PV243 Advanced Java technologies: JBoss

Part 3, Seam3

Marek Schmidt

March 2012

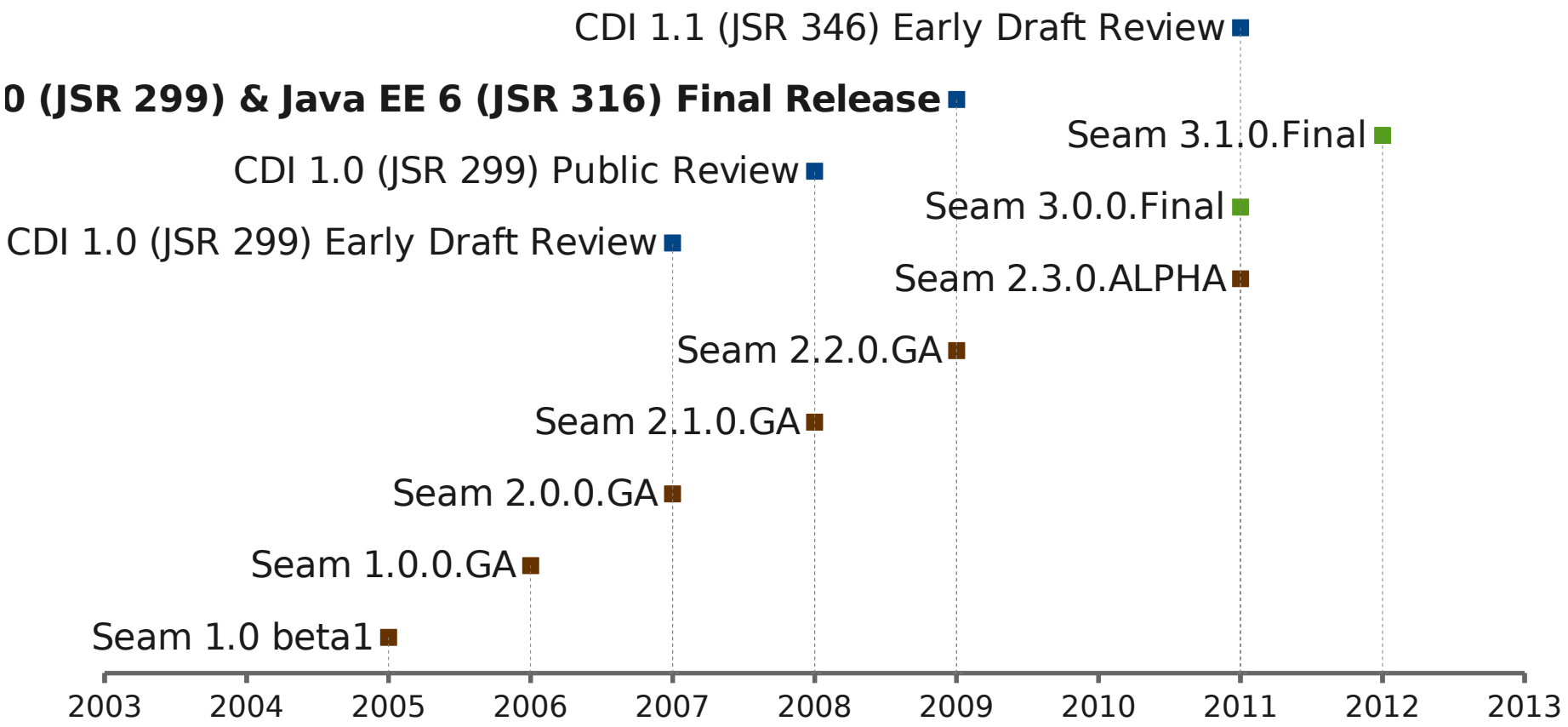
Who**am**I

Marek Schmidt <**maschmid**@redhat.com>

Seam Project

Seam's *mission is to provide a fully-integrated development platform for building* rich, standards-based *Internet applications tailored for traditional and cloud deployments.*

