Introduction to Seam

Pete Muir

JBoss, a division of Red Hat
Road Map

- **Background**
- Seam concepts
- Seam with Wicket (at the BOF)
- Seam Extras
Advantages of JSF/JPA over Struts/EJB 2

- Fewer, finer grained artifacts
  - No DTOs required
  - Clean MVC
- Less noise
  - No Struts/EJB 2.x boilerplate code
  - No direct calls to HttpSession or HttpRequest
- Simple ORM
  - Even simpler than the Hibernate API!
Advantages of JSF/JPA over Struts/EJB 2

• JSF is flexible and extensible
  - Custom UI widget suites (open source)
  - Good AJAX support
• JPA
  - Powerful object/relational mapping, far beyond EJB 2.x CMP entity beans
• All components are POJO so easily testable with TestNG or JUnit
But, still some problems

• JSF
  - Backing bean couples layers and is just noise
  - Hard to refactor all the XML and String outcomes
  - No support for the business layer
  - Validation breaks DRY
  - XML is too verbose

• How do we write our business layer
  - EJB3? – can’t be used directly by JSF
  - EJB3? – no concept of scopes
And, some other stuff to solve!

- **Workflow**
  - Ad-hoc back buttoning not supported
  - No stateful navigation
  - Long running business processes?
- **Multi-tab/window support is not built in**
  - All operations happen in the session – leakage
  - No support for a conversation context
  - Memory leak – objects don’t get cleaned up quickly
Simple JSF with Seam

• Reference the entities directly!

```xml
<h:form>
  Item:  <h:outputText value="#{itemEditor.id} />
  Name: <h:inputText value="#{itemEditor.item.name}"/>
      <f:validateLength maximum="255"/>
  </h:inputText>
  Price (EUR): <h:inputText value="#{itemEditor.item.price}"/>
               <f:convertNumber type="currency" pattern="$###.##"/>
  </h:inputText>
  <h:messages />
  <h:commandButton value="Save" action="#{itemEditor.save}" />
</h:form>
```
@Name("itemEditor") @Scope(CONVERSATION) public class EditItemBean implements EditItem {

@In EntityManager entityManager;

Long id;
Item item;
// getter and setter pairs

@Begin public String find(Long id) {
    item = entityManager.find(Item.class, id);
    return item == null ? "notFound" : "success";
}

@End public String save(Item item) {
    item = entityManager.merge(item);
    return "success";
}
Road Map

- Background
- **Seam concepts**
- Seam with Wicket
- Seam Extras
Contextual variables

Contexts available in Seam

- Event
- Page
- Conversation
- Session
- Business Process
- Application
JSF lifecycle – quick review

- **RESTORE VIEW**: Restore the tree of UI components
- **APPLY REQUEST VALUES**: Synchronize request parameters with UI components
- **PROCESS VALIDATIONS**: Validate state of UI components
- **UPDATE MODEL**: Synchronize UI components with bound backing bean properties
- **INVOKE APPLICATION**: Notify action listeners, call action methods
- **RENDER RESPONSE**: Render a new tree of UI components
Application lifecycle

- Seam provides hierarchical, stateful contexts
How is state stored?

Depends on the context:

- **Conversation context**
  - Segmented HttpSession – times out if not used

- **Page context**
  - Stored in the component tree of the JSF view (page)
  - Can be stored in HttpSession or serialized to client

- **Business Process context**
Bijection

- Seam provides hierarchical, stateful contexts
- (Dependency) Injection fine for stateless applications BUT stateful applications need bidirectional wiring. Think about aliasing a stateful object into a context

```java
@Name("passwordChanger") public class PasswordChanger {

    @In EntityManager entityManager;

    @In @Out User currentUser;

    public void changePassword() {
        entityManager.merge(currentUser);
    }
}
```

Bijection: before the method call, inject the current user; after the method call, save it back into the context.
JPA Persistence Context

• What is the Persistence Context?
  - “a HashMap of all the objects I’ve loaded and stored”
  - holds (at most) one in-memory object for each database row while the PC is active
  - a natural first-level cache
  - can do dirty checking of objects and write SQL as late as possible (automatic or manual flushing)

• The Persistence Context has a flexible scope
  - default: same scope as the system transaction (JTA)
  - extended: the PC is bound to a stateful session bean
Which PC scope to use?

