1. Introduction to CDI tooling ................................................................. 1
2. Creating a CDI Web Project ............................................................... 3
3. Adding CDI Support to a project .......................................................... 17
4. Wizards and Dialogs ........................................................................... 19
   4.1. Wizards ...................................................................................... 19
   4.2. Dialogs ...................................................................................... 20
       4.2.1. Open CDI Named Bean ..................................................... 20
5. Validation ............................................................................................ 23
6. Hyperlink navigation ........................................................................... 27
A. Revision History .................................................................................. 31
Chapter 1.

Introduction to CDI tooling

Context and Dependency Injection (CDI) tooling allows you to easily add and work with the CDI programming model for your web-based applications. This guide demonstrates how to add CDI to existing projects and create new projects with the CDI wizard. For information on the specifics of CDI outside of the tooling, see http://jcp.org/en/jsr/detail?id=299.
Creating a CDI Web Project

From the workbench go to File → New → Other.

Select a wizard

Create a CDI Web Project

Figure 2.1. Selecting the CDI Web Project wizard

The first screen of the wizard will ask you to define the attributes according to the options outlined in Table 2.1, “New CDI Web Project”. Click the Next button after defining the attributes in order to continue customizing your project or click the Finish button to accept all other defaults and create your application.
Chapter 2. Creating a CDI Web...

CDI Web Project

Create a standalone CDI Web project or add it to a new or existing Enterprise Application.

Project name: testCDI

Project location

- Use default location

Location: /home/irooskov/Work/JBT/workspace/testCDI

Target runtime

- JBoss EAP 6.0 Runtime
- New Runtime...

Dynamic web module version

3.0

Configuration

- Dynamic Web Project with CDI (Context and Dependency Injection)
- Modify...

Configures a Dynamic Web application to use Context and Dependency Injection (CDI)

EAR membership

- Add project to an EAR

EAR project name: testCDIEAR

- Working sets

- Add project to working sets

Working sets:

Figure 2.2. Main project creation screen
### Table 2.1. New CDI Web Project

<table>
<thead>
<tr>
<th>Field</th>
<th>Mandatory Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project name</td>
<td>yes</td>
<td>Enter a project name.</td>
</tr>
<tr>
<td>Project location</td>
<td>yes</td>
<td>The use default location checkbox will be selected automatically to define the project location as the Eclipse workspace. However you can define a custom path in the Location field by deselecting the Use default location checkbox.</td>
</tr>
<tr>
<td>Target runtime</td>
<td>no</td>
<td>Select a pre-configured runtime from the available options or configure a new runtime environment.</td>
</tr>
<tr>
<td>Dynamic web module version</td>
<td>yes</td>
<td>Select the required web module version.</td>
</tr>
<tr>
<td>Configuration</td>
<td>yes</td>
<td>Select the project configuration from the available options.</td>
</tr>
<tr>
<td>EAR membership</td>
<td>no</td>
<td>Add the project to an existing EAR project.</td>
</tr>
<tr>
<td>Working sets</td>
<td>no</td>
<td>Add the project to an existing working set.</td>
</tr>
</tbody>
</table>
Table 2.2. New CDI Web Project - Dynamic web module version

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0</td>
<td>This web module version corresponds to the JEE 6 implementation.</td>
</tr>
<tr>
<td>2.5</td>
<td>This web module version corresponds to the JEE 5 implementation.</td>
</tr>
<tr>
<td>2.4</td>
<td>This web module version corresponds to the J2EE 1.4 implementation.</td>
</tr>
<tr>
<td>2.3</td>
<td>This web module version corresponds to the J2EE 1.3 implementation.</td>
</tr>
<tr>
<td>2.2</td>
<td>This web module version corresponds to the J2EE 1.2 implementation.</td>
</tr>
</tbody>
</table>

Table 2.3. New CDI Web Project - Configuration

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Web Project with CDI</td>
<td>The default configuration option. Choosing this configuration will create a Dynamic Web Project with the CDI facet enabled.</td>
</tr>
<tr>
<td>Default configuration for the runtime specified in Target runtime</td>
<td>Depending on the runtime configuration, you may have to manually add CDI facet support through the Modify button.</td>
</tr>
<tr>
<td>JavaServer Faces v2.0 Project</td>
<td>Configures a project to use JSF v2.0. You will need to manually add CDI support to this configuration through the Modify button.</td>
</tr>
<tr>
<td>Minimal Configuration</td>
<td>The minimum required facets are installed. You will need to manually add CDI support to this configuration through the Modify button.</td>
</tr>
<tr>
<td>&lt;custom&gt;</td>
<td>Total manual configuration is required through the Modify button. Ensure you enable the CDI facet for CDI Web Project creation to succeed.</td>
</tr>
</tbody>
</table>

The Java configuration screen allows you to define Source folders on the build path and the Default output folder. The defaults for each are recommended, however you have the option to change them as you see fit.
Java

Configure project for building a Java application.

