Web Beans: Java Contexts and Dependency Injection

The new standard for dependency injection and contextual state management

Gavin King
JSR-299 specification lead
Red Hat Middleware LLC

Pete Muir
Web Beans (JSR-299 Reference Implementation) lead
Red Hat Middleware LLC

David Allen
######: Nicola Benaglia, Francesco Milesi
Spanish Translation: Gladys Guerrero
Red Hat Middleware LLC
Korean Translation: Eun-Ju Ki,

Red Hat Middleware LLC
Traditional Chinese Translation: Terry Chuang
Red Hat Middleware LLC
Simplified Chinese Translation: Sean Wu
Kava Community

Note

JSR-299 has recently changed its name from "Web Beans" to "Java Contexts and Dependency Injection". The reference guide still refers to JSR-299 as "Web Beans" and the JSR-299 Reference Implementation as the "Web Beans RI". Other documentation, blogs, forum posts etc. may use the new nomenclature, including the new name for the JSR-299 Reference Implementation - "Web Beans".

You'll also find that some of the more recent functionality to be specified is missing (such as producer fields, realization, asynchronous events, XML mapping of EE resources).

I. ä½¿ç#¨æ¦#å¿µä¸#ç##ç#©ä»¶

Web Beans#JSR-299#### Java EE ##################Web Bean ###### Java ####### JavaBeans ## Enterprise Java Beans#################interaction model######### Java EE #######Programming Model##Web Bean ######

- ######################## *contexts#*
- ###### *dependency injection*######## typesafe############
- ## *event notification*##################
- ### *interceptors*####################### *decorator*############################

contextual ###################### API

- #############
- ##############
- #########multithread##
- #########
- ###############
- ############################
- ######## indirection layer ###################
- ###################

Web Bean ######### Web Bean ### Web Bean ################## Web Bean ##

######################### *loose-coupling*########

- *event notifications*######## decouple producer # event consumer#######
- *interceptors*############# decouple #######
- *decorators*###################

######Web Bean ### *typesafe* ###############Web Bean ###############identifier#################### XML ######################Web Bean ### Java ######### typing ########## *binding annotations*############## Web Bean##################################

Web Bean ################ Java EE #########

- ## JavaBean#
- ## EJB###
- ## Servlet#

Web Bean ################ Java EE ########################### Web Bean ######## Web Bean ############# Web Bean #####

Web Bean ######## Java framework### Seam#Guice ## Spring########Web Bean ############ Seam ##### typesafe## Spring # stateful ####### XML ############################ Guice ###

######Web Bean ##### Java EE ######### EJB Lite ### Java SE ##### JCP ###

Web Bean

################ Web Bean ######### Web Bean ## Web Bean ##################### Web Bean ####

1.1. ##### Web Bean

################################## constructor # java class ### Web Bean######## JavaBean###### EJB 3-style # session bean #### Web Bean########### JavaBean # EJB ####### Web Bean ###################### Web Bean ### # ####### Web Bean #### Web Bean XML ###################interceptor######decorator# # ##############

####################### Java class#### class ###############

```
public class SentenceParser {
    public List<String
> parse(String text) { ... }
}
```

class ########### session bean

```
@Stateless
public class SentenceTranslator implements Translator {
    public String translate(String sentence) { ... }
}
```

`Translator` ######

```
@Local
public interface Translator {
    public String translate(String sentence);
}
```

####################### class################### Web Bean ##

```
public class TextTranslator {

    private SentenceParser sentenceParser;
```

```
    private Translator sentenceTranslator;

    @Initializer
    TextTranslator(SentenceParser sentenceParser, Translator sentenceTranslator) {
        this.sentenceParser = sentenceParser;
        this.sentenceTranslator = sentenceTranslator;
    }

    public String translate(String text) {
        StringBuilder sb = new StringBuilder();
        for (String sentence: sentenceParser.parse(text)) {
            sb.append(sentenceTranslator.translate(sentence));
        }
        return sb.toString();
    }

}
```

########### `TextTranslator` # instance ## Web Bean#Servlet ## EJB ###### instance#

```
@Initializer
public setTextTranslator(TextTranslator textTranslator) {
    this.textTranslator = textTranslator;
}
```

############## Web Bean ###### method ##### instance#

```
TextTranslator tt = manager.getInstanceByType(TextTranslator.class);
```

######`TextTranslator` ######### constructor######## Web Bean ######### constructor # class ######## `@Initializer` # constructor ########## Web Bean#

#########`@Initializer` ##############`@Initializer` ###### Web Bean # constructor ## method ### Web Bean ###### Web Bean ##### constructor # method#Web Bean ####### Web Bean #### constructor # method #####

#########Web Bean ######################## Web Bean ################## `Translator` ## # # `SentenceTranslator` EJB ##### # Web Bean ######### `UnsatisfiedDependencyException`########### `Translator` ##### Web Bean ######### `AmbiguousDependencyException`#

1.2. Web Bean

Web Bean

Web Bean ############## class#Web Bean ## Java ################# Unified EL ####Web Bean ##########Web Bean ######### Web Bean ############## Web Bean ### *stateful*##### # *contextual*##### ##Web Bean ######lifecycle##### Web Bean ########

########contextual################## Web Bean ################## bean instance################## session bean########## servlet ##### bean########### Web Bean ################ Web Bean################### Web Bean ### instance#

######################### session bean ################ instance ### instance #########Web Bean # *scope* ####

- Web Bean ### instance #######
- ####### Web Bean ### instance ####

Web Bean ################## Web Bean # scope ### *active context*### context ###################### Web Bean # request scoped ############################## Web Bean # session scoped ######################## application scoped ####

context ######### Web Bean######## Web Bean instance##### context ########## instance#

Contextual model ################## Web Bean ########################## Web Bean ########################Web Bean ################ Web Bean ####################Web Bean ######loosely coupled######

- ############ API #####
- ############ decouple #

########## Web Bean ######### Web Bean ########## API ########### scope## Web Bean#####Web Bean ############## Web Bean ############### *# 4.2, "####"* ####

######## Web Bean ###### Web Bean### Servlet ###### Bean ####### # ######### contextual object # ######### Web Bean ####

#####

Web Bean ####

- #######API ###API type#
- ###############binding annotation type#
- ## scope

- #######deployment type#
- ########## Web Bean ##
- ##########interceptor binding type#
- ## Web Bean ###Web Bean implementation#

########## Web Bean ##########

1.2.1. API

Web Bean ##########dependency injection###### Web Bean ######################contract######### Web Bean ##########

- ## API #####
- #######

API ####### class ###### Web Bean ## EJB session bean ##### API #### `@Local` #### bean-class # local view############################ API ######

################## `@BindingType` ################### `PaymentProcessor` API #### `@CreditCard` #####

```
@CreditCard PaymentProcessor paymentProcessor
```

############################ `@Current` ######

Web Bean ##################### API ########## Web Bean##### Web Bean ###

Web Bean ###### `@CreditCard` ##### `PaymentProcessor` ## API

```
@CreditCard
public class CreditCardPaymentProcessor
   implements PaymentProcessor { ... }
```

Web Bean ################################### `@Current`#

Web Bean ############## *resolution algorithm*###################### Web Bean ######## container ########## *# 4, #####Dependency injection#* ######

1.2.2. #####Deployment type#

Deployment type ##########deployment scenario###### Web Bean ######################## `@Mock#@Staging # @AustralianTaxLaw`################### Web Bean ################## Web Bean ########

Web Bean ########## `@Production`############################## Web Bean ### `@Production`

################ `SentenceTranslator` ## Web Bean ######mock ####

```
@Mock
public class MockSentenceTranslator implements Translator {
   public String translate(String sentence) {
      return "Lorem ipsum dolor sit amet";
   }
}
```

############ `@Mock` ## deployment type ### `MockSentenceTranslator` ########## `@Mock` # Web Bean #######

 # 4.2, "####"

1.2.3. Scope

Scope ### Web Bean instance ############Web Bean # context model ############ scope############ scope ###### Web Bean ####### Web Bean #######Scope ##############

################ *session scoped* # Web Bean#

```
@SessionScoped
public class ShoppingCart { ... }
```

session scoped # Web Bean # instance ########## session############ session # context

#####Web Bean ###### *dependent pseudo-scope* ### scope##### scope # Web Bean ##

 # 5, Scope # context ##### scope#

1.2.4. Web Bean ### Unified EL

Web Bean ################## Unified EL ######## Web Bean ########

```
@SessionScoped @Named("cart")
public class ShoppingCart { ... }
```

########## JSF # JSP ##### Web Bean#

```
<h:dataTable value="#{cart.lineItems}" var="item">
   ....
</h:dataTable
>
```

Web Bean

```
@SessionScoped @Named
public class ShoppingCart { ... }
```

############## `shoppingCart` # ####### class ##################

1.2.5.

Web Bean ## EJB 3 ############## EJB bean #### Java # class #######Web Bean ################## EJB bean #### Web Bean#

########## `@Interceptors` #########interceptor#class#

```
@SessionScoped
@Interceptors(TransactionInterceptor.class)
public class ShoppingCart { ... }
```

interceptor binding type############################

```
@SessionScoped @Transactional
public class ShoppingCart { ... }
```

 # 7, ####Interceptor# # *# 8, ####Decorators#* ######## Web Bean

1.3. ###### Web Bean#

JavaBean#EJB ###### Java class #### Web Bean### Web Bean

1.3.1. ### Web Bean

Web Bean ################ Java class ### *simple*####Web Bean#

- ### EJB#Servlet # JPA ##### EE container #######

- ################static inner class##
- ###############
- ####### constructor######## `@Initializer` # constructor#

######## JavaBean ###### Web Bean#

Web Bean ################## Web Bean # API ###Class ### superclass #### API

1.3.2. #### Web Bean

Web Bean ######## EJB 3 ### session # singleton # bean ####### Web Bean###### bean #### Web Bean # ############### # ########## Web Bean ###############dependency injection#######interceptor##

wildcard ############### Web Bean ########### superinterface ###### Web Bean # API #### EJB bean ## bean class local view ####### bean class ###### superclass ##### API

session bean ######### remove method ######## `@Destructor` # remove method#Web Bean ######## method ################## session bean instance### method ###### Web Bean # *destructor* method#

```
@Stateful @SessionScoped
public class ShoppingCart {

   ...

   @Remove
   public void destroy() {}

}
```

############## Web Bean ###### Web Bean ######## EJB ################

- method ############
- #####concurrency management##
- #### session bean # instance ### passivation ##### session bean ######instance-pooling##
- ############
- ######## method#

################## Web Bean###################### Web Bean ###

Web Bean##### session # application scoped # Web Bean#########concurrent access#####EJB 3.1 #########concurrency management############ session # application scoped # Web Bean #### EJB#

###################### Web Bean ####### passivation ###### EJB `@Stateless`/`@Stateful`/`@Singleton` ######### container #########

################ method #########method ############# method ###### method ############

Web Bean ################## `@Stateless`#`@Stateful` ## `@Singleton` ########## EJB

1.3.3. Producer method

producer method ##### context ### instance ### Web Bean ######## Web Bean instance ### method#Producer method #############instantiation########## Web Bean ##########

```
@ApplicationScoped
public class Generator {

   private Random random = new Random( System.currentTimeMillis() );

   @Produces @Random int next() {
      return random.nextInt(100);
   }

}
```

producer method ######### Web Bean #######

```
@Random int randomNumber
```

method #################/######## producer method # API ########### class ######## superclass #### API ###

producer method

```
@Produces @RequestScoped Connection connect(User user) {
   return createConnection( user.getId(), user.getPassword() );
}
```

producer method ###### *disposal methods*#

```
void close(@Disposes Connection connection) {
   connection.close();
}
```

disposal method ############ Web Bean

6, *Producer method* ##### producer method#

1.3.4. JMS ###endpoints#

###JMS ###queue#####topic##### Web Bean#Web Bean ############################ JMS ############### # 13.4, *“JMS ##”* ##### JMS ###

