EasyBeans Developer's guide

Florent BENOIT, OW2 consortium

EasyBeans Developer's guide

by Florent BENOIT

Publication date $Id: developerguide.xml 216 2006-03-16 19:01:07Z benoitf $

Abstract

The EasyBeans developer guide is intended for developers wanting to work with the source distribution of EasyBeans. People wanted to contribute to EasyBeans should read this documentation.

Table of Contents

Chapter 1. Building EasyBeans From Source.

1.1. Requirements

1.1.1. JDK

A Java SE 5 [http://java.sun.com/javase/downloads/index_jdk5.jsp] is required to build EasyBeans. Make sure that the JDK used to build EasyBeans is compliant with the new Java 5 features.

1.1.2. Maven

The maven tool is used with `pom.xml` files to build EasyBeans. This tool is available at http://maven.apache.org. The 2.0.7 or later version is recommanded

1.1.3. Ant

Some tests are not yet available as maven tests, then the ant tool is needed to run the tests. The ant tool is available at http://ant.apache.org

1.1.4. TestNG

The test suite of EasyBeans uses the TestNG tool. This tool is available at http://www.testng.org.

1.1.5. Clover

The test suite of EasyBeans uses Clover, which is a code-coverage, analysis tool. Atlassian has granted licenses to open source projects. Refer to http://www.atlassian.com/software/clover/ for more about Clover.

1.2. Optional Requirements

1.2.1. Eclipse

The EasyBeans project provides .project and .classpath for Eclipse 3.1 or greater. A project is ready to use once the source has been imported using the Eclipse tool. Eclipse tool is available at http://www.eclipse.org.

1.2.2. Eclipse Plugins

1.2.2.1. Maven Plugin

EasyBeans code is using maven tool. In order to download Maven2 dependencies in Eclipse, the M2 Eclipse plugin can be used. This plugin is available at http://m2eclipse.sonatype.org/

1.2.2.2. Checkstyle Plugin

The eclipse-checkstyle plugin is used to check the javadoc of Easybeans project. A warning will print if the EasyBeans coding convention is not used. This plugin is available at http://eclipse-cs.sourceforge.net.

1.2.2.3. AnyEdit Plugin

As part of the EasyBeans coding convention, the use of tabulation characters is disallowed. Files should contain only spaces. The AnyEdit plugin allows tabs to be converted to spaces when saving the file. Also, trailing spaces can be removed automatically.

This plugin is available at http://andrei.gmxhome.de/anyedit/.

1.2.2.4. Asm Plugin

EasyBeans uses bytecode enhancement. This is done using the OW2 ASM project. [http://asm.objectweb.org] ASM provides a plugin that allows the ASM code of a given class to be obtained. The plugin is available at http://asm.objectweb.org/eclipse/index.html.

1.2.2.5. TestNG Plugin

The EasyBeans test suite uses TestNG. A plugin is available for Eclipse: http://testng.org/doc/eclipse.html.

1.3. Compiling EasyBeans

To compile EasyBeans, launch the command **mvn** in the root directory of the project (named easybeans by default) being launched.

Note

The default maven goal is install if not specified.

Once the command has been run successfully, the maven artifacts generated by maven are available in the maven local repository. The `target` directories contain the generated jars or assemblies.

mvn clean is used to clean the generated classes.

1.4. Maven assemblies

EasyBeans build generates several assemblies. Assemblies are located in the `assemblies` folder.

1.4.1. End-User assemblies

The packages are available with Apache OpenJPA [http://openjpa.apache.org] or Hibernate Entity Manager [http://www.hibernate.org] as persistence provider. Packages contain examples and are available with the zip or tgz format.

1.4.1.1. Jetty

EasyBeans can be launched within Jetty [http://jetty.mortbay.org] web container. The assemblies for Jetty are available in the `assemblies/distrib/jetty/target` folder.

1.4.1.2. JOnAS

EasyBeans can be launched within JOnAS 4 J2EE application server. The assemblies for JOnAS are available in the `assemblies/distrib/jonas/target` folder.

