An Introduction to Seam 3

Reza Rahman
Expert Group Member, Java EE 6 and EJB 3.1
Resin EJB 3.1 Lite Container Developer
Author, EJB 3 in Action
reza@caucho.com
CDI, Weld and Seam 3

- CDI is the dependency injection standard for Java EE
- Weld is the reference implementation of CDI
- Seam 3 is a set of CDI portable extension modules
Contexts and Dependency Injection for Java EE (CDI)
CDI Overview

- Next-generation dependency injection for Java EE
- Also known as JSR-299, formerly known as WebBeans
- Synthesizes best ideas from Seam 2, Guice and Spring
- Many innovative features on its own right
- Focus on loose-coupling, Java-centric type-safety, annotations, expressiveness and ease-of-use
- Makes Java EE much more flexible, testable, pluggable and extensible
CDI Features

- Basic dependency injection
  - @Inject, @Qualifier, @Stereotype, @Alternative, Instance, @All, @Any

- Component naming
  - @Named

- Context management
  - @Dependent, @RequestScoped, @SessionScoped, @ConversationScoped, @ApplicationScoped, @Scope
CDI Features

- Custom Object Factories
  - `@Produces`, `@Disposes`, `InjectionPoint`

- Lightweight Events
  - `Event`, `@Observes`

- Interceptors/Decorators
  - `@Interceptor`, `@InterceptorBinding`, `@AroundInvoke`, `InvocationContext`, `@Decorator`, `@Delegate`

- Portable extensions SPI*
JBoss Weld
Weld Overview

- CDI reference implementation and compatibility test kit from JBoss
- Included in GlassFish and JBoss AS
- Can be used with Tomcat, Jetty
- Can be used in Java SE
Weld Runtime Environments

- JBoss AS 6
- GlassFish V3
- Tomcat
- Jetty
- Java SE
Weld Event Based Java SE Bootstrap

@Singleton
public class AlertListener {
    ...
    public void init(
        @Observes ContainerInitialized event,
        @Parameters List<String> parameters) {
            showSplash();
            setupUi();
            showUi(parameters);
        }
    }
    ...
}
Weld Java SE Programmatic Bootstrap

```java
public static void main(String[] arguments) {
    WeldContainer weld = new Weld().initialize();

    weld.instance().select(
        AlertListenerBean.class).startListening();

    weld.shutdown();
}
```
JBoss Seam 3
Seam 3

- Set of CDI portable extensions independent of Weld
- Each module developed separately
- There are Seam “umbrella” releases
## Seam Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solder</td>
<td>General enhancements to CDI</td>
</tr>
<tr>
<td>XML Configuration</td>
<td>XML configuration for managed beans</td>
</tr>
<tr>
<td><strong>Persistence</strong></td>
<td>Transactions and persistence for non-EJB managed beans</td>
</tr>
<tr>
<td>Faces</td>
<td>JSF enhancements</td>
</tr>
<tr>
<td>Servlet</td>
<td>Servlet enhancements</td>
</tr>
<tr>
<td><strong>JMS</strong></td>
<td>JMS integration</td>
</tr>
<tr>
<td>REST</td>
<td>REST enhancements</td>
</tr>
<tr>
<td>Remoting</td>
<td>JavaScript remoting of managed beans</td>
</tr>
</tbody>
</table>
# Seam Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Catch</td>
<td>Exception handling framework</td>
</tr>
<tr>
<td>International</td>
<td>Internationalized/localized locale, time, messages</td>
</tr>
<tr>
<td>Security</td>
<td>Higher level security API for Java EE</td>
</tr>
<tr>
<td>Mail</td>
<td>JavaMail integration</td>
</tr>
<tr>
<td>Cron</td>
<td>CDI based scheduling</td>
</tr>
<tr>
<td>Document</td>
<td>Document generation (PDF, Excel)</td>
</tr>
<tr>
<td>Seam 2</td>
<td>Seam 2 backwards compatibility</td>
</tr>
<tr>
<td>Spring</td>
<td>Spring interoperability</td>
</tr>
</tbody>
</table>
## Seam Modules