Source folders on build path:

- src

Default output folder:

- build/classes

Figure 2.3. Java configuration screen
The Web Module configuration screen allows you to define the Context root and the Content directory to be created, the option to Generate web.xml deployment descriptor is selected by default. The defaults for each are recommended, however you have the option to change them as you see fit.
Web Module
Configure web module settings.

Context root: testCDI

Content directory: WebContent

Generate web.xml deployment descriptor

Figure 2.4. Web Module configuration screen
Chapter 2. Creating a CDI Web...

The **Context and Dependency Injection (CDI) Facet** settings configuration screen allows you to **Generate beans.xml file**, selected by default. The *beans.xml* file is a bean archive descriptor that allows you to enable CDI services for a bean archive. It is recommended that you leave the option selected.
Context and Dependency Injection (CDI) Facet
Configure Context and Dependency Injection (CDI) Facet Settings

Generate beans.xml file: ✓

Figure 2.5. Web Module configuration screen
Chapter 2. Creating a CDI Web...

The final screen of the wizard pertains to **JSF Capabilities** configuration.
**JSF Capabilities**

Add JSF capabilities to this Web Project

**JSF Implementation Library**

*Type:* Library Provided by Target Runtime

The targeted runtime is able to provide the library required by this facet. Selecting this option will configure the project to use that library.

- **Configure JSF servlet in deployment descriptor**
  
  **JSF Configuration File:** /WEB-INF/faces-config.xml
  
  **JSF Servlet Name:** Faces Servlet
  
  **JSF Servlet Class Name:** javax.faces.webapp.FacesServlet
  
  **URL Mapping Patterns:** *

---

**Figure 2.6. Web Module configuration screen**
Table 2.4. JSF Capabilities

<table>
<thead>
<tr>
<th>Field</th>
<th>Mandatory Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSF Implementation Library Type</td>
<td>yes</td>
<td>Select an implementation library from the list provided.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This option adds support for the JSF implementation library, required for the project to function correctly. For a description of available options see: Table 2.5, “JSF Capabilities - JSF Implementation Library”.</td>
</tr>
<tr>
<td>Configure JSF servlet in deployment descriptor</td>
<td>no</td>
<td>Selected by default, decide whether to configure the deployment descriptor for the JSF servlet or not.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This option, if selected, opens up the ability to configure various deployment descriptor fields. For a description of fields see: Table 2.6, “JSF Capabilities - JSF Servlet Configuration”.</td>
</tr>
</tbody>
</table>

Table 2.5. JSF Capabilities - JSF Implementation Library

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Library Provided by Target Runtime</td>
<td>If you selected a target runtime on the first screen, this option will configure the project to use the required facet library that is provided by the target runtime.</td>
</tr>
<tr>
<td>User Library</td>
<td>Selecting the User Library option allows you to specify one or more custom user libraries to be used. At least one must be selected for the wizard to complete successfully.</td>
</tr>
<tr>
<td>Disable Library Configuration</td>
<td>If you choose to disable library configuration you will have to configure the project classpath at a later time, by alternate means, before running the project.</td>
</tr>
</tbody>
</table>

Table 2.6. JSF Capabilities - JSF Servlet Configuration

<table>
<thead>
<tr>
<th>Field</th>
<th>Mandatory Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSF Configuration File</td>
<td>Yes</td>
<td>Enter the location of the JSF configuration file (the default is usually correct).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You must specify where the JSF configuration file is located.</td>
</tr>
<tr>
<td>JSF Servlet Name</td>
<td>Yes</td>
<td>Specify a name for your JSF servlet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The name of the servlet.</td>
</tr>
<tr>
<td>JSF Servlet Class Name</td>
<td>No</td>
<td>Specify a name for your servlet class.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Naming your servlet class will allow for easy reference to it when you are coding for your project in the future.</td>
</tr>
<tr>
<td>Field</td>
<td>Mandatory Instruction</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>URL Mapping</td>
<td>No</td>
<td>A default mapping pattern is provided. You are able to remove and add patterns as you wish.</td>
</tr>
<tr>
<td>Patterns</td>
<td></td>
<td>This option allows for the creation of constant, readable URLs, based on patterns, that map to your project.</td>
</tr>
</tbody>
</table>

Click the **Finish** button to create your CDI Web Project.
Adding CDI Support to a project

CDI support can be added either to an existing project or during project creation.