Short History of Seam



Seam 2 vs CDI

- Seam 2
  - is an **application framework**
  - built to “fix holes/fill gaps” in specification (Java EE 5)
  - the idea of **“Reinvesting in Java EE”** → fixes should find way back into the next revision of the standards

- CDI
  - is a **JCP specification**
  - originally Web Beans
  - version 1.0 (JSR 299) is a part of Java EE 6 (JSR 316)
  - implementations include:
    - Weld (RI)
    - Apache OpenWebBeans
    - CanDI
  - Seam 3 is a set of modules which extend/use CDI

Seam3

Seam3

Seam3 *is a collection of modules and developer tooling tailored for* Java EE 6 *application development, with* CDI *as the central piece.*

SeamPersistence

SeamMail

SeamSocial

SeamRest

SeamJMS

SeamSecurity

SeamRemoting

SeamJCR

SeamFaces

SeamInternational

SeamSpring

Solder

CDI

Servlet

Java EE 6

Solder

By Kevin L Neff (CC-by 2.0)
http://en.wikipedia.org/wiki/File:Propane_torch_soldering_copper_pipe.jpg

JBoss Solder

- “Weld Extensions”
- Features
  - Utilities for CDI Extension writers
  - Enhancements of the CDI Model
  - Resource Injection
  - Logging
  - XML Config
  - Servlet
  - Exception Handling

Solder Enhancements of the CDI Model

- @Veto
- @Requires
- @Exact
- @Unwrap
- @DefaultBean
- Generic Beans

@Veto

- Veto the processing of the type.
  - Any beans or observer methods defined by this class will not be installed.
  - When placed on package, all beans in the package are prevented from being installed.
- To be in CDI 1.1

```
@Veto
class Utilities {
  ...
}
```

@Requires

- Ignore type if the class dependencies cannot be satisfied.

```
@Requires({"org.joda.time.DateTimeZone"})
public class DefaultDateTimeZoneProducer {
   ...
}
```

@Exact

- Selects an exact implementation of the injection point type to inject.

```
interface PaymentService {...}

class ChequePaymentService
   implements PaymentService {...}

class CardPaymentService
   implements PaymentService {...}

class PaymentProcessor {
   @Inject @Exact(CardPaymentService.class)
   PaymentService paymentService;
   ...
}
```

@Unwraps 1/3

```
@SessionScoped
class PermissionManager {
   Permission permission;
   void setPermission(Permission permission) {
      this.permission=permission;
   }

   @Unwraps @Current
   Permission getPermission() {
      return this.permission;
   }
}

@SessionScoped
class SecurityManager {
   @Inject @Current
   Permission permission;

   boolean checkAdminPermission() {
      return permission.getName().equals("admin");
   }
}
```

@Unwraps 2/3

```
public class SomeSensitiveOperation {
   @Inject
   PermissionManager permissionManager;

   public void perform() {
      try {
         permissionManager.setPermission(Permissions.ADMIN);
         // Do some sensitive operation
      } finally {
         permissionManager.setPermission(Permissions.USER);
      }
   }
}
```

@Unwraps 3/3

- Unwrapping producer
- “Self managed” life cycle
  - changes to the unwrapped object are immediately visible to all clients.
  - Unwrapping producer invoked on every invocation on the injected proxy
- Must be proxyable

@DefaultBean 1/2

- ```
  @DefaultBean(FunctionMapper.class)
  @Mapper
  class FunctionMapperImpl extends FunctionMapper {
     @Override
     public Method resolveFunction(String prefix, String localName) {
        return null;
     }
  }
  ```

- ```
  class FunctionMapperProvider {
     @Produces
     @Mapper
     FunctionMapper produceFunctionMapper() {
        return
  FacesContext.getCurrentInstance().getELContext().getFunctionMapper();
     }
  }
  ```

@DefaultBean 2/2

- Similar use case as CDI alternatives
  - No need to modify all the beans.xmls
- Default Producer
  - ```
    class CacheManager {
        @Produces
        @DefaultBean(value = Cache.class)
        Cache getCache() {
            ...
        }
    }
    ```
    ```

Solder Resource Injection

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

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

Solder Logging

