or immediately after) object creation
  - Natively supported by JSF’s MBCF
- Dependency Outjection
  - The ability of an object to *expose* objects to the container after a method has been executed
  - Supported only in Seam
- Dependency Bijection
  - Injection + Outjection

```
<h:form>
Please enter your name:<br/>
<h:inputText value="#{person.name}" size="15"/><br/>
<h:commandButton type="submit" value="Say Hello"
                 action="#{manager.sayHello}"/>
</h:form>
```

The web page

```
@Entity
@Name("person")
public class Person implements Serializable {

    private long id;
    private String name;
```

Named entity bean to back data fields  
in JSF forms

```
@Stateless
@Name("manager")
public class ManagerAction implements Manager {

    @In @Out Person person;
    @Out List <Person> fans;
    @PersistenceContext EntityManager em;

    public String sayHello () {
        person = p;
        em.persist (person);
```

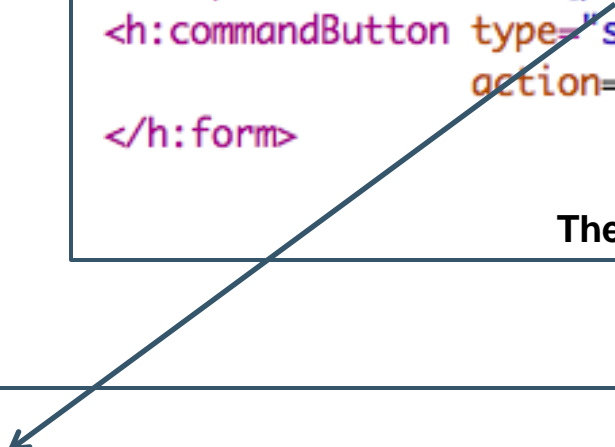
Named session to back JSF actions;  
Dependency bijection

```

<h:form>
Please enter your name:<br/>
<h:inputText value="#{person.name}" size="15"/><br/>
<h:commandButton type="submit" value="Say Hello"
                action="#{manager.sayHello}"/>
</h:form>

```

**The web page**



```

@Entity
@Name("person")
public class Person implements Serializable {

    private long id;
    private String name;
}

```

**Named entity bean to back data fields  
in JSF forms**

```

@Stateless
@Name("manager")
public class ManagerAction implements Manager {

    @In @Out Person person;
    @Out List <Person> fans;
    @PersistenceContext EntityManager em;

    public String sayHello () {
        person = p;
        em.persist (person);
    }
}

```

**Named session to back JSF actions;  
Dependency bijection**

```

<h:form>
Please enter your name:<br/>
<h:inputText value="#{person.name}" size="15"/><br/>
<h:commandButton type="submit" value="Say Hello"
                action="#{manager.sayHello}"/>
</h:form>

```

The web page

```

@Entity
@Name("person")
public class Person implements Serializable {

    private long id;
    private String name;
}

```

**Named entity bean to back data fields  
in JSF forms**

```

@Stateless
@Name("manager")
public class ManagerAction implements Manager {

    @In @Out Person person;
    @Out List <Person> fans;
    @PersistenceContext EntityManager em;

    public String sayHello () {
        person = p;
        em.persist (person);
    }
}

```

**Named session to back JSF actions;  
Dependency bijection**

# No XML Hell

- Do not use XML for code
- XML is used for the following scenarios
  - Configure the framework components
  - Configure web page parameters and actions
  - Configure business processes and pageflows

# Integrated ORM Support

- Integrated support for Open Session in View pattern
- Can keep database session open during conversations
- Stateless DAOs close the database session when the page rendering starts in the “view” layer

Seam is invented by Gavin King, the father of Hibernate

# Integrated ORM Support

- Automatic transaction rollback
  - Based on outcome or exceptions
- Lazy loading is essential for ORM performance
  - Seam “just works” with lazy loading
  - No need for DTOs (Data Transfer Objects) between layers

Seam is invented by Gavin King, the father of Hibernate

# RESTful Web Apps

- Inject HTTP GET parameters directly to Seam components before loading the page
- Invoke arbitrary Seam methods when the page is loaded
- Seam's JSF tags (e.g., `<s:link>`) can render into direct HTTP GET links

# Expand JSF EL

- Expand the EL syntax

```
<h:form>
Please enter your name:<br/>
<h:inputText value="#{person.name}" size="15"/><br/>
<h:commandButton type="submit" value="Say Hello"
                 action="#{manager.sayHello(person)}/>
</h:form>
```

- Expand the EL usage
  - Web pages
  - Annotation parameters
  - XML configuration files
  - TestNG test scripts

# Conversations

- More finely grained than HTTP session
- Support “web transaction”
- Support multiple browser windows/tabs
- Support BACK buttons
- Great for wizards, shopping carts, etc.



