

JBoss Seam

Stitching things together

Shashank Tiwari

(stiwari@saventech.com / www.shanky.org)

July 17 & Aug 2, 2007

Chicago Java Users Group (www.cjug.org)

Agenda

- Who am I?
- Basics of Seam
- Building stateful applications
- Adding richness using AJAX
- Integrating with the server side
- Enhancing existing Java EE frameworks
- Using POJOs
- Built-in components in Seam
- Facilitating agile methodology and RAD
- More perspectives
- Living harmoniously
- Where can I find more information on Seam?
- FAQ (s)

Who am I....

- Developer, speaker, author, open source enthusiast, blogger
- Principal architect – Saven Technologies
(www.saventech.com)
- Expert group member – jsr 274 (bean shell), jsr 283 (JCR 2.0), jsr 301 (JSF portlet bridge), jsr 312 (JBI 2.0)
- My blog on O'Reilly -
<http://www.oreillynet.com/pub/au/2799>
- My book on Seam – coming later this year!

Basics of Seam (1/3)

- Built by JBoss (Championed by Gavin King – the man behind Hibernate)
- Java framework
- Built on Java EE 5 +
- Leverages the embeddable JBoss EJB3 container (when an EJB3 container isn't available but you still need it)
- 2.0.0.BETA 1 is the current release
- And of course it is Open source 😊

Basics of Seam (2/3)

- Integrates Java EE frameworks and more
 - EJB3, JPA, JSF, Hibernate, Facelets, jBPM, jPDL
 - JavaScript
- Makes things easy – facilitates faster development
 - Annotations
 - Seam-gem
 - Not to forget the stitching it does
- A best practices blueprint
- Breaks the lightweight and heavyweight barrier (also thanks to Java EE itself incorporating so called lightweight features)

Basics of Seam (3/3)

- Achieves statefulness
- Adds AJAX
- Facilitates effective persistence
- Enhances beans (backing beans/managed beans)
- Makes POJOs powerful
- Minimizes configuration
- Provides developer tools
- Promotes reuse – avoids reinventing the wheel

- Above all gives you more to choose from !

Building stateful applications (1/6)

- Defines stateful contexts at different levels of granularity
 - Event – single JSF request
 - Page
 - Conversation – series of requests that constitute a logical bundle (default)
 - Session – good old HttpSession
 - Business process
 - Application – The Servlet context
 - Stateless
- Making sense of scopes
 - Request, Session, Application (also in Java Servlets) – user interaction infrastructure
 - Conversation, Business process – application and business logic

Building stateful applications (2/6)

- Stateful components
 - EJB 3.0 – session, entity, message-driven
 - POJOs – JavaBeans
- Component lifecycle

Bijection

- Inject – good old dependency injection - @In
- Outject – new ! - @Out

Example - A needs B which creates C and D needs C

(B injected into A, C outjected into the context, C injected into D)

Building stateful applications (3/6)

- Injection

```
Instance variable - @In MyObjectType myObject;  
Setter - @In  
    public void setMyObject(MyObjectType myObject) {  
        this.myObject=myObject; }  
  
@In("contextName")  
@In(create=true)  
@In(required=false)  
@AutoCreate - autocreate at component level  
@In("#{myObject.propertyName}") String propertyName;
```

- Outjection

```
Instance variable - @Out MyObjectType myObject;  
Getter - @Out  
    public MyObjectType getMyObject() {  
        return myObject; }
```

- You can do both at the same time - @In @Out on an instance variable or @In setter and @Out Getter

Building stateful applications (4/6)

- Declaration and creation

```
import static org.jboss.seam.ScopeType.SESSION; ....
```

```
@Entity /* EJB3 entity bean declaration */
```

```
@Name("myentity") /* this name is used throughout Seam to refer to this object */
```

```
@Scope (SESSION) /* Session level state management */
```

```
public class MySeamEntity implements Serializable {...
```

```
-----  
import static org.jboss.seam.ScopeType.CONVERSATION; ....
```

```
@Stateful /* EJB3 stateful session bean declaration */
```

```
@Name("myaction")
```

```
@Scope (CONVERSATION) /* Conversation level state management */
```

```
public class MyAction implements MyActionInterface {...
```

```
-----  
@Roles({@Role(name="caseOneUser", scope=EVENT), @Role(name="caseTwoUser", scope=SESSION)})
```