```
import org.jboss.solder.logging.Logger;

public class LogService {
    @Inject
    private Logger log;

    public void logMessage() {
        log.info("Hey sysadmins!");
    }
}
```

Typed Logger

```
import org.jboss.solder.messages.Message;
import org.jboss.solder.logging.Log;
import org.jboss.solder.logging.MessageLogger;

@MessageLogger
public interface TrainSpotterLog {
    @Log @Message("Spotted %s diesel trains")
    void dieselTrainsSpotted(int number);
}


@Inject @Category("trains")
private TrainSpotterLog log;
```

Typed Message Bundles

```
import org.jboss.solder.messages.Message;
import org.jboss.solder.messages.MessageBundle;

@MessageBundle
public interface TrainMessages {
    @Message("No trains spotted due to %s")
    String noTrainsSpotted(String cause);
}


@Inject @MessageBundle
private TrainMessages messages;
```

Typed Messages Localization

- TrainMessages.i18n_fr.properties

```
noTrainsSpotted=pas de trains repéré en raison de %s
```

- Solder Annotation Processor

```
<dependency>
    <groupId>org.jboss.solder</groupId>
    <artifactId>solder-tooling</artifactId>
    <scope>provided</scope>
</dependency>
```

Solder XML Configuration

- META-INF/seam-beans.xml
- Creating/replacing/modifying beans

```
class Report {
  String filename;

  @Inject
  Datasource datasource;

  //getters and setters
}

interface Datasource {
   public Data getData();
}

@SalesQualifier
class SalesDatasource
 implements Datasource {

  public Data getData()
  {
   //return sales data
  }
}

class BillingDatasource
 implements Datasource {

  public Data getData()
  {
   //return billing data
  }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans
xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns:s="urn:java:ee"
xmlns:r="urn:java:org.example.reports">

  <r:Report>
    <s:modifies/>
    <r:filename>sales.jrxml<r:filename>
    <r:datasource>
      <r:SalesQualifier/>
    </r:datasource>
  </r:Report>

  <r:Report filename="billing.jrxml">
    <s:replaces/>
    <r:datasource>
      <s:Inject/>
      <s:Exact>
org.example.reports.BillingDatasource
     </s:Exact>
    </r:datasource>
  </r:Report>
</beans>
```

Solder XML Initial Field Values

- `<r:MyBean company="Red Hat Inc" />`
- ```
  <r:MyBean>
    <r:company>Red Hat Inc</r:company>
  </r:MyBean>
  ```
- ```
  <r:MyBean>
    <r:company>
     <s:value>Red Hat Inc<s:value>
     <r:SomeQualifier/>
    </r:company>
  </r:MyBean>
  ```

Solder XML Inline Bean

```
<my:Knight>
  <my:sword>
    <value>
      <my:Sword type="sharp"/>
    </value>
  </my:sword>
  <my:horse>
    <value>
      <my:Horse>
        <my:name>
          <value>billy</value>
        </my:name>
        <my:shoe>
          <Inject/>
        </my:shoe>
      </my:Horse>
    </value>
  </my:horse>
</my:Knight>
```

Solder XML ...

- Configuring
  - methods
  - bean constructor
  - meta annotations
    - (e.g. existing annotation into a qualifier)
  - Virtual producer fields
    - ```
      <s:EntityManager>
        <s:Produces/>
        <s:PersistenceContext unitName="customerPu" />
      </s:EntityManager>
      ```
- Demo
  - `http://princessrescue-seam3.rhcloud.com`
      ```

Solder Servlet Integration

- Producers for Servlet objects
  - ```
    @Inject
    private HttpServletRequest httpRequest;
    ```
- Propagating Servlet events to CDI
  - ```
    public void setupEncoding(
     @Observes @Initialized ServletResponse res,
                            ServletRequest req) {
      ...
    }
    ```
- Forwarding uncaught exceptions to Solder's exception handling chain

Solder Servlet Exception Handling