JSF

################################ JSF ############/############### Web Bean ###################

```
@Named @RequestScoped
public class Credentials {

    private String username;
    private String password;

    public String getUsername() { return username; }
    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }
    public void setPassword(String password) { this.password = password; }

}
```

Web Bean ##### JSF

```
<h:form>
    <h:panelGrid columns="2" rendered="#{!login.loggedIn}">
        <h:outputLabel for="username"
>Username:</h:outputLabel>
        <h:inputText id="username" value="#{credentials.username}"/>
        <h:outputLabel for="password"
>Password:</h:outputLabel>
        <h:inputText id="password" value="#{credentials.password}"/>
    </h:panelGrid>
    <h:commandButton value="Login" action="#{login.login}" rendered="#{!login.loggedIn}"/>
    <h:commandButton value="Logout" acion="#{login.logout}" rendered="#{login.loggedIn}"/>
</h:form
>
```

####### session ### Web Bean ######## Web Bean ###################### `User` ## entity ##### Web Bean#

```
@SessionScoped @Named
public class Login {
```

```
    @Current Credentials credentials;
    @PersistenceContext EntityManager userDatabase;

    private User user;

    public void login() {

        List<User
> results = userDatabase.createQuery(
            "select u from User u where u.username=:username and u.password=:password")
            .setParameter("username", credentials.getUsername())
            .setParameter("password", credentials.getPassword())
            .getResultList();

        if ( !results.isEmpty() ) {
            user = results.get(0);
        }

    }

    public void logout() {
        user = null;
    }

    public boolean isLoggedIn() {
      return user!=null;
    }

    @Produces @LoggedIn User getCurrentUser() {
        return user;
    }

}
```

###`@LoggedIn` ########

```
@Retention(RUNTIME)
@Target({TYPE, METHOD, FIELD})
@BindingType
public @interface LoggedIn {}
```

####### Web Bean ##############

```
public class DocumentEditor {

    @Current Document document;
    @LoggedIn User currentUser;
    @PersistenceContext EntityManager docDatabase;

    public void save() {
        document.setCreatedBy(currentUser);
        docDatabase.persist(document);
    }

}
```

######### Web Bean #######Programming Model############################ Web Bean #####Web Beans dependency injection##

Web Beans, the Reference Implementation of JSR-299

The Web Beans is being developed at *the Seam project* [http://seamframework.org/WebBeans]. You can download the latest developer release of Web Beans from the *the downloads page* [http://seamframework.org/Download].

Web Beans comes with a two deployable example applications: `webbeans-numberguess`, a war example, containing only simple beans, and `webbeans-translator` an ear example, containing enterprise beans. There are also two variations on the numberguess example, the tomcat example (suitable for deployment to Tomcat) and the jsf2 example, which you can use if you are running JSF2. To run the examples you'll need the following:

- the latest release of Web Beans,
- JBoss AS 5.0.1.GA, or
- Apache Tomcat 6.0.x, and
- Ant 1.7.0#

3.1. Using JBoss AS 5

You'll need to download JBoss AS 5.0.1.GA from *jboss.org* [http://www.jboss.org/jbossas/downloads/], and unzip it. For example:

```
$ cd /Applications
$ unzip ~/jboss-5.0.1.GA.zip
```

Next, download Web Beans from *seamframework.org* [http://seamframework.org/Download], and unzip it. For example

```
$ cd ~/
$ unzip ~/webbeans-$VERSION.zip
```

######### Web Bean ## JBoss ########## `jboss-as/build.properties` #### `jboss.home` ######

```
jboss.home=/Applications/jboss-5.0.1.GA
```

To install Web Beans, you'll need Ant 1.7.0 installed, and the `ANT_HOME` environment variable set. For example:

```
$ unzip apache-ant-1.7.0.zip
$ export ANT_HOME=~/apache-ant-1.7.0
```

Then, you can install the update. The update script will use Maven to download Web Beans automatically.

```
$ cd webbeans-$VERSION/jboss-as
$ ant update
```

###################

##

The build scripts for the examples offer a number of targets for JBoss AS, these are:

- `ant restart` - ###########
- `ant explode` - ####################
- `ant deploy` - # jar #######
- `ant undeploy` - ##########
- `ant clean` - ####

numberguess

```
$ cd examples/numberguess
ant deploy
```

Start JBoss AS:

```
$ /Application/jboss-5.0.0.GA/bin/run.sh
```

##

If you use Windows, use the `run.bat`script.

############### *http://localhost:8080/webbeans-numberguess* #######

Web Beans includes a second simple example that will translate your text into Latin. The numberguess example is a war example, and uses only simple beans; the translator example is an ear example, and includes enterprise beans, packaged in an EJB module. To try it out:

```
$ cd examples/translator
ant deploy
```

############## *http://localhost:8080/webbeans-translator*#

3.2. Using Apache Tomcat 6.0

You'll need to download Tomcat 6.0.18 or later from *tomcat.apache.org* [http://tomcat.apache.org/download-60.cgi], and unzip it. For example:

```
$ cd /Applications
$ unzip ~/apache-tomcat-6.0.18.zip
```

Next, download Web Beans from *seamframework.org* [http://seamframework.org/Download], and unzip it. For example

```
$ cd ~/
$ unzip ~/webbeans-$VERSION.zip
```

Next, we need to tell Web Beans where Tomcat is located. Edit `jboss-as/build.properties` and set the `tomcat.home` property. For example:

```
tomcat.home=/Applications/apache-tomcat-6.0.18
```

##

The build scripts for the examples offer a number of targets for Tomcat, these are:

- `ant tomcat.restart` - deploy the example in exploded format
- `ant tomcat.explode` - update an exploded example, without restarting the deployment
- `ant tomcat.deploy` - deploy the example in compressed jar format
- `ant tomcat.undeploy` - remove the example from the server
- `ant tomcat.clean` - clean the example

To deploy the numberguess example for tomcat:

```
$ cd examples/tomcat
ant tomcat.deploy
```

Start Tomcat:

```
$ /Applications/apache-tomcat-6.0.18/bin/startup.sh
```

##

If you use Windows, use the `startup.bat`script.

############### *http://localhost:8080/webbeans-numberguess* #######

3.3. Using GlassFish

TODO

3.4. numberguess

numberguess ########## 10 ######### 1 # 100

numberguess ##### Web Bean ############## Facelet JSF ######## war##############

############## `WEB-INF/` ############### `WebContent` ################ JSF ### Facelet # `faces-config.xml`#

```
<?xml version='1.0' encoding='UTF-8'?>
<faces-config version="1.2"
```

```
          xmlns="http://java.sun.com/xml/ns/javaee"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/
javaee/web-facesconfig_1_2.xsd">

   <application>
      <view-handler
>com.sun.facelets.FaceletViewHandler</view-handler>
   </application>

</faces-config
>
```

`web-beans.xml` ################# Web Bean

`web.xml`#

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app version="2.5"
   xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  http://java.sun.com/xml/ns/javaee/
web-app_2_5.xsd">

  <display-name
>Web Beans Numbergues example</display-name>

  <!-- JSF -->
                                                    (1)

  <servlet>
     <servlet-name
>Faces Servlet</servlet-name>
     <servlet-class
>javax.faces.webapp.FacesServlet</servlet-class>
                                                    (2)
     <load-on-startup
>1</load-on-startup>
  </servlet>

  <servlet-mapping>
                                                    (3)
     <servlet-name
```

```
>Faces Servlet</servlet-name>
    <url-pattern
>*.jsf</url-pattern>
  </servlet-mapping>
                                                    (4)

  <context-param>
    <param-name
>javax.faces.DEFAULT_SUFFIX</param-name>
                                                    (5)
    <param-value
>.xhtml</param-value>
  </context-param>

  <session-config>
    <session-timeout
>10</session-timeout>
  </session-config>

</web-app
>
```

1. Enable and load the JSF servlet
2. Configure requests to `.jsf` pages to be handled by JSF
3. Tell JSF that we will be giving our source files (facelets) an extension of `.xhtml`
4. Configure a session timeout of 10 minutes

#

Whilst this demo is a JSF demo, you can use Web Beans with any Servlet based web framework.

Let's take a look at the Facelet view:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://java.sun.com/jsf/facelets"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:s="http://jboss.com/products/seam/taglib">
```

```
 <ui:composition template="template.xhtml">                                    1
   <ui:define name="content">
     <h1
>Guess a number...</h1>

     <h:form id="NumberGuessMain">                                             2
       <div style="color: red">
         <h:messages id="messages" globalOnly="false"/>
           <h:outputText id="Higher" value="Higher!" rendered="#{game.number gt game.guess
 and game.guess ne 0}"/>
         <h:outputText id="Lower" value="Lower!" rendered="#{game.number lt game.guess and
 game.guess ne 0}"/>
       </div>


       <div>                                                                   3
         I'm thinking of a number between #{game.smallest} and #{game.biggest}.
         You have #{game.remainingGuesses} guesses.
       </div>

       <div>

         Your guess:                                                           4
         <h:inputText id="inputGuess"
                   value="#{game.guess}"
                   required="true"
                   size="3"

                   disabled="#{game.number eq game.guess}">                    5
           <f:validateLongRange maximum="#{game.biggest}"
                        minimum="#{game.smallest}"/>

         </h:inputText>                                                        6
         <h:commandButton id="GuessButton"
                     value="Guess"
                     action="#{game.check}"
                     disabled="#{game.number eq game.guess}"/>
       </div>
       <div>
                  <h:commandButton  id="RestartButton"  value="Reset"  action="#{game.reset}"
 immediate="true" />
       </div>
     </h:form>
   </ui:define>
 </ui:composition>
```

```
</html
>
```

1. Facelets is a templating language for JSF, here we are wrapping our page in a template which defines the header.
2. There are a number of messages which can be sent to the user, "Higher!", "Lower!" and "Correct!"
3. As the user guesses, the range of numbers they can guess gets smaller - this sentance changes to make sure they know what range to guess in.
4. This input field is bound to a Web Bean, using the value expression.
5. A range validator is used to make sure the user doesn't accidentally input a number outside of the range in which they can guess - if the validator wasn't here, the user might use up a guess on an out of range number.
6. And, of course, there must be a way for the user to send their guess to the server. Here we bind to an action method on the Web Bean.