- Initialization and destruction

- @Create – call methods after the constructor to initialize – helpful when you want to set property values before any other method is called and have access to injected objects when you do that
- @Destroy – call methods before object removed from the context –
- @Factory – call methods to instantiate before the properties are used – for those who don't map to a class
- @Startup(depends=....) – immediate instantiation for application and session contexts

Building stateful applications (5/6)

- Component lookup from Contexts
 - Known scope – e.g. `Contexts.getConversationContext()`, `Contexts.getSessionContext()` ; then call `get` (String value) – somewhat like you do with session objects.
 - Priority search - `Contexts.lookupInStatefulContexts()`
 - Priority order - Event context ->Page context ->Conversation context ->Session context ->Business process context ->Application context
- Component types and Contexts
 - Stateless session beans – `stateless` – JSF Action Listeners
 - Stateful session beans – `conversation` – Action Listeners, Backing Beans –need to define an interceptor in the `ejb-jar.xml`
 - Entity beans – `conversation` – persistence, Backing Beans (No bijection, context demarcation)
 - POJOs – `event` – use them any anything you desire (Remember no declarative transaction demarcation, declarative security, EJB 3.0 persistence, timeout methods)
 - Built-in components

Building stateful applications (6/6)

- Manages concurrency –
 - Synchronous web apps vs AJAX
 - Coarse grained vs fine grained calls
 - session, application – multithreaded, event, page – single threaded
 - `@Synchronized` – make components thread safe
- Configuring components
 - Properties file / web.xml or components.xml

Adding richness using AJAX (1/6)

- AJAX – everybody already knows what it is? ☺
- Options
 - AJAX enabled JSF components
 - Adding AJAX to existing JSF components
 - Remoting JavaScript

Adding richness using AJAX (2/6)

- Using AJAX enabled JSF Components
 - ICEfaces

<http://component-showcase.icefaces.org/component-showcase/>

Form validation example -

```
<ice:inputText value="{myObject.propertyName}" partialSubmit="true"/>
```

AutoComplete example -

```
<ice:selectInputText rows="6" width="300" listVar="city"
    valueChangeListener="{autoCompleteBean.updateList}" listValue="{autoCompleteBean.list}">
    <f:facet name="selectInputText"> <ice:panelGrid columns="3" style="margin-bottom:-
    20px;"
        columnClasses="cityCol,stateCol,zipCol">
<ice:outputText value="{city.city}"/> ...</ice:panelGrid> </f:facet> </ice:selectInputText>
```

The Seam component showcase at the ICEfaces site is a good resource for examples. The above example id from there.

Adding richness using AJAX (3/6)

- Required configuration to use ICEfaces
 - xmlns:ice=“<http://www.icesoft.com/icefaces/component>”
 - Replace jsf-facelets.jar with the ICEfaces specific implementation
 - Configure the appropriate view handler
 - In web.xml replace Seam view handler for facelets and add the ICEfaces Servlets
 - Of course add the jars in
<http://support.icesoft.com/jive/servlet/KbServlet/download/721-102-16/UsingICEfacesWithSeam.html>
- Alternatives
 - JBoss RichFaces – JBoss and Exadel

Adding richness using AJAX (4/6)

- Adding AJAX to existing JSF components

- ajax4jsf

```
<h:inputText value="#{myObject.propertyName}" ...>
```

```
<s:validate/>
```

```
<a4j:support event="onblur" reRender="....componentID1,componentID2,...."  
  oncomplete="some javascript function call"/>
```

```
</h:inputText>
```