To add CDI support to an existing project, select the project, and right-click to bring up the context menu. From the menu select Configure → Add CDI support. The following preferences page will now display:

![Figure 3.1. CDI project settings](image1)

You can also add CDI support to a project during project creation. To do so, click the Modify button beside the selected Configuration on the first screen of the project wizard.

![Figure 3.2. Modifying project configuration](image2)
From the **Project Facets** screen, add CDI to the project by selecting the checkbox beside the facet **CDI (Contexts and Dependency Injection)**. Click the **OK** button to accept the configuration change and return to the project creation wizard.

**Figure 3.3. Adding CDI facet support**
Chapter 4.

Wizards and Dialogs

4.1. Wizards

A set of wizards are available for creating a range of basic CDI artifacts. The wizards available are for the following types:

- Annotation Literal
- Bean
- Decorator
- Interceptor
- Interceptor Binding Annotation
- Qualifier Annotation
- Scope Annotation
- Stereotype Annotation
Chapter 4. Wizards and Dialogs

Figure 4.1. Artifact wizard options

4.2. Dialogs

4.2.1. Open CDI Named Bean

The Open CDI Named Bean dialog allows you to open the resource which declares a specified expression language (EL) bean name.
Open CDI Named Bean

Figure 4.2. Open CDI Named Bean dialog

This dialog is available from the Java EE perspective's tool bar.

Figure 4.3. Toolbar icon
Validation

CDI validation covers nearly all of the rules declared in the JSR-299 specification. For every type of validation rule, you can set if it should be interpreted as an error, warning or just ignored. This is achieved by navigating to Window → Preferences → JBoss Tools → CDI → CDI Validator.
Figure 5.1. Validation settings
The CDI validator can check JSP, XHTML, beans.xml and java source files. Some validation issues can be fixed through the Quick Fixes menu (Ctrl+1).

```java
@Stateful
public class FooProducer implements FooProducerLocal, FooProducerLocal2, FooProducerLocal3 {
    // Produces Foo
    public Foo createFoo() { return new Foo(); }
}
```

**Figure 5.2. Quick Fixes**
Chapter 6.

Hyperlink navigation

Hyperlinks (OpenOn integration) allows you to quickly navigate between source files. Pressing Ctrl+Left-Click or F3 on any expression language (EL) declaration will navigate you to the corresponding java source.

CDI Tools offers you a list of navigation alternatives for the bean that will be injected.

![Image]

**Figure 6.1. Example of navigation alternatives**

If multiple eligible beans are detected for the injection, you will see an option to **Show All Assignable Beans**.

![Image]

**Figure 6.2. Show All Assignable Beans**

Selecting this option will display a dialog containing all eligible beans.
Figure 6.3. Multiple eligible beans

Corresponding classes and interfaces that are mentioned in beans.xml can be opened through pressing Ctrl+Left-Click on the type name.

Observers and Events can be navigated to through the same method as well; Show CDI Observer Methods and Show CDI Events.
Figure 6.4. Observers example

CDI Observer Methods

- MyGenericBean2.myObserver() - org.jboss.generic
- MyGenericBean2.myObserver2() - org.jboss.generic

Figure 6.5. Events example

CDI Events

- MyBeanInjections.event2 - org.jboss.generic
- MyBeanInjections.event3 - org.jboss.generic

Producer and Disposer methods, if available, will appear in the context menu for an injection.
Figure 6.6. Producer example

Figure 6.7. Disposer example
Appendix A. Revision History

Revision History
Revision 6.0.1-1         Tue Aug 27 2013         MichelleMurray<mmurray@redhat.com>
TOOLSDOC-382: Rebuilt for Customer Portal, content unchanged
Revision 1-0            Fri Jan 06 2012            IsaacRooskov<irooskov@redhat.com>
Initial creation of book