Apache OpenWebBeans

User guide

ASL-License implementation of the JSR-299

© The Apache Software Foundation

Purpose and scope of this document

This document is part of the Apache OpenWebBeans release.

Abstract

OWB will be an ASL-licensed implementation of the Web Beans Specification which is defined as JSR-299.

In addition to the implementation of the specification, the project aims to create a new set of WebBeans components exposing functionality of other Apache Foundation projects.

Version control

This document is updated continuously. Major modifications on content or size will lead to new release numbers, whereas textual revisions are reflected as new level numbers. The following list shows the document's history.

Version	Date	Author	Reason of modification
0.1	2009/11/24	gerdogdu	Initial release

Table of Contents

List of Tables

Preface

In this guide we merged all kind of information needed to configure parts of the Apache OpenWebBeans (from now on simply called OWB throughout the rest of this document). A comprehensive index has been provided which should help you to quickly find the information you are looking for. We have also included a glossary that contains explanations of terms that are mentioned throughout this guide.

We hope that this user guide helps you to increase your knowledge and understanding of OWB. We are sure that it will make development of applications based on OWB a lot easier for you. Feedback is always welcome.

Conventions used in this document

Typography

the following typographical conventions are used in this document:

`database`
: used for database elements (tables, fields, keys, views,...)

`class` , `interface` , `method`
: used for (Java) classes, interfaces and methods

`<tag/>` , `attribute` , `value`
: used for XML elements (tags, attributes and values)

property `N` , *`parameter`* `N`
: used for property- and parameternames. The ' `N` ' (or other literal) is a variable part of the name, which usually stands for a running number.

`value`
: used for defined values. This can be a default value an example or the value of a constant.

`bean name="myBean"`
: used for inline sourcecode examples

```
<bean name="beanName" class="fqn.of.the.bean">
 <property name="beanPropertyName" ref="with its value"/>
</bean>
```

used for longer source code examples

ℹ	**Note** Indicates a tip, suggestion or general note

⚠	**Warning** Indicates a warning

Chapter 1. About OpenWebBeans

This chapter explains what is OWB and team.

Subtitle formatting

OpenWebBeans is an ASL-License implementation of the JSR-299, Contexts and Dependency Injection for the Java EE platform.

The specification can be found here: http://www.jcp.org/en/jsr/detail?id=299 .

1.1. The team

The following team members have been developing the OpenWebBeans.

Table 1.1. Team composition.

Id	Name	Email	Roles
`gerdogdu`	Gurkan Erdogdu	`gerdogdu at apache dot org`	PMC Chair
`kevan`	Kevan Miller	`kevan dot miller at gmail dot com`	PMC
`struberg`	Mark Struberg	`struberg at yahoo dot de`	PMC
`dblevins`	David Blevins	`david dot blevins at visi dot com`	PMC
`bergmark`	Joe Bergmark	`bergmark at gmail dot com`	PMC
`covener`	Eric Covener	`covener at gmail dot com`	PMC
`matzew`	Matthias Wessendorf	`matzew at apache dot org`	Committer

Chapter 2. Introduction

2.1. What is OpenWebBeans?

OpenWebBeans is an ASL-licensed implementation of the JSR-299: Contexts and Dependency Injection and JSR-330: Dependency Injection for Java.

2.1.1. OpenWebBeans Features

Warning

Currently OpenWebBeans does not fully implements JSR-299 specification but it is JSR-299 compatible.

Currently OpenWebBeans implementation supports the following main features.

- *JSR-330 Support:* It supports the JSR-330 based injections.
- *Managed Beans Support:* Supports the configuration and injection of Managed Beans
- *Session Beans Support in Embeddable OpenEJB in Tomcat 6.x:* It supports Session Beans using in embeddable Apache OpenEJB.
- *Producer Field and Method Support:* It supports producer based beans.
- *Injection of @Resource, @PersistenceUnit and @PersistenceContext:* Currently it is able to inject these resource types.
- *Java Messaging Service (JMS) Injection Support:* It supports injection of JMS Connection Factory, JMS Sessions etc.
- *Event/Observer Support:* It supports Event and Observers.
- *Interceptor and Decrotator Support for Managed Beans:* It supports interceptors and decorators for managed beans. Currently it does not support @InterceptorBinding style interceptors and decorators on Session beans.
- *Java SE Support:* It can be used in Java SE environments like Java Swing applications.
- *Java EE Web Application Support:* It can be used in Java EE Web containers like Tomcat, Jetty etc.