4 ############################ `@Random`

```
@Target( { TYPE, METHOD, PARAMETER, FIELD })
@Retention(RUNTIME)
@Documented
@BindingType
public @interface Random {}
```

################ `@MaxNumber` #####

```
@Target( { TYPE, METHOD, PARAMETER, FIELD })
@Retention(RUNTIME)
@Documented
@BindingType
public @interface MaxNumber {}
```

`Generator` ##

```
@ApplicationScoped
public class Generator {

   private java.util.Random random = new java.util.Random( System.currentTimeMillis() );

   private int maxNumber = 100;
```

```
  java.util.Random getRandom()
  {
    return random;
  }

  @Produces @Random int next() {
    return getRandom().nextInt(maxNumber);
  }

  @Produces @MaxNumber int getMaxNumber()
  {
    return maxNumber;
  }

}
```

`Generator`

######## Web Bean # session ### `Game`#

########### `@Named` ############# JSF #### EL ##### bean########## constructor injection ## `FacesMessage` ######

```
package org.jboss.webbeans.examples.numberguess;


import javax.annotation.PostConstruct;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.webbeans.AnnotationLiteral;
import javax.webbeans.Current;
import javax.webbeans.Initializer;
import javax.webbeans.Named;
import javax.webbeans.SessionScoped;
import javax.webbeans.manager.Manager;

@Named
@SessionScoped
public class Game
{
  private int number;

  private int guess;
  private int smallest;
```

```
  private int biggest;
  private int remainingGuesses;

  @Current Manager manager;

  public Game()
  {
  }

  @Initializer
  Game(@MaxNumber int maxNumber)
  {
    this.biggest = maxNumber;
  }

  public int getNumber()
  {
    return number;
  }

  public int getGuess()
  {
    return guess;
  }

  public void setGuess(int guess)
  {
    this.guess = guess;
  }

  public int getSmallest()
  {
    return smallest;
  }

  public int getBiggest()
  {
    return biggest;
  }

  public int getRemainingGuesses()
  {
    return remainingGuesses;
  }
```

```
  public String check()
  {
    if (guess
>number)
    {
      biggest = guess - 1;
    }
    if (guess<number)
    {
      smallest = guess + 1;
    }
    if (guess == number)
    {
      FacesContext.getCurrentInstance().addMessage(null, new FacesMessage("Correct!"));
    }
    remainingGuesses--;
    return null;
  }

  @PostConstruct
  public void reset()
  {
    this.smallest = 0;
    this.guess = 0;
    this.remainingGuesses = 10;
    this.number = manager.getInstanceByType(Integer.class, new AnnotationLiteral<Random
>(){});
  }

}
```

3.4.1. The numberguess example for Tomcat

The numberguess for Tomcat differs in a couple of ways. Firstly, Web Beans should be deployed as a Web Application library in `WEB-INF/lib`. For your convenience we provide a single jar suitable for running Web Beans on Tomcat `webbeans-tomcat.jar`.

##

Of course, you must also include JSF and EL, as well common annotations (`jsr250-api.jar`) which a JEE server includes by default.

Secondly, we need to explicitly specify the Tomcat servlet listener (used to boot Web Beans, and control it's interaction with requests) in `web.xml`:

```
<listener>
  <listener-class>org.jboss.webbeans.environment.tomcat.Listener</listener-class>
</listener>
```

3.5.

##############################

########### ear #### EJB########### numberguess ########

#

EJB 3.1 # Jave EE 6 #### EJB ### war###############

########### ear ######## `webbeans-translator-ear` ####Maven ######### `application.xml`#

```
<plugin>
  <groupId
>org.apache.maven.plugins</groupId>
  <artifactId
>maven-ear-plugin</artifactId>
  <configuration>
    <modules>
      <webModule>
        <groupId
>org.jboss.webbeans.examples.translator</groupId>
        <artifactId
>webbeans-translator-war</artifactId>
        <contextRoot
>/webbeans-translator</contextRoot>
      </webModule>
    </modules>
  </configuration>
</plugin
>
```

######## context ################*http://localhost:8080/webbeans-translator*##

##

Maven ############### META-INF/application.xml#

```
<?xml version="1.0" encoding="UTF-8"?>
<application xmlns="http://java.sun.com/xml/ns/javaee"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                       xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  http://
java.sun.com/xml/ns/javaee/application_5.xsd"
             version="5">
  <display-name
>webbeans-translator-ear</display-name>
  <description
>Ear Example for the reference implementation of JSR 299: Web Beans</
description>

  <module>
    <web>
      <web-uri
>webbeans-translator.war</web-uri>
      <context-root
>/webbeans-translator</context-root>
    </web>
  </module>
  <module>
    <ejb
>webbeans-translator.jar</ejb>
  </module>
</application
>
```

Next, lets look at the war. Just as in the numberguess example, we have a `faces-config.xml` (to enable Facelets) and a `web.xml` (to enable JSF) in `WebContent/WEB-INF`.

################## facelet### numberguess ############## form ################

```
<h:form id="NumberGuessMain">

  <table>
    <tr align="center" style="font-weight: bold" >
      <td>
```

```
          Your text
        </td>
        <td>
          Translation
        </td>
      </tr>
      <tr>
        <td>
        <h:inputTextarea id="text" value="#{translator.text}" required="true" rows="5" cols="80" />
        </td>
        <td>
          <h:outputText value="#{translator.translatedText}" />
        </td>
      </tr>
    </table>
    <div>
      <h:commandButton id="button" value="Translate" action="#{translator.translate}"/>
    </div>

</h:form
>
```

##

########## ejb ## webbeans-translator-ejb## src/main/resources/META-INF ######## archive ###### Web Bean ## web-beans.xml#

################################## bean#SentenceParser # TextTranslator######### bean#TranslatorControllerBean # SentenceTranslator############ Web Bean ##########################

SentenceParser # TextTranslator ######## bean### TextTranslator ### constructor ####

```
public class TextTranslator {
  private SentenceParser sentenceParser;
  private Translator sentenceTranslator;

  @Initializer
  TextTranslator(SentenceParser sentenceParser, Translator sentenceTranslator)
  {
    this.sentenceParser = sentenceParser;
    this.sentenceTranslator = sentenceTranslator;
```

TextTranslator ###### bean##############- ##################

UI ## session bean##########

```
@Stateful
@RequestScoped
@Named("translator")
public class TranslatorControllerBean implements TranslatorController
{

   @Current TextTranslator translator;
```

bean ############ getter # setter#

stateful###### session bean########## remove method#

```
   @Remove
   public void remove()
   {

   }
```

bean #####Web Bean ######## remove ## method###############

That concludes our short tour of the Web Beans examples. For more on Web Beans , or to help out, please visit *http://www.seamframework.org/WebBeans/Development*.

############ - ##########################

#####Dependency injection#

Web Bean ################

Constructor parameter injection#

```
public class Checkout {

   private final ShoppingCart cart;

   @Initializer
   public Checkout(ShoppingCart cart) {
      this.cart = cart;
   }

}
```

Initializer method parameter injection#

```
public class Checkout {

   private ShoppingCart cart;

   @Initializer
   void setShoppingCart(ShoppingCart cart) {
      this.cart = cart;
   }

}
```

direct field injection#

```
public class Checkout {

   private @Current ShoppingCart cart;

}
```

Web Bean # instance

- ###Web Bean ###### Web Bean constructor#### Web Bean ### instance#
- ####Web Bean ######## Web Bean ##########
- ###Web Bean ####### Web Bean ### initializer method#
- ##########Web Bean # `@PostConstruct` method ######

EJB Bean ### Constructor parameter injection### EJB ## EJB container ####instantiate####### Web Bean ####

####### `@Current` ######Constructor # initializer method ##

Producer method ### parameter injection#

```
@Produces Checkout createCheckout(ShoppingCart cart) {
   return new Checkout(cart);
}
```

###observer method##### # 9, ###Events# #####disposal method ## destructor method #### parameter injection#

Web Bean ################ *typesafe resolution algorithm*#typesafe ######## Web Bean ###########Web Bean ##Typesafe ##################### Web Bean ######################### `UnsatisfiedDependencyException` ## `AmbiguousDependencyException` ######

############## Web Bean ##### API #####

- ######## *binding annotations* #########
- ############################## *deployment types* ##################
- ######### *deployment type precedence* #### API ######################### API ####

Web Bean ######### Web Bean

4.1.

############## API ### Web Bean###################### Web Bean ##########`PaymentProcessor` #########

```
@PayByCheque
```

```
public class ChequePaymentProcessor implements PaymentProcessor {
    public void process(Payment payment) { ... }
}
```

```
@PayByCreditCard
public class CreditCardPaymentProcessor implements PaymentProcessor {
    public void process(Payment payment) { ... }
}
```

`@PayByCheque` # `@PayByCreditCard` ######

```
@Retention(RUNTIME)
@Target({TYPE, METHOD, FIELD, PARAMETER})
@BindingType
public @interface PayByCheque {}
```

```
@Retention(RUNTIME)
@Target({TYPE, METHOD, FIELD, PARAMETER})
@BindingType
public @interface PayByCreditCard {}
```

Web Bean ################## Web Bean

field injection#

```
@PayByCheque PaymentProcessor chequePaymentProcessor;
@PayByCreditCard PaymentProcessor creditCardPaymentProcessor;
```

initializer method injection#

```
@Initializer
public void setPaymentProcessors(@PayByCheque PaymentProcessor
 chequePaymentProcessor,
                    @PayByCreditCard PaymentProcessor creditCardPaymentProcessor) {
  this.chequePaymentProcessor = chequePaymentProcessor;
  this.creditCardPaymentProcessor = creditCardPaymentProcessor;
}
```

constructor injection#

```
@Initializer
public Checkout(@PayByCheque PaymentProcessor chequePaymentProcessor,
                @PayByCreditCard PaymentProcessor creditCardPaymentProcessor) {
  this.chequePaymentProcessor = chequePaymentProcessor;
  this.creditCardPaymentProcessor = creditCardPaymentProcessor;
}
```

4.1.1. member

####### member#

```
@Retention(RUNTIME)
@Target({TYPE, METHOD, FIELD, PARAMETER})
@BindingType
public @interface PayBy {
   PaymentType value();
}
```

######member value #####

```
@PayBy(CHEQUE) PaymentProcessor chequePaymentProcessor;
@PayBy(CREDIT_CARD) PaymentProcessor creditCardPaymentProcessor;
```

`@NonBinding` ## member ## Web Bean

4.1.2.

####################

```
@Asynchronous @PayByCheque PaymentProcessor paymentProcessor
```

#################### Web Bean #####

4.1.3. ##### producer method

producer method

```
@Produces
```

```
@Asynchronous @PayByCheque
PaymentProcessor createAsyncPaymentProcessor(@PayByCheque PaymentProcessor
 processor) {
   return new AsynchronousPaymentProcessor(processor);
}
```

4.1.4.

Web Bean ######### `@Current`###################### Web Bean ########

################### `@Current`#

- ### field ####### field ####################
- ########################## Web Bean ##

4.2.

Web Bean #### *deployment type*######### deployment type ###################### deployment ## Web Bean#

############## `@Mock` # deployment type################################## Web Bean#

```
@Retention(RUNTIME)
 @Target({TYPE, METHOD})
 @DeploymentType
 public @interface Mock {}
```

######################### Web Bean#

```
public class ExternalPaymentProcessor {

   public void process(Payment p) {
      ...
   }

}
```

Web Bean ######### deployment type##### deployment type ##### `@Production`#

###############unit testing############################ mock ###

```
@Mock
```

```
public class MockPaymentProcessor implements PaymentProcessor {

   @Override
   public void process(Payment p) {
      p.setSuccessful(true);
   }

}
```

Web Bean

4.2.1. ## deployment type

Web Bean ######## deployment type#`@Production` # `@Standard`###################### deployment type # Web Bean ############## deployment type ## `web-beans.xml` ####### deployment ####### deployment type#

################################ `@Mock` ########

```
<WebBeans>
   <Deploy>
      <Standard/>
      <Production/>
      <test:Mock/>
   </Deploy>
</WebBeans
>
```

###Web Bean ###### deployment time ########## `@Production`#`@Standard` # `@Mock` # Web Bean#

`@Standard` ## deployment type ##### Web Bean ########## Web Bean################ Web Bean ##############

`@Production` ## deployment type ####### deployment type # Web Bean ### deployment type########

4.2.2. Deployment type

################ Web Bean ########## # `ExternalPaymentProcessor` # `MockPaymentProcessor` # ############################

```
@Current PaymentProcessor paymentProcessor
```

Web Bean ### `PaymentProcessor` ##hard-coded######################## deployment time

############### deployment type #######Deployment type ############ `web-beans.xml` ##################`@Mock` ###### `@Production` ###############

############ Web Bean #################API ################ Web Bean ################################ Web Bean ################ Web Bean ################################### `MockPaymentProcessor`#

################################lightweight## container ##### classpath ## class ############## class ################### XML ######Web Bean ##### XML # Web Bean ##########################deployment type ##### Web Bean #### XML ################################## Web Bean ###########deployment scenario##

4.2.3. ## deployment type

Deployment type ########################

- ##### `@Mock` # `@Staging` deployment type
- `@AustralianTaxLaw` ######## Web Bean
- `@SeamFramework`#`@Guice` ###### Web Bean ##### framework
- `@Standard` #### Web Bean ######## Web Bean

##################...