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wicket</td>
<td>Wicket Integration</td>
</tr>
<tr>
<td>GWT</td>
<td>GWT integration</td>
</tr>
<tr>
<td>Drools</td>
<td>Drools integration</td>
</tr>
<tr>
<td>jBPM</td>
<td>jBPM integration</td>
</tr>
<tr>
<td>JBoss ESB</td>
<td>JBoss ESB integration</td>
</tr>
<tr>
<td>Clouds</td>
<td>Cloud frameworks integration (JClouds, Infinispan)</td>
</tr>
<tr>
<td>Forge</td>
<td>Rapid application development for Java EE</td>
</tr>
</tbody>
</table>
Solder

- Used to be part of Weld, good laboratory for CDI.next
- Generally useful CDI extensions for application, plug-in and framework developers
- Logging
- Injecting EL expression evaluator
- Extended resource loading
- Extended APIs for injection, naming, management
- Annotation processing and reflection utilities
Logging

@MessageLogger
public interface AccountLog {
   ...
   @LogMessage
   @Message("Account %s was overdrawn by %10.2f")
   public void overdrawn(String accountNumber, double amount);
   ...
}

@Inject @Category("accounts") @Locale("en_GB")
AccountLog log;

log.overdrawn(account.getNumber(), amount);
@Inject
private Expressions expressions;
...
Customer customer =
    expressions.evaluateValueExpression("#{account.customer}");
...
Double interest =
    expressions.evaluateMethodExpression("#{account.calculateMonthlyInterest}");
Enhanced Resource Handling

@Inject
@Resource("WEB-INF/beans.xml")
private URL beansXml;

@Inject
@Resource("WEB-INF/web.xml")
private InputStream webXml;

@Inject
@Resource("META-INF/aws.properties")
private Properties awsProperties;
XML Configuration

- CDI Focuses mainly on annotations, not XML
- XML configuration is sometimes needed or preferred
- Seam XML configuration allows portable CDI XML configuration across implementations
- More compact, schema-driven and type-safe than traditional XML configuration
- Likely candidate for CDI.next
public class MailSender {  
   ...
   public void setEmail(String email) {
       ...
   }
   ...
}

@Qualifier
@Retention(RetentionPolicy.RUNTIME)
@Target({ ElementType.FIELD, ElementType.METHOD })
public @interface Admin {}

@Qualifier
@Retention(RetentionPolicy.RUNTIME)
@Target({ ElementType.FIELD, ElementType.METHOD })
public @interface Support {{
XML Object (Re)-Wiring

<beans xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:seam="urn:java:seam:core"
    xmlns:acme="urn:java:com.acmebank">

    <acme:MailSender>
        <seam:ApplicationScoped/>
        <acme:Admin/>
        <acme:email>admin@acmebank.com</acme:email>
    </acme:MailSender>

    <acme:MailSender>
        <seam:ApplicationScoped/>
        <acme:Support/>
        <acme:email>support@acmebank.com</acme:email>
    </acme:MailSender>

    ...

</beans>
Injecting XML-Wired Objects

```java
public class CustomerSupportService {
    ...
    @Inject
    @Support
    private MailSender mail;
    ...
    mail.send();
    ...
}
```
Refining Beans with XML

@ApplicationScoped
public class CurrencyConverter {
    ...
    private String currency = "us-dollar";
    private int precision = 2;
    ...
}

<acme:CurrencyConverter currency="pound-sterling">
    <s:modifies/>
    <!-- <s:replaces/> -->
    <acme:precision>4</acme:precision>
</acme:CurrencyConverter>
Seam Persistence

- EJB transactions in CDI managed beans instead of EJB beans
- JPA bootstrap to work with Seam Persistence
- Likely standardized in Java EE 7
Persistence and Transactions in Seam