2.1.2. OpenWebBeans Plugin Architecture

OpenWebBeans have been developing as a small core package and including other pieces as a plugin. Each of the package and plugin modules are followings:

- *JSR-330 API Package:* Contains JSR-330 defined API. It will move into the Geronimo specifications in a near time.
- *JSR-299 API Package:* Contains JSR-299 defined API. It will move into the Geronimo specifications in a near time.
- *OpenWebBeans Implementation(Core) Package:* Contains core dependency injection related implementation.
- *EJB Plugin:* Session Beans injection implementation based on the Apache OpenEJB embeddable in Apache Tomcat

- *OpenWebBeans Geronimo Plugin:* Geronimo Java EE Server integration codes.
- *OpenWebBeans JMS Plugin:* Provides Java Messaging Service (JMS) related artifact injections.(Connection factories, Sessions etc.)
- *OpenWebBeans Resource Plugin:* Provides Java EE resource injections. Currently, it supports only for @PersistenceUnit, @PersistenceContext and @Resource annotations. To use @Resource based annotations without OpenEJB, you have to define it explicitly in web.xml via <resource-ref> element.

  Example:

  ```
  public class Injector{
     @Produces @Resource(name="myResource") myResource;
  }
  ```

  ```
  <resource-ref>
     <resource-name>myResource</resource-name>
     .....
  </resource-ref>
  ```
- *OpenWebBeans JSF Plugin:* Using of dependency injection service in Java Server Faces environment.
- *OpenWebBeans JPA Plugin:* It is deprecated now. Use OpenWebBeans Resource plugin instead.

2.1.2.1. How to use Plugins?

Plugins are discovered by the OpenWebBeans runtime using `java.util.ServiceLoader` class. Each plugin jar contains META-INF/services folder that contains plugin implementation class. Your sole responsbility to use plugin is to add related *Plugin JAR* into the application classpath.

2.1.2.2. OpenWebBeans SPI

OpenWebBeans interacts with Java EE specific services via SPI interfaces and their specific implementations. There are several SPI interfaces that are configured in the OpenWebBeans default configuration file. You can override and replace those SPI implementation with providing your new implementation. For example, OpenWebBeans SPI `org.apache.webbeans.spi.JNDIService` interface defines methods to interact with runtime JNDI provider. There are two different JNDIService implementation, one for example

```
org.apache.webbeans.spi.JNDIService=org.apache.webbeans.spi.se.JNDIServiceStaticImpl
org.apache.webbeans.spi.JNDIService=org.apache.webbeans.spi.ee.JNDIServiceEnterpriseImpl
```

First one is used in Java SE environment as a simple name-value `Map` implementation whereas the second one contains full blown container provided JNDI context implementation. You would look at *OpenWebBeans Configuration* section for getting all SPI interfaces and their explanations.

2.1.2.3. OpenWebBeans Configuration

OpenWebBeans has some configuration properties. Most of the configuration parameters are included within core jar file. But if you wish to update or replace existing configuration, please continue to reading

2.1.2.3.1. How Configuration Works?

OpenWebBeans' configuration is defined with Java properties file. The name of the default properties file is `"/META-INF/openwebbeans/openwebbeans-default.properties"` and it is included within `openwebbeans-impl.jar` file.

If you wish to override or replace the configuration parameters, create a properties file with name `"/META-INF/openwebbeans/openwebbeans.properties"` and put it into your application classpath. In your properties file, you can override the default configuration parameter values.