4.3. ########unsatisfied dependencies####

########### API ### Web Bean ##### deployment type #### Web Bean ################ Web Bean ##### typesafe ##########

`UnsatisfiedDependencyException` # `AmbiguousDependencyException`

`UnsatisfiedDependencyException`######### API ############### Web Bean ## # ######### API ########### Web Bean # deployment type

`AmbiguousDependencyException`############### API ###################### deployment type##### Web Bean ######### deployment type ##############`AmbiguousDependencyException` ###### Web Bean ############# deployment type

Web Bean

4.4. #########Client proxies#

Web Bean ############ Web Bean instance

########## scope # Web Bean ######### scope # Web Bean ########### scope # Web Bean ################################## scope Web Bean # instance#

########## session scope # Web Bean ########## scope # Web Bean #########session context ############################## scope # Web Bean instance ### session scope # Web Bean #######

####### Web Bean ##### `@Dependent` scope### Web Bean ########## proxy ################ Web Bean### *client proxy* ###### method ### Web Bean instance ##### context ### instance#### proxy ################## Web Bean ########## context### session context## Web Bean ########

####### Java ######## Java ##### Web Bean ######proxied############ type ########Web Bean ######### `UnproxyableDependencyException`#

Java ##### Web Bean

- ##### `final` ###### `final` method # class#
- ##### constructor #### class###
- ###array#######primitive type##

`UnproxyableDependencyException` ################ constructor ###### class############### Web Bean # scope ### `@Dependent`

4.5. ############## Web Bean

############ `Manager` ####### instance#

```
@Current Manager manager;
```

`Manager` ################## Web Bean instance # method#

```
PaymentProcessor p = manager.getInstanceByType(PaymentProcessor.class);
```

########## helper class # subclass `AnnotationLiteral` ####### Java ############

```
PaymentProcessor p = manager.getInstanceByType(PaymentProcessor.class,
                                               new AnnotationLiteral<CreditCard
>(){});
```

################### `AnnotationLiteral` ##### subclass # ############ subclass#

```
abstract class CreditCardBinding
   extends AnnotationLiteral<CreditCard
>
   implements CreditCard {}
```

```
PaymentProcessor p = manager.getInstanceByType(PaymentProcessor.class,
                                    new CreditCardBinding() {
                                       public void value() { return paymentType; }
                                    } );
```

4.6. Lifecycle callback#@Resource#@EJB # @PersistenceContext

Web Bean ## EJB ######## lifecycle callback#@PostConstruct#@PreDestroy#@PrePassivate # @PostActivate#

Web Bean ### @PostConstruct # @PreDestroy callback#

####### Web Bean ##### @Resource#@EJB # @PersistenceContext ###### Java EE ###EJB # JPA # persistence context#### Web Bean ##### @PersistenceContext(type=EXTENDED)#

@PostConstruct callback ##################

4.7. InjectionPoint

######### # ## @Dependent ## scope # Web Bean # ##

- Logger # log category ########## class#
- HTTP #### header value ######################
- EL ######expression evaluation######################

@Dependent ## scope # Web Bean ###### InjectionPoint instance ############## metadata#

############################refactoring########

```
Logger log = Logger.getLogger(MyClass.class.getName());
```

producer method ######### log category ######## JDK Logger#

```
class LogFactory {
```

```
  @Produces Logger createLogger(InjectionPoint injectionPoint) {
    return Logger.getLogger(injectionPoint.getMember().getDeclaringClass().getName());
  }

}
```

########

```
@Current Logger log;
```

######################### HTTP #################

```
@BindingType
@Retention(RUNTIME)
@Target({TYPE, METHOD, FIELD, PARAMETER})
public @interface HttpParam {
  @NonBinding public String value();
}
```

##################

```
@HttpParam("username") String username;
@HttpParam("password") String password;
```

producer method

```
class HttpParams

  @Produces @HttpParam("")
  String getParamValue(ServletRequest request, InjectionPoint ip) {
    return request.getParameter(ip.getAnnotation(HttpParam.class).value());
  }

}
```

#####`HttpParam` ### `value()` #### Web Bean ############### `@NonBinding`.#

Web Bean ######## `InjectionPoint` ##### Web Bean#

```
public interface InjectionPoint {
   public Object getInstance();
   public Bean<?> getBean();
   public Member getMember():
   public <T extends Annotation
> T getAnnotation(Class<T
> annotation);
   public Set<T extends Annotation
> getAnnotations();
}
```

Scope # context

############## *scope #####scope type annotations*######Web Bean # scope ### Web Bean instance ######lifecycle##Scope ############## Web Bean # instance### Web Bean ###scope ####

- ##### scope # Web Bean ## instance ######
- ##### scope # Web Bean ### instance ######
- ############## scope # Web Bean ### instance

######## session scope # Web Bean `CurrentUser`######## `HttpSession` # context ##### Web Bean ####### `CurrentUser` # instance## `CurrentUser` ## session ############ instance ############### session ############

5.1. Scope type

Web Bean ##### *extensible context model*##### *context* ############ scope ######### scope#

```
@Retention(RUNTIME)
@Target({TYPE, METHOD})
@ScopeType
public @interface ClusterScoped {}
```

###################### scope ######################## scope # `Context` ####### `Context` ############################framework development###

####### scope ####### Web Bean ## class ### Web Bean # scope#

```
@ClusterScoped
public class SecondLevelCache { ... }
```

############ Web Bean ### scope#

5.2. ## scope

Web Bean ######## scope#

- `@RequestScoped`
- `@SessionScoped`
- `@ApplicationScoped`

- `@ConversationScoped`

Web Bean

- ## servlet ###### active request#session ##### scope###
- ## JSF ############ conversation scope#

####### scope #############

- ### EJB ### method ##
- # EJB ####
- ########### bean ####
- ###########

################ context # scope ### Web Bean ###Web Bean ###### runtime ##### `ContextNotActiveException`#

scope ########### Java EE ##################################### scope

5.3. conversation scope

Web Bean # conversation scope #### session scope ##################################conversation scope # session scope ########

- conversation scope ##############
- ## JSF ########################

Conversation############################conversation # context ## conversation#

conversation context #### JSF ################## conversation ################ conversation ###################### *long-running conversation*#

5.3.1. Conversation demarcation######

Web Bean ######## Web Bean######## JSF ######## conversation ######## Web Bean ###########

```
@Current Conversation conversation;
```

############ conversation ## long-running conversation ################ `begin()` method######## long-running conversation context ################ `end()`#

######### conversation-scoped # Web Bean ######## conversation#

```
@ConversationScoped @Stateful
public class OrderBuilder {

   private Order order;
   private @Current Conversation conversation;
   private @PersistenceContext(type=EXTENDED) EntityManager em;

   @Produces public Order getOrder() {
      return order;
   }

   public Order createOrder() {
      order = new Order();
      conversation.begin();
      return order;
   }

   public void addLineItem(Product product, int quantity) {
      order.add( new LineItem(product, quantity) );
   }

   public void saveOrder(Order order) {
      em.persist(order);
      conversation.end();
   }

   @Remove
   public void destroy() {}

}
```

Web Bean ###### `Conversation` API ################## Web Bean

5.3.2. Conversation propagation####

conversation context ##### JSF face ####JSF form submission############ non-face ####################

######## conversation #######unique identifier############ conversation ## non-face ########Web Bean ######### `cid` ##############conversation ######### `Conversation` ########### Web Bean ### `conversation`#

########## conversation#

```
<a href="/addProduct.jsp?cid=#{conversation.id}"
>Add Product</a
>
```

Web Bean ####### conversation ################## conversation ###### long-running######### POST-then-redirect ######################flash###### construct#######Web Bean ################## URL#

5.3.3. Conversation timeout####

Web Bean ############ context ## conversation ############## Web Bean ############### timeout #### # #### Web Bean #########Timeout ## conversation ##################

`Conversation` ############ timeout # method##### Web Bean ###############

```
conversation.setTimeout(timeoutInMillis);
```

5.4. dependent pseudo-scope

######## scope#Web Bean ######## *dependent pseudo-scope* ############ scope ### Web Bean ### scope#

Web Bean # scope ### `@Dependent`#

```
public class Calculator { ... }
```

Web Bean ############## Web Bean ####### Web Bean ####instantiate##### Web Bean ##### instance ######## Web Bean # instance ########## Web Bean ############### Web Bean instance # *dependent object*#######

Web Bean # instance ######## instance

Web Bean #### Java class ## EJB bean ### instance ########## class ## EJB bean ########### scope ### Web Bean#

5.4.1. `@New`

`@New` ################## Web Bean

```
@New Calculator calculator;
```

######## scope @Dependent##### @New#API ## Calculator### class Calculator ###### @Standard # Web Bean ##########

Calculator ######## scope type############

```
@ConversationScoped
public class Calculator { ... }
```

################## Calculator ### instance#

```
public class PaymentCalc {

    @Current Calculator calculator;
    @New Calculator newCalculator;

}
```

calculator ########## Calculator # conversation-scoped instance#newCalculator ####### Calculator # instance ################### PaymentCalc#

producer method

Producer method

Producer method ##### Web Bean ####################instantiating########################## Web Bean ##### Web Bean ################ *# 12, ## XML ### Web Bean* ##########

#####

> A Web Beans producer method acts as a source of objects to be injected, where:
>
> - the objects to be injected are not required to be instances of Web Beans,
> - the concrete type of the objects to be injected may vary at runtime or
> - the objects require some custom initialization that is not performed by the Web Bean constructor

For example, producer methods let us:

- expose a JPA entity as a Web Bean,
- expose any JDK class as a Web Bean,
- define multiple Web Beans, with different scopes or initialization, for the same implementation class, or
- vary the implementation of an API type at runtime.

In particular, producer methods let us use runtime polymorphism with Web Beans. As we've seen, deployment types are a powerful solution to the problem of deployment-time polymorphism. But once the system is deployed, the Web Bean implementation is fixed. A producer method has no such limitation:

```
@SessionScoped
public class Preferences {

   private PaymentStrategyType paymentStrategy;

   ...

   @Produces @Preferred
   public PaymentStrategy getPaymentStrategy() {
      switch (paymentStrategy) {
         case CREDIT_CARD: return new CreditCardPaymentStrategy();
         case CHEQUE: return new ChequePaymentStrategy();
```

```
        case PAYPAL: return new PayPalPaymentStrategy();
        default: return null;
      }
   }

}
```

Consider an injection point:

```
@Preferred PaymentStrategy paymentStrat;
```

This injection point has the same type and binding annotations as the producer method, so it resolves to the producer method using the usual Web Beans injection rules. The producer method will be called by the Web Bean manager to obtain an instance to service this injection point.