- Convert action events to trigger AJAX calls
- Replace `<h:commandButton>` with `<a4j:commandButton>` and `reRender`
- Submit only what you need to
- Use `<a4j:region>....</a4j:region>` and when the `UICommand` component in the region is activated whatever is in it is submitted
- Poll to get the updated state – `a4j:poll`
- Include other JSF pages – `a4j:include` – use it like a frame or `iFrame`

Adding richness using AJAX (5/6)

- Remoting JavaScript

- Dojo toolkit

- Define your session bean method – called by JavaScript
 - Annotate it with `@WebRemote`
 - Don't forget to include the resource Servlet configuration in web.xml
 - Include -

```
<script type="text/javascript"
src="seam/resource/remoting/resource/remote.js">
</script>
```

- Seam generates JavaScript to access `@WebRemote` methods
 - Get a JavaScript version of the component -
`Seam.Component.getInstance("componentName")` and
`Seam.Component.newInstance()` – entity beans and POJOs
 - Javascript call returns immediately – register a callback handler

Adding richness using AJAX (6/6)

- There are many more
 - JMaki – JavaScript widgets wrapped and exposed as JSF
 - DynaFaces – adds AJAX to JSF
 - Apache Trinidad and Oracle ADF

Integrating with the server side (1/3)

- We already know that -
 - EJB3 components can be used as backing beans
 - Contextual scope boundaries include the entire application
- JSF EL extended to support calls to server side methods that take in a parameter

Integrating with the server side (2/3)

- Effective data-binding
 - databinding package – DataBinder and DataSelector interfaces
 - Example - Editable Data-Grid
Edit and delete rows of data

<h:dataTable> - iterate over a List – Collected marked as @DataModel – object properties map to table columns

How to find the current row? – identify it with @DataModelSelection annotation or use extended EL to pass the current selection in `#{controllerClass.eventHandler(currentSelection)}`

Integrating with the server side (3/3)

- Using Hibernate Validators

- Annotate entity beans getters with the validation annotations and include `<s:validate />` inside the input data fields or use `<s:validateAll>` all input text components `</s:validateAll>`

Examples –

- `@Length(max=,min=)` string length is in the range
- `@Max(value=)` numeric value maximum
- `@Min(value=)`
- `@Range(max=,min=)`
- `@NotNull`
- `@Past` check if the date is in the past
- `@Future` check if the date is in the future
- `@Pattern(regex="regexp", flag=)`
- `@Size(max=,min=)` collection or array pis inside the given range.
- `@Email`
- `@Valid` validation recursively on the associated object

http://www.hibernate.org/hib_docs/validator/reference/en/html_single/

Enhancing existing Java EE frameworks (1/3)

- Enhancing JSF
 - Seam UI tags
 - Conversation management – include external hyperlinks in the conversation context
 - Server side validation at the UI
 - PDF, email, wiki facelets template
 - JSF EL enhancement
 - Support in methods in parameters
 - Extends usage of EL beyond the web page – configuration files, jBPM processes
 - Seam filter Servlet
 - Conversation contexts preserved during redirect
 - Catches exceptions (uncaught runtime)

Enhancing existing Java EE frameworks (2/3)

- Recommends using Facelets over JSP
 - Templating system helps generate pages
 - Increases performance (directly renders to the browser – XHTML, no parsing by the JSP engine)
 - Easier debugging with Facelets
 - Good Introduction _ Jacob Hookom's articles on JSFCentral
 - http://www.javaserverfaces.info/articles/facelets_1.html
 - http://www.javaserverfaces.info/articles/facelets_2.html
 - http://www.javaserverfaces.info/articles/facelets_3.html

Enhancing existing Java EE frameworks (3/3)

- Workspace
 - Conversation context helps implement the concept of workspace
 - conversationList maintains a list of all active conversations
 - GET call creates a new conversation
 - If one needs new GET to be part of the existing conversation then pass the cid name-value pair
- Business process management
 - Long running conversations can be defined using attributes – clr
 - Business process context allows state management across users in long running business process

Using POJOs

- POJOs can be used instead of EJB3 components as Seam components.
 - They are stateful and have the conversation scope (default)
 - You will miss your EJB container services though
 - Can't use `@PersistenceContext`
 - All persistence contexts are extended (`@PersistenceContext (type=EXTENDED)`)
 - Is JPA EJB3? Yes and No
 - No interface needs to be implemented as in a session bean – it's a POJO after all !