• Transaction scoped & detached objects
  - LazyInitializationException
  - NonUniqueObjectException
  - Less opportunity for caching

• An extended persistence context of a SFSB is
  - not available during view rendering (LIE again)
  - very complicated propagation rules

• No concept of a conversation
Seam managed persistence and transactions

- Seam managed PC is conversation scoped
  - Remains active through conversation,
  - Inject using @In
  - Allows use of manual flush mode
Navigation

• Stateful
  - Pageflow powered by jBPM engine (graphical editor)
  - Back button normally disabled

• Stateless
  - Through JSF or pages.xml
  - pages.xml is very powerful compared to JSF navigation rules (outcomes, application state, raise events on navigation)
Workflow – Business Process Management

• What is it?
  - Very long running (multiple days)
  - Lots of users (tasks can be assigned)

• Can contain many tasks
  - A task is completed by one user
  - Often a conversation
Model based validation

- Validate in the user interface?
  - Yes, need to report validation errors back to the user on the correct field
  - BUT normally need to enforce same constraints at the persistence layer and the database

```java
public @Entity class Item {

  @Id @GeneratedValue Long id;
  String name;

  @Length(min=3,
      max=1000,
      message="Must be between 3 and 1000 characters")
  String description;
```
Hibernate Validator

- Many built-in validators: Max, Min, Length, Range, Size, Email, Future, Past, Pattern, Email, CreditCard, ...
- Easy (very) to write custom validators
- Validation and message/error display with Seam UI components for JSF
- Works with every JPA provider, if used with Hibernate it generates SQL DDL constraints you can use in your database schema
- Standardization effort under way – JSR 303
Road Map

- Background
- Seam concepts
- Seam with Wicket
- Seam Extras
Seam Application Framework

<components>
  <fwk:entity-home name="userHome" entity-class="User" />
</components>

<h:form>
  <h:inputText value="#{userHome.instance.name}" />
  <h:inputSecret value="#{userHome.instance.password}" />
  <h:commandButton action="#{userHome.update}" />
  <h:messages />
</h:form>
Contextual Events

@Name("hotelBooking") public class HotelBooking {

    public void confirmRoomChoice() {
        booking.setRoom(chosenRoom);
        Events.instance().raiseEvent("roomChosen");
    }
}

public @Name("RoomList") class RoomList {

    @Out List<Room> availableRooms;

    @Observer("roomChosen")
    public void refresh() {
        availableRooms = entityManager.createQuery("select ...");
    }
}
And more

- Security
- Email
- PDF
- Remoting
- Asynchronicity (Java SE, EJB3 or Quartz)
- “Google your app” using Hibernate Search
- Integration and Unit Testing
- JSF components (deep integration into JPA)
- Components in groovy
- Webservices
- > 25 examples
- Portal support
Want to learn more?

http://www.seamframework.org
Go get it!

- Seam in JBoss Enterprise Application Platform 4.2 – supported product with Seam 1.2
- Seam 2.0.1.GA – community release of Seam2
- Seam 2.1.0.A1 – preview release of Seam 2.1 with Wicket and Portal support
The future of Seam?

- Web Beans!!
- First class support for other other containers (e.g. Websphere)
- Friendly URLs
- Identity Management
- SSO for security
- Flex support (talk at JBoss World!)
- Deeper integration with JBoss Portal (inter-portlet communication)
Q&A
More Seam at JBoss World

• Gavin is talking about Web Beans at 1:40pm on Friday
• Seam, Tools and Web Beans BOF tonight at 8pm
• 2 (!) Seam Case Studies on Friday at 9am
• “Agile development” on Friday at 1:40pm on Friday