6.1. producer method # scope

producer method ### scope # `@Dependent`##### Web Bean ##################### producer method #####`@Dependent` ############## session ####### `PaymentStrategy` ### instance ###

############ `@SessionScoped` ####### method#

```
@Produces @Preferred @SessionScoped
public PaymentStrategy getPaymentStrategy() {
   ...
}
```

producer method ######## `PaymentStrategy` ###### session context#Producer method ####### session

6.2. ## producer method

##############`CreditCardPaymentStrategy` ######## Java # `new` operator ####instantiate############################dependency injection#########

###################### producer method ##### Web Bean # instance#

```
@Produces @Preferred @SessionScoped
public PaymentStrategy getPaymentStrategy(CreditCardPaymentStrategy ccps,
                                          ChequePaymentStrategy cps,
                                          PayPalPaymentStrategy ppps) {
```

```
   switch (paymentStrategy) {
      case CREDIT_CARD: return ccps;
      case CHEQUE: return cps;
      case PAYPAL: return ppps;
      default: return null;
   }
}
```

CreditCardPaymentStrategy #### scope # Web Bean ############## producer method ###### request scope # instance####session scope ############## bug#request scope #### session #### Web Bean ###################### session scope ########## Web Bean ########### producer method ## Web Bean # instance

###################### CreditCardPaymentStrategy ### scope######### Web Bean ################ producer method # scope ### @Dependent # @RequestScoped#

################## @New #####

6.3. ## @New # producer method

producer method#

```
@Produces @Preferred @SessionScoped
public PaymentStrategy getPaymentStrategy(@New CreditCardPaymentStrategy ccps,
                                          @New ChequePaymentStrategy cps,
                                          @New PayPalPaymentStrategy ppps) {
   switch (paymentStrategy) {
      case CREDIT_CARD: return ccps;
      case CHEQUE: return cps;
      case PAYPAL: return ppps;
      default: return null;
   }
}
```

CreditCardPaymentStrategy # *dependent* instance ######### producer method## producer method ########## session context## Preferences ### session #########dependent

II. é##ç#¼é¬#æ#£è#¦å##ï¼#loosely-coupledï¼#ç##ç¨#å¼#ç¢¼

Web Bean ######## *loose coupling*################## loose coupling ####

- *deployment types*##################deployment time polymorphism##
- *producer methods* ### runtime Polymorphism###########
- *contextual lifecycle management* # decouple Web Bean ######

#################### loose coupling################# API ##

Loose coupling ###################################### framework ######################### type safety #####Web Bean ##### typesafe ########## loose coupling ####

Web Bean ################## loose coupling#

- *interceptors*##################
- *decorators*####################
- *event notifications*################event producer#######event consumer##

###########interceptor##

####Interceptor#

Web Bean ##### EJB 3.0 ######################

- ### session bean### Web Bean #######
- ########## Web Bean#Web Bean ###############

EJB ###############

- business method ######
- lifecycle callback ####

business method ######### Web Bean #### Web Bean # method ###

```
public class TransactionInterceptor {
    @AroundInvoke public Object manageTransaction(InvocationContext ctx) { ... }
}
```

lifecycle callback ######### container # lifecycle callback ####

```
public class DependencyInjectionInterceptor {
    @PostConstruct public void injectDependencies(InvocationContext ctx) { ... }
}
```

class ### lifecycle callback ## business method#

7.1.

############ Web Bean ###### Web Bean############ ########*interceptor binding annotation*############ Web Bean ####

```
@InterceptorBindingType
@Target({METHOD, TYPE})
@Retention(RUNTIME)
public @interface Transactional {}
```

############ `ShoppingCart` ###########

```
@Transactional
```

```
public class ShoppingCart { ... }
```

###################### method ######

```
public class ShoppingCart {
   @Transactional public void checkout() { ... }
}
```

7.2.

`########################################` EJB `#########` `@Interceptor` # `@Transactional#`

```
@Transactional @Interceptor
public class TransactionInterceptor {
   @AroundInvoke public Object manageTransaction(InvocationContext ctx) { ... }
}
```

Web Bean ######## Web Bean###############dependency injection### contextual lifecycle

```
@ApplicationScoped @Transactional @Interceptor
public class TransactionInterceptor {

   @Resource Transaction transaction;

   @AroundInvoke public Object manageTransaction(InvocationContext ctx) { ... }

}
```

######################

7.3.

######## `web-beans.xml` ##########

```
<Interceptors>
   <tx:TransactionInterceptor/>
</Interceptors
```

```
>
```

########

XML ##########

- ########################total ordering###############
- ##################### class#

######################### `TransactionInterceptor` ######

```
<Interceptors>
   <sx:SecurityInterceptor/>
   <tx:TransactionInterceptor/>
</Interceptors
>
```

######################

7.4.

################### `@Transactional` ####

```
@InterceptorBindingType
@Target({METHOD, TYPE})
@Retention(RUNTIME)
public @interface Transactional {
   boolean requiresNew() default false;
}
```

Web Bean #### `requiresNew` #############`TransactionInterceptor` # `RequiresNewTransactionInterceptor`#######

```
@Transactional(requiresNew=true) @Interceptor
public class RequiresNewTransactionInterceptor {
   @AroundInvoke public Object manageTransaction(InvocationContext ctx) { ... }
}
```

######### `RequiresNewTransactionInterceptor`#

```
@Transactional(requiresNew=true)
public class ShoppingCart { ... }
```

############################## `requiresNew` ################# `@NonBinding` ###

```
@InterceptorBindingType
@Target({METHOD, TYPE})
@Retention(RUNTIME)
public @interface Secure {
   @NonBinding String[] rolesAllowed() default {};
}
```

7.5.

############################## Web Bean############## `TransactionInterceptor` # `SecurityInterceptor` ###### Web Bean#

```
@Secure(rolesAllowed="admin") @Transactional
public class ShoppingCart { ... }
```

###################################

```
@Transactional @Secure @Interceptor
public class TransactionalSecureInterceptor { ... }
```

########################## `checkout()` ## method#

```
public class ShoppingCart {
   @Transactional @Secure public void checkout() { ... }
}
```

```
@Secure
public class ShoppingCart {
   @Transactional public void checkout() { ... }
}
```

```
@TransactionI
public class ShoppingCart {
   @Secure public void checkout() { ... }
}
```

```
@Transactional @Secure
public class ShoppingCart {
   public void checkout() { ... }
}
```

7.6. ######## inheritance

Java ######################## inheritance############ reuse###############

```
public @interface Action extends Transactional, Secure { ... }
```

#####Web Bean ###### Java ## ############## Web Bean ####### meta-annotation #######

```
@Transactional @Secure
@InterceptorBindingType
@Target(TYPE)
@Retention(RUNTIME)
public @interface Action { ... }
```

`@Action` # Web Bean ###### `TransactionInterceptor` # `SecurityInterceptor`##### `TransactionalSecureInterceptor`#########

7.7. ## `@Interceptors`

######## Web Bean ### EJB ###### `@Interceptors` ######

```
@Interceptors({TransactionInterceptor.class, SecurityInterceptor.class})
public class ShoppingCart {
   public void checkout() { ... }
}
```

##############

- ######## hardcode # business code ###
- ######################
- ############## # ######## class #########

######### Web Bean #########

####Decorators#

######Interceptors## Java ######invocation## ###############################

####*decorators*################ java ##

###############

```
public interface Account {
   public BigDecimal getBalance();
   public User getOwner();
   public void withdraw(BigDecimal amount);
   public void deposit(BigDecimal amount);
}
```

############## Web Bean ## `Account` ###

####decorator###### Web Bean################### `@Decorator`#

```
@Decorator
public abstract class LargeTransactionDecorator
      implements Account {

   @Decorates Account account;

   @PersistenceContext EntityManager em;

   public void withdraw(BigDecimal amount) {
      account.withdraw(amount);
      if ( amount.compareTo(LARGE_AMOUNT)
>0 ) {
         em.persist( new LoggedWithdrawl(amount) );
      }
   }

   public void deposit(BigDecimal amount);
      account.deposit(amount);
      if ( amount.compareTo(LARGE_AMOUNT)
>0 ) {
```

```
        em.persist( new LoggedDeposit(amount) );
      }
    }

}
```

Web Bean ##################abstract class################## method ################## method#

8.1. Delegate##########

######### *delegate* ###delegate ####################### Web Bean#delegate ########################

delegate ############### `Account` # Web Bean#

```
@Decorates Account account;
```

Delegate ######## binding annotation######################### binding # Web Bean#

```
@Decorates @Foreign Account account;
```

################## Web Bean#

- #### API ### delegate #######
- ## delegate #############

######## delegate ############### `InvocationContext.proceed()` ########

8.2.

`web-beans.xml`

```
<Decorators>
   <myapp:LargeTransactionDecorator/>
</Decorators
>
```

####declaration########## `<Interceptors>` ###############

- ##########################total ordering###############

- ######################

method ########## method

###Events#

Web Bean ######## Web Bean ##### decouple ###############*producers*###############
Web Bean #############*observers*######## schema ############/####################

- ############## decouple############## decouple ##
- ##############selectors#############################
- ##

9.1. ######Event observers#

observer method ## Web Bean # method ###### `@Observes` ####

```
public void onAnyDocumentEvent(@Observes Document document) { ... }
```

################*event parameter*######### type #############*event type*##Observer method #####selector######## Web Bean #####binding types## instance###*event binding type*##

```
@BindingType
@Target({PARAMETER, FIELD})
@Retention(RUNTIME)
public @interface Updated { ... }
```

############### observer method ######

```
public void afterDocumentUpdate(@Observes @Updated Document document) { ... }
```

observer method ############ # ############ type

Observer method ###################### Web Bean method #######parameter injection semantic#####

```
public void afterDocumentUpdate(@Observes @Updated Document document, User user) { ... }
```

9.2. ######Event producers#

####################*event notifier*# ###

```
@Observable Event<Document
> documentEvent
```

`@Observable` ######### `@Dependent` ## scope # `@Standard` ######### Web Bean ################ Web Bean#

######## `Event` ### `fire()` method###### *event object* ######

```
documentEvent.fire(document);
```

Event object ####### type variable ## wildcard type ##### Java class # instance################## observer method#

- ##### event object ########
- ######### observer method#

Web Bean ######## observer method### event object ################## observer method ##### exception#Web Bean ######### observer method##### exception ## `fire()` method #####

#######selector################## instance ### `fire()` ## method#

```
documentEvent.fire( document, new AnnotationLiteral<Updated
>(){} );
```

`AnnotationLiteral` ## helper class ################ Java ##########

################## method#

- ##### event object ########
- ################ `fire()` ######

#########################

```
@Observable @Updated Event<Document
> documentUpdatedEvent
```

######### `Event` # instance ## method#

- ##### event object ########
- ################## `fire()` ##########################

9.3.