2.1.2.3.2. Configuration Parameters

OpenWebBeans defines a several comfuration parameters. Those are listed below:

- *org.apache.webbeans.spi.JNDIService:* JNDI service SPI interface. Configures JNDI Service implementation for getting objects from JNDI tree.
- *org.apache.webbeans.spi.JNDIService.jmsConnectionFactoryJndi:* Configures JMS ConnectionFactory object jndi name.
- *org.apache.webbeans.conversation.Conversation.periodicDelay:* Conversation removing thread periodic delay. Unused conversations are removed by the container periodically.
- *org.apache.webbeans.spi.TransactionService:* Transaction service SPI interface. Configures Transaction Service provider implementation for getting transaction manager and transaction.
- *org.apache.webbeans.resource.spi.ResourceService:* Resource service SPI interface. Defines how to inject Java EE based resources like @PersistenceContext, @PersistenceUnit, @WebServiceRef and @Resource.
- *org.apache.webbeans.spi.deployer.MetaDataDiscoveryService:* Discovery service SPI interfaces. Configures Discovery Service implementation for discovering JSR-299 beans and configuration xmls.
- *org.apache.webbeans.spi.deployer.UseEjbMetaDataDiscoveryService:* Defines whether or not use EJB session bean discovery. If application is run in the OpenEJB embeddable container, one must define this as true.
- *org.apache.webbeans.useOwbSpecificXmlConfig:* Use OpenWebBeans specific configuration or not. Generally it is not required to change this. Not portable!
- *org.apache.webbeans.fieldInjection.useOwbSpecificInjection:* Use @Inject annotation at injection fields or not. If it is set to true you can inject objects into beans without @Inject annotation. Not portable!
- *org.apache.webbeans.spi.conversation.ConversationService:* Conversation Service SPI. Configures Conversation Service implementation for getting conversation related informations. Currently conversation service is implemented for Java Server Faces based applications.
- *org.apache.webbeans.application.jsp:* Defines configuration parameter to specify that application is full Java Server Pages(JSP) application. If application is JSP, OpenWebBeans register ELResolver with JSP application.

- *org.apache.webbeans.spi.lifecycle.Lifecycle:* Bootstrap service SPI interface. Defines how OpenWebBeans container is bootstrapped.

2.1.2.3.3. Default Configuration Values

OpenWebBeans comes with default configuration parameters are set. As explained above, those default parameter values are defined in the provided jar file of the OpenWebBean. Default parameter values are the following

-
  ```
  org.apache.webbeans.spi.JNDIService=org.apache.webbeans.spi.se.JNDIServiceStaticImpl
  ```

Chapter 3. Using OpenWebBeans within Java EE Web

3.1. OpenWebBeans in 5 Minute

In this part of the user guide, we will look at how to configure and use OpenWebBeans in a *Java EE Web* projects.

3.1.1. Simple JSP Application

TODO

3.1.2. Using with JSF

TODO

3.1.3. Using with JPA

TODO

Chapter 4. Using OpenWebBeans within OpenEJB

4.1. OpenWebBeans within OpenEJB

TODO

Chapter 5. Using OpenWebBeans within Java SE

5.1. OpenWebBeans in Java SE Environment

TODO

Appendix A. Legal notes

Apache OpenWebBeans® is a registered trademark.

All rights, including in respect of the translation, printing and duplication of this document or parts thereof, are retained. No part of this document may be reproduced or processed with or without electronic means, in any form, without the prior written permission of The Apache Software Foundation. This includes for training purposes.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

All other brand names, designs, service marks and trademarks (whether or not registered) referenced or used herein are the property of their respective owners.

Glossary

This is the glossary of the OWB

E

Enterprise JavaBeans	Enterprise JavaBeans™ (EJB) is a managed, server-side component architecture for modular construction of enterprise applications.

O

Apache OpenEJB	OpenEJB is an open source, embeddable and lightweight EJB Container System and EJB Server, released under the Apache 2.0 License. OpenEJB has been integrated with J2EE application servers such as Geronimo[1], and WebObjects[2]. The role of OpenEJB is to create a runtime environment to run EJBs and let others access them in a unified way. Regardless of which application wants to access the beans, they all do it the same way, by accessing the appropriate Java Naming and Directory Interface (JNDI) context and looking up a home object. Think about JNDI as a catalog of names with objects bound to them. JNDI Initial Context is the starting point when working with the catalog--looking up a name or accessing the object bound to it. OpenEJB can run in two modes: Local (AKA IntraVM) Server and Remote Server.

Index