- ```
  @HandlesExceptions
  public class Handler {
    void handleAccountNotFound(
      @Handles
      CaughtException<AccountNotFound> caught,
      HttpServletResponse response) {
        System.out.println("Account " +
           caught.getException().getId() +
           " not found");
        evt.markHandled();
    }
  }
  ```
- ```
  @ExceptionHandled
  public void selectAccount(String id) {
    ...
    throw AccountNotFound(id);
  }
  ```

Solder Servlet Exception Handling

- Starting the exception handling process
  - Firing an event
    - @Inject
      private Event<ExceptionToCatch> catchEvent;
  - @ExceptionHandled interceptor
  - Module integration
    - Solder Servlet, Seam REST, ...

SeamPersistence

Seam Persistence

- Motivation
  - LazyInitializationException
  - Create/modify data in a conversation
    - Optimistic transaction with multiple requests
    - No need to merge()
    - Commit at conversation end
  - Seam-managed Extended Persistence Context
  - Manual flush mode
    - Non-portable hibernate-specific feature!

Seam Persistence

- Conversation-scoped extended persistence context
  - @ExtensionManaged
    @Produces
    @PersistenceUnit
    @ConversationScoped
    EntityManagerFactory producerField;
  - @Inject
    EntityManager em;

Public Domain
http://en.wikipedia.org/wiki/File:VeniceShopWindow.jpg

Seam Faces

- @Inject'able Faces artifacts
- JSF Events Propagation
- Access to other JSF scopes
  - @ViewScoped
  - @Inject Flash flash;
- @RenderScoped
- @Begin/@End conversation annotations
- Messages
- Seam Faces Components
- Injection into validators and converters

Seam Faces ...

- ViewConfig
  - Seam Security integration
  - PrettyFaces URL rewriting

Seam Faces @ViewConfig

```
@ViewConfig
public interface MyAppViewConfig {
    static enum Pages {
        @ViewPattern("/admin.xhtml")
        @Admin
        @RestrictAtPhase({PhaseIdType.RESTORE_VIEW,
PhaseIdType.INVOKE_APPLICATION})
        ADMIN,

        @UrlMapping(pattern = "/item/#{id}/")
        @ViewPattern("/item.xhtml")
        @Owner
        ITEM,

        @FacesRedirect
        @ViewPattern("/*")
        @AccessDeniedView("/denied.xhtml")
        @LoginView("/login.xhtml")
        ALL;
    }
}
```

Seam Faces s:validateForm

- ```
  <h:form id="locationForm">
     <h:inputText id="city" value="#{bean.city}" />
     <h:inputText id="zip" value="#{bean.zip}" />
     <h:commandButton id="submit" value="Submit" action="#{bean.submitPost}" />
     <s:validateForm validatorId="locationValidator" />
  </h:form>
  ```
- ```
  @FacesValidator("locationValidator")
  public class LocationValidator implements Validator {
    @Inject
    Directory directory;

    @Inject
    @InputField
    private Object city;

    @Inject
    @InputField
    private ZipCode zip;

    @Override
    public void validate(final FacesContext context, final UIComponent comp, final
  Object values) throws ValidatorException   {
        if(!directory.exists(city, zip)) {
           throw new ValidatorException(
              new FacesMessage("Sorry, that location is not in our database."));
        }
     }
  }
  ```

s:viewAction

- http://localhost:8080/blog/entry.jsf?id=10
- ```
  <f:metadata>
   <f:viewParam name="id" value="#{blogManager.entryId}"/>
   <s:viewAction action="#{blogManager.loadEntry}"/>
  </f:metadata>
  ```
- ```
  <navigation-rule>
    <from-view-id>/entry.xhtml</from-view-id>
    <navigation-case>
      <from-action>#{blogManager.loadEntry}</from-action>
      <if>#{empty entry}</if>
      <to-view-id>/home.xhtml</to-view-id>
    </navigation-case>
  </navigation-rule>
  ```

SeamSecurity

Seam Security

- Authentication
  - credentials (username/password)
- Identity Management
  - PicketLink
  - users, groups, roles
- External Authentication
  - OpenID
- Authorization
- Events
  - LoggedInEvent, ...

Seam Security Identity Bean

- Identity bean
  - `<h:commandButton action="#{identity.login}" value="Log in"/>`
- Credentials bean
  - username/credential
  - password
  - `<h:inputText id="name" value="#{credentials.username}"/>`
    `<h:inputSecret id="pw" value="#{credentials.password}"/>`

Seam Security Authentication

- Authenticators
  - Solder XML Config
  - ```
    <security:IdentityImpl>
       <s:modifies/>
       <security:authenticatorClass>
         com.acme.MyCustomAuthenticator
      </security:authenticatorClass>