################################ `Observer` ######## `observe()` method ############### instance#

```
documentEvent.observe( new Observer<Document
>() { public void notify(Document doc) { ... } } );
```

################################## instance ### `observe()` method#

```
documentEvent.observe( new Observer<Document
>() { public void notify(Document doc) { ... } },
                                    new AnnotationLiteral<Updated
>(){} );
```

9.4. member

################

```
@BindingType
@Target({PARAMETER, FIELD})
@Retention(RUNTIME)
public @interface Role {
   RoleType value();
}
```

member value#################################

```
public void adminLoggedIn(@Observes @Role(ADMIN) LoggedIn event) { ... }
```

#####################################

```
@Observable @Role(ADMIN) Event<LoggedIn
```

```
> LoggedInEvent;}}
```

`AnnotationLiteral` # abstract subclass

```
abstract class RoleBinding
   extends AnnotationLiteral<Role
>
   implements Role {}
```

######### class ### instance ### `fire()`#

```
documentEvent.fire( document, new RoleBinding() { public void value() { return user.getRole();
 } } );
```

9.5. #######Multiple event binding#

###############

```
@Observable @Blog Event<Document
> blogEvent;
...
if (document.isBlog()) blogEvent.fire(document, new AnnotationLiteral<Updated
>(){});
```

############## observer method ######

```
public void afterBlogUpdate(@Observes @Updated @Blog Document document) { ... }
```

```
public void afterDocumentUpdate(@Observes @Updated Document document) { ... }
```

```
public void onAnyBlogEvent(@Observes @Blog Document document) { ... }
```

```
public void onAnyDocumentEvent(@Observes Document document) { ... }}}
```

9.6. ######Transactional observers#

observer method ############### context ############## `Category` tree

```
public void refreshCategoryTree(@AfterTransactionSuccess @Observes CategoryUpdateEvent
 event) { ... }
```

############

- `@AfterTransactionSuccess` ###################################
- `@AfterTransactionFailure` ###################################
- `@AfterTransactionCompletion` ##################
- `@BeforeTransactionCompletion` #################

########## Web Bean # stateful ############# state ########## atomic transaction#############

################# scope ## JPA #####

```
@ApplicationScoped @Singleton
public class Catalog {

   @PersistenceContext EntityManager em;

   List<Product
> products;

   @Produces @Catalog
   List<Product
> getCatalog() {
      if (products==null) {
         products = em.createQuery("select p from Product p where p.deleted = false")
            .getResultList();
      }
      return products;
   }

}
```

`Product` ######################### `Product` # catalog############################

`Product` # Web Bean

```
@Stateless
public class ProductManager {

   @PersistenceContext EntityManager em;
   @Observable Event<Product
> productEvent;

   public void delete(Product product) {
      em.delete(product);
      productEvent.fire(product, new AnnotationLiteral<Deleted
>(){});
   }

   public void persist(Product product) {
      em.persist(product);
      productEvent.fire(product, new AnnotationLiteral<Created
>(){});
   }

   ...

}
```

####`Catalog` ################

```
@ApplicationScoped @Singleton
public class Catalog {

   ...

   void addProduct(@AfterTransactionSuccess @Observes @Created Product product) {
      products.add(product);
   }

   void addProduct(@AfterTransactionSuccess @Observes @Deleted Product product) {
      products.remove(product);
   }

}
```

III. æ##å¤§ç¨#åº¦å#°ä½¿ç#¨å¼·é¡#å##ï¼#strong typingï¼#

Web Bean ######## *strong typing*######### Web Bean ## typesafe Java ####

Web Bean ######################### framework ################# # ####configuration by convention# #

############# IDE ######################autocompletion#####validation#####refactoring#################################### level##

Web Bean ###############

- `@Asynchronous`#
- `@Mock`#
- `@Secure` #
- `@Updated`#

#######

- `asyncPaymentProcessor`#
- `mockPaymentProcessor`#
- `SecurityInterceptor` ##
- `DocumentUpdatedEvent` #######

##############################common qualities###

Web Bean *stereotype* ##############################stereotype model###################*role*################ scope#######interceptor binding#######deployment type########################

Web Bean XML # metadata #######XML ######## Web Bean ### XML schema ###### XML ## Java ############ XML ################ Java

################## Web Bean ##

########Stereotypes#

Web Bean

> ######################## Web Bean ############## framework ################################## Web Bean ###### metadata#
>
> ###################
>
> - ########
> - ### scope ###
> - ## Web Bean scope ####
> - Web Bean ################
> - ##########
>
> ######################## Web Bean ###### Web Bean ###
>
> Web Bean #################

####### Java ################### MVC ###### class#

```
@Retention(RUNTIME)
@Target(TYPE)
@Stereotype
public @interface Action {}
```

######### Web Bean ##########

```
@Action
public class LoginAction { ... }
```

10.1. ########## scope

################### Web Bean ##### scope #/################ `@WebTier` ######################## Web Bean ################## action class ########

```
@Retention(RUNTIME)
@Target(TYPE)
@RequestScoped
```

```
@WebTier
@Stereotype
public @interface Action {}
```

############################

```
@Dependent @Mock @Action
public class MockLoginAction { ... }
```

################## scope ############

10.2. ############ scope # type

################ scope#Web Bean ################### Web Bean ######## scope####

```
@Retention(RUNTIME)
@Target(TYPE)
@RequestScoped
@WebTier
@Stereotype(supportedScopes=RequestScoped.class)
public @interface Action {}
```

action class ####### Web Bean #### scope ###Web Bean ############## exception#

#################### Web Bean ########### class#

```
@Retention(RUNTIME)
@Target(TYPE)
@RequestScoped
@WebTier
@Stereotype(requiredTypes=AbstractAction.class)
public @interface Action {}
```

action class #### `AbstractAction` ## class ###Web Bean ############## exception#

10.3.

############################## Web Bean ###

```
@Retention(RUNTIME)
```

```
@Target(TYPE)
@RequestScoped
@Transactional(requiresNew=true)
@Secure
@WebTier
@Stereotype
public @interface Action {}
```

######## business code ########

10.4.

######################## Web Bean ##### Web Bean ###### Web Bean ############## JSP ################## use case########## `@Named` #####

```
@Retention(RUNTIME)
@Target(TYPE)
@RequestScoped
@Transactional(requiresNew=true)
@Secure
@Named
@WebTier
@Stereotype
public @interface Action {}
```

###`LoginAction` ###### `loginAction`#

10.5.

Web Bean ##################`@Interceptor` # `@Decorator`#

Web Bean ##################

```
@Named
@RequestScoped
@Stereotype
@Target({TYPE, METHOD})
@Retention(RUNTIME)
public @interface Model {}
```

############ JSF##### JSF ### bean######## `@Model` Web Bean ####### JSF #########

####Specialization#

####### Web Bean ####################*override*#### API ############## Web Bean ####### `PaymentProcessor` ## API ######

```
@CreditCard @Stateless
public class CreditCardPaymentProcessor
      implements PaymentProcessor {
   ...
}
```

###########staging environment########### Web Bean #### `PaymentProcessor` ####

```
@CreditCard @Stateless @Staging
public class StagingCreditCardPaymentProcessor
      implements PaymentProcessor {
   ...
}
```

`StagingCreditCardPaymentProcessor` ############## deployment ####### `AsyncPaymentProcessor`### deployment ##`@Staging` ## deployment type ##### `@Production` #### deployment type

```
@CreditCard PaymentProcessor ccpp
```

`StagingCreditCardPaymentProcessor` ### instance#

############

- ###### Web Bean ########### Web Bean ### API ###
- ###### Web Bean ########### Web Bean ########
- ###### Web Bean ############ Web Bean ########
- ####### Web Bean ######## producer method#disposal method ## observer method#

################ Web Bean ###### runtime ######################

Web Bean ############ *specialization*##################Specialization ##

11.1. ## specialization

Specialization ########## Web Bean ########## specialization######### Web Bean ###

- ####### Web Bean ### subclass###
- ###### Web Bean ##### Web Bean######### Web Bean######### Web Bean ###### Web Bean############ Web Bean###
- #### `@Specializes`#

```
@Stateless @Staging @Specializes
public class StagingCreditCardPaymentProcessor
      extends CreditCardPaymentProcessor {
   ...
}
```

Web Bean # *specializes* ## superclass#

11.2. Specializarion

specialization

- superclass ############## `@Specializes` # Web Bean #####
- superclass # Web Bean ########## `@Specializes` # Web Bean #####
- superclass #### producer method#disposal method ## observer method ####### `@Specializes` # Web Bean instance ###

########`CreditCardPaymentProcessor` # `@CreditCard` ###### `StagingCreditCardPaymentProcessor` ###

###Web Bean ################

- superclass ### API ######## `@Specializes` # Web Bean # API ###superclass enterprise bean ########## subclass #######
- ### `@Specializes` # Web Bean # deployment type ### superclass # deployment type ##########
- ######## Web Bean # specialize # superclass#

###############Web Bean ############## exception#

############## `@Specializes` # Web Bean ############superclass ######### deployment #####

XML ### Web Bean

######################## Web Bean ######################## Web Bean#

- #### class ##############
- ###### class ###### Web Bean#

########Web Bean ##########

- #### producer method###
- ## XML ### Web Bean#

####### XML #### Java class ### metadata####Web Bean ########### Java class ######## method ################ class ######## XML ###### string value#Web Bean ##### class ######### XML ######

################### XML ######### XML schema############## Java ########## XML schema##

12.1. ## Web Bean class

Web Bean ###### Java ######## XML namespace### namespace #### `urn:java:` ### Java ###########`com.mydomain.myapp` ##### XML namespace ## `urn:java:com.mydomain.myapp`#

Java type ############## namespace #### XML ########### Java type ###### type #### method #### namespace ############ type

######## XML ## `<util:Date/>` #### `java.util.Date` ## class#

```
<WebBeans xmlns="urn:java:javax.webbeans"
      xmlns:util="urn:java:java.util">

   <util:Date/>

</WebBeans
>
```

##################`Date` ##### Web Bean#`Date` # instance ######### Web Bean ###

```
@Current Date date
```

12.2. ## Web Bean metadata

######### Web Bean ##########################

```
<myapp:ShoppingCart>
   <SessionScoped/>
   <myfwk:Transactional requiresNew="true"/>
   <myfwk:Secure/>
</myapp:ShoppingCart
>
```

########################

```
<util:Date>
   <Named
>currentTime</Named>
</util:Date>

<util:Date>
   <SessionScoped/>
   <myapp:Login/>
   <Named
>loginTime</Named>
</util:Date>

<util:Date>
   <ApplicationScoped/>
   <myapp:SystemStart/>
   <Named
>systemStartTime</Named>
</util:Date
>
```

`@Login` # `@SystemStart` ########binding annotations type##

```
@Current Date currentTime;
@Login Date loginTime;
@SystemStart Date systemStartTime;
```

#######Web Bean ##########

```
<myapp:AsynchronousChequePaymentProcessor>
   <myapp:PayByCheque/>
   <myapp:Asynchronous/>
</myapp:AsynchronousChequePaymentProcessor
>
```

############ Web Bean############### Web Bean #######

```
<myfwk:TransactionInterceptor>
   <Interceptor/>
   <myfwk:Transactional/>
</myfwk:TransactionInterceptor
>
```

12.3. ## Web Bean

####

12.4. ######inline#Web Bean

Web Bean ############ Web Bean####

```
<myapp:System>
   <ApplicationScoped/>
   <myapp:admin>
      <myapp:Name>
         <myapp:firstname
>Gavin</myapp:firstname>
         <myapp:lastname
>King</myapp:lastname>
         <myapp:email
>gavin@hibernate.org</myapp:email>
      </myapp:Name>
   </myapp:admin>
</myapp:System
>
```

`<Name>` ################### `@Dependent` ## scope ## `Name` class ### Web Bean#

############## Web Bean XML ######## Java ##### graph########## databinding ###############

12.5. ## schema

######## XML ###### Java ## schema#### Web Bean###### schema #########

```
<WebBeans xmlns="urn:java:javax.webbeans"
        xmlns:myapp="urn:java:com.mydomain.myapp"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="urn:java:javax.webbeans http://java.sun.com/jee/web-beans-1.0.xsd
                          urn:java:com.mydomain.myapp http://mydomain.com/xsd/myapp-1.2.xsd">

   <myapp:System>
      ...
   </myapp:System>

</WebBeans
>
```

XML schema #########Web Bean RI ################ Java ######## XML schema

IV. Web Beans è##
Java EE æ##ç#¨ç³»çµ±

Web Bean ####### *integration*######Web Bean ##Web Bean ############ Java EE ################## portable extension ################ framework ############