@ApplicationScoped
@TransactionAttribute(
    TransactionAttributeType.REQUIRED)
@Audited
public class DefaultAccountService
    implements AccountService {
    @Inject
    private EntityManager entityManager;

    public void addAccount(Account account) {
        ...
    }

    ...
}
Seam Faces

- CDI Enhancements for JSF 2
- More CDI scopes to match JSF 2: @ViewScoped, @RenderedScoped, @FlashScoped
- Cross-field validation, page pre-load processing
- @*Scoped and @Inject in Validators and Converters
- Injecting Faces objects
- Bridging CDI Events with the JSF life-cycle
- Likely standardized in Java EE 7
Cross-Field Validation Form

<h:form id="locationForm">
   <h:inputText id="city" value="#{bean.city}"/>
   <h:inputText id="state" value="#{bean.state}"/>
   <h:inputText id="zip" value="#{bean.zip}"/>
   <h:commandButton id="submit" value="Submit"
       action="#{bean.submit}"/>
   <seam:validateForm
       validatorId="locationValidator"/>
</h:form>
Cross-Field Validator

@FacesValidator @RequestScoped
public class LocationValidator implements Validator {
    ...
    private @Inject Directory directory;

    private @Inject @InputField String city;
    private @Inject @InputField String state;
    private @Inject @InputField String zip;
    ...
    public void validate(FacesContext context, UIComponent component, Object values) throws ValidatorException {
        if (!directory.exists(city, state, zip))
            throw new ValidatorException(new FacesMessage("Invalid location, try again"));
    }
}
JSF Pre-Load

```xml
<f:metadata>
   <f:viewParam name="id"
      value="#{accountManager.accountId}"/>
   <s:viewAction
      action="#{accountManager.loadAccount}"/>
</f:metadata>
```
JSF Object Injection

@inject FacesContext facesContext;
@inject ExternalContext externalContext;
@inject NavigationHandler navigationHandler;
@inject Flash flash;
Seam JMS

- Injecting connections, sessions, queues, topics, message producers, message receivers
- CDI Event Observers as JMS message listeners
- CDI Events sent to JMS destinations
- Some features could be standardized in Java EE 7
public class TransferSender {
    ...
    @Inject
    @JmsSession(
        transacted=true,
        acknowledgementType=Session.AUTO_ACKNOWLEDGE)
    private Session session;

    @Inject
    @JmsDestination(jndiName="jms/TransferQueue")
    private MessageProducer producer;
    ...
    public void sendTransfer(Transfer transfer) {
        ...
        producer.send(session.createObjectMessage(transfer));
        ...
    }
    ...
}
Catch

- Event-based exception handling framework
- Automatic exception chain handling
- Type-based handler callback hierarchy
@HandlesExceptions
public class IOExceptionHandler {
    ...
    public void handleIOExceptions(
            @Handles CaughtException<IOException> event) {
        System.out.println("I/O Problem: " +
            event.getException().getMessage());
    }
    ...
}
Seam REST

- Configure exception handling rules
- Integrate with Seam Catch
- Integrate Bean Validation with JAX-RS
- Templates for JAX-RS output
- RESTEasy client integration
- Some features could be standardized in Java EE 7
Seam REST Exception Mapping

<rest:SeamRestConfiguration>
  <rest:mappings>
    <seam:value>
      <exceptions:Mapping
        exceptionType="javax.persistence.NoResultException"
        statusCode="404">
        <exceptions:message>
          Resource does not exist.
        </exceptions:message>
      </exceptions:Mapping>
    </s:value>
    <s:value>
      <exceptions:Mapping
        exceptionType="java.lang.IllegalArgumentException"
        statusCode="400">
        <exceptions:message>Illegal value.</exceptions:message>
      </exceptions:Mapping>
    </s:value>
  </rest:mappings>
</rest:SeamRestConfiguration>
Seam REST Bean Validation

@Stateless
@Path("/bid")
public class DefaultBidService implements BidService {

  ...
  @POST
  @Consumes("text/xml")
  @ValidateRequest
  public void addBid(Bid bid) {
    ...
  }
  ...