    </security:IdentityImpl>
    ```
  - org.jboss.seam.security.jaas.JaasAuthenticator
    org.jboss.seam.security.management.IdmAuthenticator
    org.jboss.seam.security.external.openid.OpenIdAuthenticator
    ```

Seam Security Custom Authenticator

- ```
  public class SimpleAuthenticator extends BaseAuthenticator
  implements Authenticator {
     @Inject
     Credentials credentials;

     @Override
     public void authenticate() {
        if ("demo".equals(credentials.getUsername()) &&
              credentials.getCredential() instanceof
                  PasswordCredential &&
              "demo".equals(((PasswordCredential)
                  credentials.getCredential()).getValue()))  {
           setStatus(AuthenticationStatus.SUCCESS);
           setUser(new SimpleUser("demo"));
        }
     }
  }
  ```
  ```

Seam Security Authorization

- Security Binding
  - ```
    @SecurityBindingType
    @Retention(RetentionPolicy.RUNTIME)
    @Target({ElementType.TYPE, ElementType.METHOD})
    public @interface Admin {}
    ```
- ```
  @Secures
  @Admin
  public boolean isAdmin(Identity identity) {
    return ...
  }
  ```
- AuthorizationException
- @LoggedIn

Seam Security SecurityInterceptor

- SecurityInterceptor

```
<interceptors>
  <class>
    org.jboss.seam.security.SecurityInterceptor
  </class>
</interceptors>
```

Other Seam Modules

- Seam International
- Seam JMS
- Seam JCR
- Seam Mail
- Seam Remoting
- Seam REST
- Seam Social

Seam International

- Inject
  - Timezone
  - Application/Client locale
  - Messages
    - ```
      @Inject
      Messages messages;

      ...
      messages.error(“Show this message in a UI”);
      ```
      ```

Seam JMS

- Injection of JMS Resources
- Bridge CDI Event bus over JMS
  - Inbound
    - Routes CDI events to JMS destinations
  - Outbound
    - Fires CDI events based on the reception of JMS messages

Seam JCR

- Content Repository API for Java
- ```
  @Inject
  Session session;
  ```

- ```
  public void observeAdded(@Observes @NodeAdded Event evt)
  {
        // Called when a node is added
  }
  ```

Seam Mail

- Java Mail API
- Configuration through seam-beans.xml

Seam Remoting

- Remotely accessing CDI Beans trough AJAX
- ```
  @Named
  public class HelloAction implements HelloLocal {
      @WebRemote
      public String sayHello(String name) {
          return "Hello, " + name;
      }
  }
  ```
- ```
  <script type="text/javascript"
  src="seam/resource/remoting/interface.js?helloAction”/>
  ```
- ```
  function sayHello() {
      var name = prompt("What is your name?");
      Seam.createBean("helloAction").sayHello(name, sayHelloCallback);
  }
  function sayHelloCallback(result) {
      alert(result)
  }
  ```
  ```

Seam REST

- JAX-RS
- Integration with Solder Exception handling
- Bean Validation Integration
- Templating
  - FreeMarker, Apache Velocity

Seam Social

- API for oAuth services
  - Facebook, Twitter, LinkedIn

Future of Seam

- DeltaSpike
  - Apache MyFaces CODI, Seam3
  - “...to create a de-facto standard of extensions that is developed and maintained by the Java community, and to act as an incubator for features that may eventually become part of the various Java SE and EE-related specifications.”
  - Apache Incubation
  - https://cwiki.apache.org/confluence/display/DeltaSpike/Index

The End