####### Web Bean ###### EJB # JSF### EJB ######## JSF ############Web Bean ########################## Framework ####################Java EE ############### Java ################ Web Bean #### Java EE ######################

############## Web Bean ########### Java EE ################## Web Bean ## portable extension # SPI################## SPI#################################### extension ###############

Java EE

Web Bean ###### Java EE ####Web Bean ### Java EE #### JPA persistence context########## JSF ## JSP #### Unified EL ##################### Servlets ####### Bean ### Web Beans##

13.1. # Java EE #### Web Bean

######## Web Bean #### `@Resource#@EJB` ## `@PersistenceContext` ##### Java #####dependency injection##############################

```
@Transactional @Interceptor
public class TransactionInterceptor {

   @Resource Transaction transaction;

   @AroundInvoke public Object manageTransaction(InvocationContext ctx) { ... }

}
```

```
@SessionScoped
public class Login {

   @Current Credentials credentials;
   @PersistenceContext EntityManager userDatabase;

   ...

}
```

######## Web Bean #### Java EE `@PostConstruct` # `@PreDestroy` # callback#`@PostConstruct` ## method ################

################ Web Bean #### `@PersistenceContext(type=EXTENDED)#`

13.2. #### Servlet ### Web Bean

Java EE 6 ### Servlet ### Web Bean ########## Web Bean #### initializer method ##### Web Bean

```
public class Login extends HttpServlet {
```

```
    @Current Credentials credentials;
    @Current Login login;

    @Override
    public void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        credentials.setUsername( request.getAttribute("username") ):
        credentials.setPassword( request.getAttribute("password") ):
        login.login();
        if ( login.isLoggedIn() ) {
            response.sendRedirect("/home.jsp");
        }
        else {
            response.sendRedirect("/loginError.jsp");
        }
    }

}
```

Web Bean #### proxy ######### HTTP session ### Servlet # `Credentials` # `Login` ###### routing method ###

13.3. ####### Bean ### Web Bean

Web Bean ####### EJB######## Web Bean ################### JNDI ######## `@EJB` ######################## Bean ### Web Bean ####### Bean ###### Web Bean ##########

######## Bean######## Web Bean ######

```
@Transactional @MessageDriven
public class ProcessOrder implements MessageListener {

    @Current Inventory inventory;
    @PersistenceContext EntityManager em;

    public void onMessage(Message message) {
        ...
    }

}
```

Web Bean ################################## Bean ######## session #### context### `@RequestScoped` # `@ApplicationScoped` Web Bean

Web Bean

13.4. JMS

#################### JMS ################ queue#### `Queue#QueueConnectionFactory#QueueConnection#QueueSession` ## `QueueSender`#### topic ### `Topic#TopicConnectionFactory#TopicConnection#TopicSession` ## `TopicPublisher`##threading model##

Web Bean ############### `web-beans.xml` ### queue ## topic################connection factory##

```
<Queue>
   <destination
>java:comp/env/jms/OrderQueue</destination>
   <connectionFactory
>java:comp/env/jms/QueueConnectionFactory</connectionFactory>
   <myapp:OrderProcessor/>
</Queue
>
```

```
<Topic>
   <destination
>java:comp/env/jms/StockPrices</destination>
   <connectionFactory
>java:comp/env/jms/TopicConnectionFactory</connectionFactory>
   <myapp:StockPrices/>
</Topic
>
```

####### queue ## `Queue#QueueConnection#QueueSession` # `QueueSender`#### topic ## `Topic#TopicConnection#TopicSession` ## `TopicPublisher` ###

```
@OrderProcessor QueueSender orderSender;
@OrderProcessor QueueSession orderSession;

public void sendMessage() {
   MapMessage msg = orderSession.createMapMessage();
   ...
```

```
    orderSender.send(msg);
}
```

```
@StockPrices TopicPublisher pricePublisher;
@StockPrices TopicSession priceSession;

public void sendMessage(String price) {
    pricePublisher.send( priceSession.createTextMessage(price) );
}
```

JMS ########## Web Bean

13.5.

Web Bean ########## archive#### Web Bean ### JARs#EJB-JARs ## WARs # ###### classpath ################## Web Bean # archive # `META-INF` ## `WEB-INF` ############## `web-beans.xml` ################## `web-beans.xml` ### archive ## Web Bean ############

Java SE ######Web Bean ############## EJB ########## EJB Lite container ################## `web-beans.xml`

Web Bean

Web Bean ############################Web Bean ###### Web Bean # portable extension ######## SPI############## Web Bean #########

- ############
- ######### Spring#Seam#GWT # Wicket########
- ## Web Bean #######Programming Model######

Web Bean ###### `Manager`

14.1. `Manager`

`Manager` ############### Web Bean#####interceptor######decorator######observer#### context#

```
public interface Manager
{

  public <T
> Set<Bean<T
>
> resolveByType(Class<T
> type, Annotation... bindings);

  public <T
> Set<Bean<T
>
> resolveByType(TypeLiteral<T
> apiType,
      Annotation... bindings);

  public <T
> T getInstanceByType(Class<T
> type, Annotation... bindings);

  public <T
> T getInstanceByType(TypeLiteral<T
> type,
      Annotation... bindings);

  public Set<Bean<?>
> resolveByName(String name);
```

```
  public Object getInstanceByName(String name);

  public <T
> T getInstance(Bean<T
> bean);

  public void fireEvent(Object event, Annotation... bindings);

  public Context getContext(Class<? extends Annotation
> scopeType);

  public Manager addContext(Context context);

  public Manager addBean(Bean<?> bean);

  public Manager addInterceptor(Interceptor interceptor);

  public Manager addDecorator(Decorator decorator);

  public <T
> Manager addObserver(Observer<T
> observer, Class<T
> eventType,
      Annotation... bindings);

  public <T
> Manager addObserver(Observer<T
> observer, TypeLiteral<T
> eventType,
      Annotation... bindings);

  public <T
> Manager removeObserver(Observer<T
> observer, Class<T
> eventType,
      Annotation... bindings);

  public <T
> Manager removeObserver(Observer<T
> observer,
      TypeLiteral<T
> eventType, Annotation... bindings);
```

```
  public <T
> Set<Observer<T
>
> resolveObservers(T event, Annotation... bindings);

  public List<Interceptor
> resolveInterceptors(InterceptionType type,
        Annotation... interceptorBindings);

  public List<Decorator
> resolveDecorators(Set<Class<?>
> types,
        Annotation... bindings);

}
```

########## Manager ### instance#

```
@Current Manager ###
```

14.2. Bean class

Bean ## abstract class # instance ## Web Bean########## Web Bean ##### Manager ##### Bean instance#

```
public abstract class Bean<T> {

   private final Manager manager;

   protected Bean(Manager manager) {
      this.manager=manager;
   }

   protected Manager getManager() {
      return manager;
   }

   public abstract Set<Class> getTypes();
   public abstract Set<Annotation> getBindingTypes();
   public abstract Class<? extends Annotation> getScopeType();
   public abstract Class<? extends Annotation> getDeploymentType();
   public abstract String getName();
```

```
    public abstract boolean isSerializable();
    public abstract boolean isNullable();

    public abstract T create();
    public abstract void destroy(T instance);

}
```

`Manager.addBean()` ### `Bean` class ### instance ######### Web Bean ########### Web Bean#producer method ## JMS #### Web Bean ############### `Bean` class ################## Web Bean

Web Bean ##### `Bean` ### subclass#`Interceptor` # `Decorator`#

14.3. `Context`

`Context` ######### Web Bean ####################

```
public interface Context {

    public Class<? extends Annotation> getScopeType();

    public <T> T get(Bean<T> bean, boolean create);

    boolean isActive();

}
```

############ `Context` ############# Web Bean######### Wicket #################

######

Web Beans

###Web Beans ####### Web Beans ################ 100 ## http://jcp.org/en/jsr/detail?id=299 #########

Web Beans Reference ###### http://seamframework.org/WebBeans#RI ###### Web Beans spec lead # blog ## http://in.relation.to#################### blog ###

V. Web Beans Reference

Web Beans is the reference implementation of JSR-299, and is used by JBoss AS and Glassfish to provide JSR-299 services for Java Enterprise Edition applications. Web Beans also goes beyond the environments and APIs defined by the JSR-299 specification and provides support for a number of other environments (such as a servlet container such as Tomcat, or Java SE) and additional APIs and modules (such as logging, XSD generation for the JSR-299 XML deployment descriptors).

If you want to get started quickly using Web Beans with JBoss AS or Tomcat and experiment with one of the examples, take a look at *# 3, Web Beans, the Reference Implementation of JSR-299*. Otherwise read on for a exhaustive discussion of using Web Beans in all the environments and application servers it supports, as well the Web Beans extensions.

Application Servers and environments supported by Web Beans

16.1. Using Web Beans with JBoss AS

No special configuration of your application, beyond adding either `META-INF/beans.xml` or `WEB-INF/beans.xml` is needed.

If you are using JBoss AS 5.0.1.GA then you'll need to install Web Beans as an extra. First we need to tell Web Beans where JBoss is located. Edit `jboss-as/build.properties` and set the `jboss.home` property. For example:

```
jboss.home=/Applications/jboss-5.0.1.GA
```

Now we can install Web Beans:

```
$ cd webbeans-$VERSION/jboss-as
$ ant update
```

#

A new deployer, `webbeans.deployer` is added to JBoss AS. This adds supports for JSR-299 deployments to JBoss AS, and allows Web Beans to query the EJB3 container and discover which EJBs are installed in your application.

Web Beans is built into all releases of JBoss AS from 5.1 onwards.

16.2. Glassfish

TODO

16.3. Tomcat (or any plain Servlet container)

Web Beans can be used in Tomcat 6.0.

#

Web Beans doesn't support deploying session beans, injection using `@EJB`, or `@PersistenceContext` or using transactional events on Tomcat.

Web Beans should be used as a web application library in Tomcat. You should place `webbeans-tomcat.jar` in `WEB-INF/lib`. `webbeans-tomcat.jar` is an "uber-jar" provided for your convenience. Instead, you could use its component jars:

- `jsr299-api.jar`
- `webbeans-api.jar`
- `webbeans-spi.jar`
- `webbeans-core.jar`
- `webbeans-logging.jar`
- `webbeans-tomcat-int.jar`
- `javassist.jar`
- `dom4j.jar`

You also need to explicitly specify the Tomcat servlet listener (used to boot Web Beans, and control its interaction with requests) in `web.xml`:

```
<listener>
  <listener-class
>org.jboss.webbeans.environment.servlet.Listener</listener-class>
</listener
>
```

Tomcat has a read-only JNDI, so Web Beans can't automatically bind the Manager. To bind the Manager into JNDI, you should add the following to your `META-INF/context.xml`:

```
<Resource name="app/Manager"
        auth="Container"
        type="javax.inject.manager.Manager"
        factory="org.jboss.webbeans.resources.ManagerObjectFactory"/>
```

and make it available to your deployment by adding this to `web.xml`:

```
<resource-env-ref>
  <resource-env-ref-name>
    app/Manager
  </resource-env-ref-name>
  <resource-env-ref-type>
    javax.inject.manager.Manager
  </resource-env-ref-type>
</resource-env-ref>
```

Tomcat only allows you to bind entries to `java:comp/env`, so the Manager will be available at `java:comp/env/app/Manager`

Web Beans also supports Servlet injection in Tomcat. To enable this, place the `webbeans-tomcat-support.jar` in `$TOMCAT_HOME/lib`, and add the following to your `META-INF/context.xml`:

```
<Listener className="org.jboss.webbeans.environment.tomcat.WebBeansLifecycleListener" />
```

16.4. Java SE

Apart from improved integration of the Enterprise Java stack, Web Beans also provides a state of the art typesafe, stateful dependency injection framework. This is useful in a wide range of application types, enterprise or otherwise. To facilitate this, Web Beans provides a simple means for executing in the Java Standard Edition environment independently of any Enterprise Edition features.