# DEMO

Hotel bookings demo



# Seam and Ajax

- Use Ajax JSF components
- Use Ajax4jsf to wrap regular JSF components
- Use Seam Remoting to directly access Seam components from JavaScript
  - Dojo integration
  - GWT integration

# Elegant input validation

- Use Hibernate validators on the data model to validate input fields
- Decorate the invalid fields
- Ajax validation supported out of the box



# DEMO

Ajax4jsf and GWT demos



# Failing Gracefully

- Redirect any Exception to any page
- Configure error pages on a per-page basis
- Redirect to previous page after login exceptions

# Fine-grained Security

- Username / password authentication
  - Authenticate against any backend
  - Any authentication logic in Java code
- Finely-grained access control
  - Web pages
  - Components on a web page
  - Java Methods
- Rule-based instance-level access control

# Protect a web page

```

<div>
  <h:outputLabel for="username">Login Name</h:outputLabel>
  <h:inputText id="username" value="#{identity.username}"/>
</div>
<div>
  <h:outputLabel for="password">Password</h:outputLabel>
  <h:inputSecret id="password" value="#{identity.password}"/>
</div>

```

The web page

```

public boolean authenticate() {
    List results = em.createQuery(
        "select u from User u where
        u.username=#{identity.username} and
        u.password=#{identity.password}")
        .getResultList();

    if ( results.size()==0 ){
        return false;
    } else {
        user = (User) results.get(0);
        return true;
    }
}

```

Authentication logic

```

<pages no-conversation-view-id="/main.xhtml"
        login-view-id="/home.xhtml">

  <page view-id="/main.xhtml"
        login-required="true">
    <navigation from-action="#{hotelBooking.selectHotel(hot)}">
      <redirect view-id="/hotel.xhtml"/>
    </navigation>

    <navigation from-action="#{bookingList.cancel}">
      <redirect/>
    </navigation>
  </page>

```

Declare protected pages

# Role-based Access Control

```
public boolean authenticate() {
    List results = em.createQuery(
        "select u from User u where
        u.username=#{identity.username} and
        u.password=#{identity.password}")
        .getResultList();

    if ( results.size()==0 ){
        return false;
    } else {
        user = (User) results.get(0);
        if (user.getRoles() != null) {
            for (UserRole mr : user.getRoles())
                identity.addRole(mr.getName());
        }
        return true;
    }
}
```

Add roles for each user

```
<pages no-conversation-view-id="/main.xhtml"
        login-view-id="/home.xhtml">

    <page view-id="/inventory.xhtml">
        <restrict>#{s:hasRole('admin')}</restrict>
    </page>
```

Page only accessible to users  
with the “admin” role

```
<s:div rendered="#{s:hasRole('admin')}">
    ... restricted content ...
</s:div>
```

Page element restricted to “admin” users

# Business Process and Rules

- Manage long-lived objects in the “process” stateful scope
- Map business process actors and actions to web actions via JSF EL
- Integration with JBoss Rules

# Seam Testing Framework

- TestNG based framework
- Mock database sessions for easy unit tests
- Mock the entire JSF interaction cycle for full test scripts
- Data binding in test via JSF EL

```

@Test
public void testSayHello() throws Exception {
    new FacesRequest("/hello.xhtml") {
        @Override
        protected void updateModelValues() throws Exception {
            setValue("#{person.name}", "Michael Yuan");
        }

        @Override
        protected void invokeApplication() {
            assert getValue("#{person.name}").equals("Michael Yuan");
            assert invokeMethod("#{manager.sayHello}") == null;
            assert getValue("#{person.name}") == null;
        }

        @Override
        protected void renderResponse() {
            List<Person> fans = (List<Person>) getValue("#{fans}");
            assert fans!=null;
            assert fans.get(fans.size()-1).getName().equals("Michael Yuan");
        }
    }.run();
}

```

# Seam Gen

- Ruby-on-Rails like application generator
  - CRUD app for complex inter-related tables
  - Ajax, security, and REST support built in
- Comprehensive tools support
  - ANT command line
  - Eclipse integration
  - NetBeans integration
- Come to BOF 4400 next for cool Seam Gen demos!

# Agenda

- What is Seam?
- Key Seam Features
- **Other Goodies**
- Application Requirements
- Future Directions
- Summary

# Other Goodies

- Template-based PDF output from JSF / Facelets pages
- Template-based email output
- Data Binding API
- CRUD framework
- Spring integration
- Enhanced Date/Time selector UI widget
- Correct JSF timezone issues
- Partial page cache

# Agenda

- What is Seam?
- Key Seam Features
- Other Goodies
- **Application Requirements**
- Future Directions
- Summary

# Application Requirements

- Requires JDK 1.5
- Servers:
  - Tomcat 5.5
    - Supports EJB3 in plain Tomcat via the JBoss Microcontainer
  - Glassfish
  - JBoss 4.0.5+ or 5.0 beta
  - WebLogic 9
  - WebSphere 6
- J2EE 1.4 servers require use of POJOs for backing beans instead of EJB session beans

# Agenda

- What is Seam?
- Key Seam Features
- Other Goodies
- Application Requirements
- **Future Directions**
- Summary

# Future Directions

- Enhancements to the security framework
- Better tools integration for Eclipse and NetBeans
- Seam/WS
  - Integration with EJB3 web services
- ESB and SOA integration
- Better portlet support
- Performance improvements
- WebBeans JSR
- Portions will be part of JSF 2.0

# Agenda

- What is Seam?
- Key Seam Features
- Other Goodies
- Application Requirements
- Future Directions
- **Summary**

# Summary

- Seam is made for JSF
- Lightweight Java EE framework
- Enhances and extends JSF
  - No backing beans required
  - Conversations
  - Bijection
  - Integrated ORM support
  - Testing framework
  - Seam Gen

# Resources

- Official Seam site
  - <http://www.jboss.com/products/seam>
- JSF Central
  - <http://www.jsfcentral.com>
- Michael Yuan's blog
  - <http://www.michaelyuan.com>

# Q&A

Michael Yuan

Product Manager, Red Hat

<http://www.michaelyuan.com/seam/>

Kito D. Mann

Author, JSF in Action

Principal Consultant

Virtua, Inc.