}
Seam REST Output Templates

@GET
@Produces({ "application/json",
               "application/categories+xml",
               "application/categories-short+xml" })

@ResponseTemplate.List({
    @ResponseTemplate(
        @ResponseTemplate(
            value="/freemarker/categories.ftl",
            produces = "application/categories+xml"),
        @ResponseTemplate(
            value="/freemarker/categories-short.ftl",
            produces = "application/categories-short+xml")}

public List<Category> getCategories() {
Seam RESTEasy Client

@Inject
@RestClient("http://services.actionbazaar.com/bid")
private BidService bidService;

... 

bidService.addBid(bid);
Seam Remoting

- Export CDI beans to AJAX via remoting
- RPC and domain object models of remoting
- Remote context
- Client-side bean validation
Seam Remoting on the Server

@Named("bidService")
@Stateless
public class DefaultBidService implements BidService
{
    ...
    @WebRemote
    public void addBid(String bidder, String item,
        Double amount) {
        ...
    }
    ...
}
Seam Remoting on the Client

```javascript
<script type="text/javascript">
  function addBid() {
    ...
    Seam.createBean("bidService").addBid(bidder, item, amount);
    ...
  }
</script>

<button onclick="javascript:addBid()">Bid!</button>
```
Seam Servlet

- Bridging CDI Events with the Servlet life-cycle
- Injecting Servlet objects
- Integration with Seam Catch
- Some features could be standardized in Java EE 7
public void onServletContextInitialization(
    @Observes @Initialized ServletContext context) {
    ...
}

public void onSessionInitialization(
    @Observes @Initialized HttpSession session) {
    ...
}

public void onRequestInitialization(
    @Observes @Initialized @Path("/bid") HttpServletRequest request) {
    ...
}
Servlet Object Injection

@Inject  ServletConfig configuration;
@Inject  ServletContext context;
@Inject  HttpSession session;
@Inject  HttpServletRequest request;
@Inject  @RequestParam("id") String bidId;
@Inject  @HeaderParam("User-Agent") String userAgent;
@Inject  HttpServletResponse response;
@Inject  List<Cookie> cookies;
International

- Managing locale/time-zone for the application or session
- Internationalized, localized messages
Getting and Setting Default Locale

```java
@Inject Locale defaultLocale;
@Inject DateTimeZone defaultTimeZone;

<locale:DefaultLocaleProducer>
  <seam:specializes/>
  <locale:defaultLocaleKey>
    en_US
  </locale:defaultLocaleKey>
</locale:DefaultLocaleProducer>
<timezone:DefaultTimeZoneProducer>
  <s:specializes/>
  <timezone:defaultTimeZoneId>
    America/New_York
  </timezone:defaultTimeZoneId>
</timezone:DefaultTimeZoneProducer>
```
Getting and Setting User Locale

@Inject @UserLocale Locale userlocale;
@Inject @UserTimeZone DateTimeZone userTimeZone;

@Inject @Changed Event<Locale> localeEvent;
@Inject @Changed Event<DateTimeZone> timeZoneEvent;

public void setUserLocale() {
    localeEvent.fire(Locale.CANADA);
}

public void setUserTimeZone() {
    timeZoneEvent.fire(DateTimeZone.forID("America/Vancouver"));
}
Summary

- CDI next generation dependency injection for Java EE
- Weld reference implementation from JBoss included in GlassFish and JBoss AS
- Seam a large set of CDI portable extensions
- Logging, XML, RAD, persistence/tx, JSF, JMS, JAX-RS, JavaScript remoting, security, i18n & l10n, JavaMail, scheduling, documents, Servlet, Spring, Wicket, GWT, Drools, jBPM, JBoss ESB, exceptions, cloud
- More on the way!
References

- Weld, the JBoss reference implementation for JSR 299, [http://seamframework.org/Weld](http://seamframework.org/Weld)
- Seam 3, [http://seamframework.org](http://seamframework.org)
Thanks!

Email: reza@caucho.com

Twitter: cauchoresin

Facebook:
http://www.facebook.com/pages/Caucho-Technology/8671323790

Presentation materials: http://www.caucho.com/articles/