When executing in the SE environment the following features of Web Beans are available:

- `Simple Web Beans (POJOs)`
- `Typesafe Dependency Injection`
- `Application and Dependent Contexts`
- `Binding Types`
- `Stereotypes`
- `Decorators`
- `(TODO: Interceptors ?)`
- `Typesafe Event Model`

16.4.1. Web Beans SE Module

To make life easy for developers Web Beans provides a special module with a main method which will boot the Web Beans manager, automatically registering all simple Web Beans found on the classpath. This eliminates the need for application developers to write any bootstrapping code. The entry point for a Web Beans SE applications is a simple Web Bean which observes the standard `@Deployed Manager` event. The command line paramters can be injected using either of the following:

```
@Parameters List<String
> params;
@Parameters String[] paramsArray; // useful for compatability with existing classes
```

Here's an example of a simple Web Beans SE application:

```
@ApplicationScoped
public class HelloWorld
{
   @Parameters List<String
> parameters;

   public void printHello( @Observes @Deployed Manager manager )
   {
      System.out.println( "Hello " + parameters.get(0) );
   }
}
```

Web Beans SE applications are started by running the following main method.

```
java org.jboss.webbeans.environments.se.StartMain <args
>
```

If you need to do any custom initialization of the Web Beans manager, for example registering custom contexts or initializing resources for your beans you can do so in response to the `@Initialized Manager` event. The following example registers a custom context:

```
public class PerformSetup
{

   public void setup( @Observes @Initialized Manager manager )
```

```
  {
     manager.addContext( ThreadContext.INSTANCE );
  }
}
```

#

The command line parameters do not become available for injection until the `@Deployed Manager` event is fired. If you need access to the parameters during initialization you can do so via the `public static String getParameters()` method in `StartMain`.

JSR-299 extensions available as part of Web Beans

##

These modules are usable on any JSR-299 implementation, not just Web Beans!

17.1. Web Beans Logger

TODO

17.2. XSD Generator for JSR-299 XML deployment descriptors

TODO

A. # Web Bean RI

###Web Bean RI ### JBoss AS 5 ###### RI ##### EE ############ Glassfish ############### servlet ##### Tomcat############ EJB3.1 ##################################

#

SE ##### Web Bean#################### context ######Web Bean RI ######################### Web Bean RI # class

A.1. Web Beans RI SPI

The Web Beans SPI is located in `webbeans-spi` module, and packaged as `webbeans-spi.jar`. Some SPIs are optional, if you need to override the default behavior, others are required.

SPI ################### `Forwarding` class#

A.1.1. Web Bean

```
public interface WebBeanDiscovery {
  /**
   * Gets list of all classes in classpath archives with web-beans.xml files
   *
   * @return An iterable over the classes
   */
  public Iterable<Class<?>
> discoverWebBeanClasses();

  /**
   * Gets a list of all web-beans.xml files in the app classpath
   *
   * @return An iterable over the web-beans.xml files
   */
  public Iterable<URL
> discoverWebBeansXml();

}
```

Web Bean ### `web-bean.xml` ################ JSR-299 ##### 11.1 #########

A.1.2. EJB services

Web Bean RI #### EJB3 bean discovery # container######### EJB3 ##### `ejb-jar.xml`############ EJB ######## EJBDescriptor#

```
public interface EjbServices
{

  /**
   * Gets a descriptor for each EJB in the application
   *
   * @return The bean class to descriptor map
   */
  public Iterable<EjbDescriptor<?>> discoverEjbs();
```

```
public interface EjbDescriptor<T
> {

  /**
   * Gets the EJB type
   *
   * @return The EJB Bean class
   */
  public Class<T
> getType();

  /**
   * Gets the local business interfaces of the EJB
   *
   * @return An iterator over the local business interfaces
   */
  public Iterable<BusinessInterfaceDescriptor<?>
> getLocalBusinessInterfaces();

  /**
   * Gets the remote business interfaces of the EJB
   *
   * @return An iterator over the remote business interfaces
   */
  public Iterable<BusinessInterfaceDescriptor<?>
> getRemoteBusinessInterfaces();
```

```
    /**
     * Get the remove methods of the EJB
     *
     * @return An iterator over the remove methods
     */
    public Iterable<Method
> getRemoveMethods();

    /**
     * Indicates if the bean is stateless
     *
     * @return True if stateless, false otherwise
     */
    public boolean isStateless();

    /**
     * Indicates if the bean is a EJB 3.1 Singleton
     *
     * @return True if the bean is a singleton, false otherwise
     */
    public boolean isSingleton();

    /**
     * Indicates if the EJB is stateful
     *
     * @return True if the bean is stateful, false otherwise
     */
    public boolean isStateful();

    /**
     * Indicates if the EJB is and MDB
     *
     * @return True if the bean is an MDB, false otherwise
     */
    public boolean isMessageDriven();

    /**
     * Gets the EJB name
     *
     * @return The name
     */
    public String getEjbName();
```

```
}
```

`EjbDescriptor` ###################### EJB ############# metadata###################### `BusinessInterfaceDescriptor`########## EJB instance # interface class ## jndi ####

The resolution of `@EJB` and `@Resource` is delegated to the container. You must provide an implementation of `org.jboss.webbeans.ejb.spi.EjbServices` which provides these operations. Web Beans passes in the `javax.inject.manager.InjectionPoint` the resolution is for, as well as the `NamingContext` in use for each resolution request.

A.1.3. JPA services

Just as resolution of `@EJB` is delegated to the container, so is resolution of `@PersistenceContext`.

OPEN ISSUE: Web Beans also requires the container to provide a list of entities in the deployment, so that they aren't discovered as simple beans.

A.1.4. Transaction Services

The Web Beans RI must delegate JTA activities to the container. The SPI provides a couple hooks to easily achieve this with the `TransactionServices` interface.

```
public interface TransactionServices
{
  /**
   * Possible status conditions for a transaction. This can be used by SPI
   * providers to keep track for which status an observer is used.
   */
  public static enum Status
  {
    ALL, SUCCESS, FAILURE
  }

  /**
   * Registers a synchronization object with the currently executing
   * transaction.
   *
   * @see javax.transaction.Synchronization
   * @param synchronizedObserver
   */
  public void registerSynchronization(Synchronization synchronizedObserver);

  /**
   * Queries the status of the current execution to see if a transaction is
```

```
    * currently active.
    *
    * @return true if a transaction is active
    */
   public boolean isTransactionActive();
}
```

The enumeration `Status` is a convenience for implementors to be able to keep track of whether a synchronization is supposed to notify an observer only when the transaction is successful, or after a failure, or regardless of the status of the transaction.

Any `javax.transaction.Synchronization` implementation may be passed to the `registerSynchronization()` method and the SPI implementation should immediately register the synchronization with the JTA transaction manager used for the EJBs.

To make it easier to determine whether or not a transaction is currently active for the requesting thread, the `isTransactionActive()` method can be used. The SPI implementation should query the same JTA transaction manager used for the EJBs.

A.1.5. The application context

Web Beans expects the Application Server or other container to provide the storage for each application's context. The `org.jboss.webbeans.context.api.BeanStore` should be implemented to provide an application scoped storage. You may find `org.jboss.webbeans.context.api.helpers.ConcurrentHashMapBeanStore` useful.

A.1.6. Bootstrap and shutdown

The `org.jboss.webbeans.bootstrap.api.Bootstrap` interface defines the bootstrap for Web Beans. To boot Web Beans, you must obtain an instance of `org.jboss.webbeans.bootstrap.WebBeansBootstrap` (which implements `Boostrap`), tell it about the SPIs in use, and then request the container start.

The bootstrap is split into phases, bootstrap initialization and boot and shutdown. Initialization will create a manager, and add the standard (specification defined) contexts. Bootstrap will discover EJBs, classes and XML; add beans defined using annotations; add beans defined using XML; and validate all beans.

The bootstrap supports multiple environments. Different environments require different services to be present (for example servlet doesn't require transaction, EJB or JPA services). By default an EE environment is assumed, but you can adjust the environment by calling `bootstrap.setEnvironment()`.

To initialize the bootstrap you call `Bootstrap.initialize()`. Before calling `initialize()`, you must register any services required by your environment. You can do this by calling `bootstrap.getServices().add(JpaServices.class, new MyJpaServices())`. You must also provide the application context bean store.

Having called `initialize()`, the `Manager` can be obtained by calling `Bootstrap.getManager()`.

To boot the container you call `Bootstrap.boot()`.

To shutdown the container you call `Bootstrap.shutdown()`. This allows the container to perform any cleanup operations needed.

A.1.7. JNDI

The Web Beans RI implements JNDI binding and lookup according to standards, however you may want to alter the binding and lookup (for example in an environment where JNDI isn't available). To do this, implement `org.jboss.webbeans.resources.spi.NamingContext`:

```
public interface NamingContext extends Serializable {

   /**
    * Typed JNDI lookup
    *
    * @param <T
> The type
    * @param name The JNDI name
    * @param expectedType The expected type
    * @return The object
    */
   public <T
> T lookup(String name, Class<? extends T
> expectedType);

   /**
    * Binds an item to JNDI
    *
    * @param name The key to bind under
    * @param value The item to bind
    */
   public void bind(String name, Object value);

}
```

A.1.8.

Web Beans RI ######## classpath ######################### RI ### classloader ################################## `org.jboss.webbeans.spi.ResourceLoader#`

```
        public interface ResourceLoader {

   /**
    * Creates a class from a given FQCN
    *
    * @param name The name of the clsas
    * @return The class
    */
   public Class<?> classForName(String name);

   /**
    * Gets a resource as a URL by name
    *
    * @param name The name of the resource
    * @return An URL to the resource
    */
   public URL getResource(String name);

   /**
    * Gets resources as URLs by name
    *
    * @param name The name of the resource
    * @return An iterable reference to the URLS
    */
   public Iterable<URL
> getResources(String name);

}
```

A.1.9. Servlet injection

Java EE / Servlet does not provide any hooks which can be used to provide injection into Servlets, so Web Beans provides an API to allow the container to request JSR-299 injection for a Servlet.

To be compliant with JSR-299, the container should request servlet injection for each newly instantiated servlet after the constructor returns and before the servlet is placed into service.

To perform injection on a servlet call `WebBeansManager.injectServlet()`. The manager can be obtained from `Bootstrap.getManager()`.

A.2. # container

Web Bean RI ## container ######## API ############

Classloader ##

Web Bean RI ############################ Web Bean ######################### classloader

Servlet listener and filters

Web Bean ##### Servlet ############# Servlet # Web Bean ##################### `org.jboss.webbeans.servlet.WebBeansListener` ##### Servlet listener#

If you are integrating the Web Beans into a JSF environment you must register `org.jboss.webbeans.servlet.ConversationPropagationFilter` as a Servlet listener, either automatically, or through user configuration, for each Web Beans application which uses JSF. This filter can be registered for all Servlet deployment safely.

Session Bean ###

Web Bean ##### EJB ################ enterprise bean # Web Bean ############### EJB ################## `org.jboss.webbeans.ejb.SessionBeanInterceptor` ##### EJB

##

You must register the `SessionBeanInterceptor` as the inner most interceptor in the stack for all EJBs.

The `webbeans-core.jar`

If you are integrating the Web Beans into an environment that supports deployment of applications, you must insert the `webbeans-core.jar` into the applications isolated classloader. It cannot be loaded from a shared classloader.