Built-in components in Seam (1/3)

- Out-of-box
 - Configuration required only to alter default (components.xml)
 - Replace them by reusing the components name
 - Components provide varied functionality – injection, utility, internationalization, controlling conversations, controlling jBPM, security, JMS, mail, etc....
- Context injection components
 - `org.jboss.seam.core.contexts` (access to current context associated with the thread)
 - `org.jboss.seam.faces.facesContext` (access to FacesContext instance)

Built-in components in Seam (2/3)

- Components that control conversations
 - `org.jboss.seam.core.conversation` - control of attributes of the current Seam conversation.
 - `getId()` — returns the current conversation id
 - `isNested()` — is the current conversation a nested conversation?
 - `isLongRunning()` — is the current conversation a long-running conversation?
 - `getParentId()` — returns the conversation id of the parent conversation
 - `setTimeout(int timeout)` — sets the timeout for the current conversation
 - `setViewId(String outcome)` — sets the view id to be used when switching back to the current conversation from the conversation switcher, conversation list, or breadcrumbs.
 - `setDescription(String description)` — sets the description of the current conversation to be displayed in the conversation switcher, conversation list, or breadcrumbs.
 - `leave()` — exit the scope of this conversation, without actually ending the conversation.
 -
 - `org.jboss.seam.core.conversationList` - conversation list
 - `org.jboss.seam.core.conversationStack` - conversation stack (breadcrumbs)
 - `org.jboss.seam.faces.switcher` - conversation switcher

Built-in components in Seam (3/3)

- Utility Components
 - `org.jboss.seam.faces.facesMessages` - propagates faces success messages across a browser redirect.
 - `org.jboss.seam.faces.redirect` - performs redirects with parameters (this is especially useful for bookmarkable search results screens).
 - `org.jboss.seam.faces.httpError` - sends HTTP errors.
 - `org.jboss.seam.core.interpolator` - interpolates the values of JSF EL expressions in Strings.
 - `org.jboss.seam.core.expressions` - creates value and method bindings.
 - `org.jboss.seam.core.pojoCache` - manager component for a JBoss Cache PojoCache instance.

Facilitating agile methodology & RAD

(1/2)

- Seam-gen
 - ANT script that does wonders
 - For Eclipse, go to the seam directory and type the following commands
 - seam setup
 - seam new-project
 - seam generate-entities (define the database tables first)
- CRUD application generator – Hibernate tools
 - ANT script and Eclipse plug-in
 - Based on the hibernate configuration files – hibernate.cfg.xml
 - Annotated EJB3
 - Database schema

Facilitating agile methodology & RAD

(2/2)

- IDE plug-in
- Eclipse
 - Hibernate Tools
 - Seam-gen
 - JBoss IDE
- NetBeans
 - Seam plug-in exists

More perspectives

- Internationalization
- Pageflows and jPDL
 - jPDL-based pageflow - specify the name of the process definition using a `@Begin`, `@BeginTask` or `@StartTask` annotation
- JBoss Rules can be called from within Seam
- Asynchronous messaging dispatcher options
 - `java.util.concurrent.ScheduledThreadPoolExecutor` (by default)
 - the EJB timer service (for EJB 3.0 environments)
 - Quartz

Living harmoniously (1/6)

- Seam utilizes standard Java EE as far as possible
- Also integrates
 - JPA and Hibernate3 - persistence
 - EJB Timer Service and Quartz – lightweight asynchronicity
 - jBPM - workflow
 - JBoss Rules - business rules
 - Meldware Mail - email
 - Hibernate Search and Lucene - full text search
 - JMS - messaging
 - JBoss Cache - page fragment caching
 - JAAS & JBoss Rules – security
 - JSF tag lib - rendering PDF, outgoing email, charts and wikitext

Living harmoniously (2/6)

- Seam components can be called
 - Synchronously – web service
 - Asynchronously – e.g. JavaScript, GWT

Living harmoniously (3/6)

- JBoss Seam and GWT

- GWT widgets and Seam components – direct interaction

For Seam component to be called from a GWT widget, create synchronous and asynchronous interfaces for the methods you want to expose.

Extend `com.google.gwt.user.client.rpc.RemoteService`

- `public interface SeamGWTService extends RemoteService { public String myMethod(String myParameter); }` - synchronous
- `public interface SeamGWTAsyncService extends RemoteService { public void myMethod(String myParameter, AsyncCallback callback); }` - asynchronous
- Annotate the method you exposed with `@WebRemote` in the Seam component

Get a reference to the asynchronous GWT client stub `....SeamGWTAsyncService`
`getService`

Write the GWT widget that invokes the method on the client stub

```
getService().myMethod(input, new AsyncCallback() { public void onFailure(Throwable t) {  
    Window.alert(t.getMessage()); } public void onSuccess(Object data) { Window.alert((String) data); }  
});
```

The action listener should invoke this method (possibly indirectly)

Living harmoniously (4/6)

- JBoss Seam and the Spring framework

- Move Spring to Seam
- Access Seam's features from Spring

- Inject Seam component instances into Spring beans

- Add Seam namespace to the spring beans definition

```
<bean id="someSpringBean" class="SomeSpringBeanClass" scope="prototype"> <property name="someProperty"> <seam:instance name="someComponent"/> </property> </bean>
```

- Seam component instance injected into a singleton Spring bean at construction time and not at method call time

- Inject Spring beans into Seam components

- Spring-JSF integration can be done using `DelegatingVariableResolver`. This makes all spring beans available in the EL by their bean id.

```
@In("#{springImportantService}") private SeamImportantService seamImportantService;
```

Living harmoniously (5/6)

- Convert Spring beans into Seam components

```
<bean id="aSpringBean" class="aSpringBeanClass" scope="..."> <seam:component /> </bean>
```

Within the declaration of the bean – spring bean scope prototype in case the seam scope is defined

Seam scope is defined as an attribute of <seam:component /> tag

- Facilitate Spring beans to live in any Seam context

Seam's scope can be used as a Spring 2.0 style custom scope.

```
<!-- Only needs to be specified once per bean factory--> <seam:configure-scopes /> ... <bean  
id="someSpringBean" class="SomeSpringBeanClass" scope="seam.CONVERSATION" />
```

Living harmoniously (6/6)

- JBoss Seam and Groovy
 - Groovy is dynamic and uses Java syntax
 - Use Groovy 1.1 Beta1+ (annotations support)
 - Groovy supports properties natively
 - Nothing special ! Do what you do with Java code
- JBoss Seam and sandbox-based rich technologies?
 - JBoss Seam and Flex
 - JBoss Seam and Swing + Java 2D + Java 3D

More information on Seam

- <http://labs.jboss.com/jbossseam/>
- <http://docs.jboss.com/seam/2.0.0.B1/reference/en/html/index.html> - documentation
- <http://docs.jboss.com/seam/2.0.0.B1/api/> - Javadoc
- <http://docs.jboss.com/seam/2.0.0.B1/ui/apidocs/> - UI API
- <http://www.saventech.com/frameworks/jbossseam/> - this presentation & more....
- Bunch of books available today, my book will come in a few months

Frequently answered questions

- Can Seam run outside the JBoss AS?
 - Yes
- Can it be used without JSF?
 - Yes, but depends on what are you going to use instead
- Can I use JBoss Seam in a proprietary application?
 - Yes, licensed under GNU LGPL

Thank You

- Please provide your feedback. Mail me.
- You are welcome to reach out to me with your questions over email.
 - My email – stiwari@saventech.com
- Where can you find this presentation?
 - http://www.saventech.com/documents/jboss_seam.pdf