Note

EasyBeans is already included by default in JOnAS 5.0

1.4.1.3. Tomcat

EasyBeans can be launched within Apache Tomcat [http://tomcat.apache.org] web container. The assemblies for Jetty are available in the `assemblies/distrib/tomcat/target` folder.

1.4.1.4. UberJar

EasyBeans can be launched without any container. This is done by using the uberjar file. EasyBeans is launched by using **java -jar easybeans-uberjar-xxx.jar** command.

1.4.2. Java EE modules assemblies

These assemblies are simple Java EE modules, without example or documentation. These assemblies are then packaged into End-User assemblies.

1.4.2.1. War module

The war module is then used for Apache Tomcat or Jetty web container.

1.4.2.2. Rar module

This module is used for JOnAS 4 J2EE Application server.

Chapter 2. Getting EasyBeans From the SVN Repository

Anyone can check out source code from the SVN server using the following command (for GUI SVN client use, configuration values are the same as for command line use):

svn checkout svn://svn.forge.objectweb.org/svnroot/easybeans/trunk/easybeans

Chapter 3. Running EasyBeans server.

3.1. Requirements

Review the requirements discussed in Chapter 1, "Building EasyBeans from Source."

3.2. Running

The `build.xml` file located in the project root will be used to launch the EasyBeans server. This file is contained in the source distribution.

Use the following command: **ant run.server**

The EasyBeans server will be launched and the following output will be printed:

```
$ -ant -run.server
Buildfile: -build.xml

init-maven-task:

run.server:
 - - - - -[java] -9/29/
07 -3:45:20 -PM -(I) -PolicyProvider.init -: -Using -EasyBeans -policy -provider -'org.ow2.easybeans.security.jac
 - - - - -[java] -9/29/
07 -3:45:20 -PM -(I) -PolicyProvider.init -: -Using -EasyBeans -PolicyConfigurationFactory -provider -and -EasyBe
 - - - - -[java] -9/29/07 -3:45:20 -PM -(W) -Embedded.configure -: -Directory -/home/benoitf/
workspace/easybeans/easybeans-deploy -created.
 - - - - -[java] -9/29/
07 -3:45:20 -PM -(I) -Embedded.configure -: -Using -directories -'[easybeans-
deploy]' -as -deploy -directories
 - - - - -[java] -9/29/
07 -3:45:20 -PM -(I) -TraceCarol.infoCarol -: -Name -service -for -jrmp -is -started -on -port -1099
 - - - - -[java] -9/29/07 -3:45:21 -PM -(I) -Current.<init> -: -JOTM -2.0.10
 - - - - -[java] -9/29/
07 -3:45:21 -PM -(I) -JOTMComponent.start -: -Register -javax.transaction.UserTransaction -as -transaction -manag
 - - - - -[java] -9/29/
07 -3:45:21 -PM -(I) -JoramComponent.start -: -Joram -version -'5.0.6' -started -on -localhost:16030.
 - - - - -[java] -9/29/
07 -3:45:21 -PM -(I) -HSQLDBComponent.start -: -Starting -'HSQLDB -server' -'1.8.0' -on -port -'9001'
 - - - - -[java] -9/29/
07 -3:45:21 -PM -(I) -HSQLDBComponent.start -: -HSQLDB -server -started -with -URL -jdbc:hsqldb:hsql:/
/localhost:9001/jdbc_1
 - - - - -[java] -9/29/
07 -3:45:21 -PM -(I) -HSQLDBComponent.start -: -Starting -'HSQLDB -server' -'1.8.0' -on -port -'9002'
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -HSQLDBComponent.start -: -HSQLDB -server -started -with -URL -jdbc:hsqldb:hsql:/
/localhost:9002/jdbc_2
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -MailComponent.start -: -Binding -javax.mail.Session -Mail -factory -with -JNDI -name -mailS
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -MailComponent.start -: -Binding -javax.mail.internet.MimePartDataSource -Mail -factory -wit
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -JDBCPoolComponent.start -: -DS -'jdbc_1', -URL -'jdbc:hsqldb:hsql://
localhost:9001/jdbc_1', -Driver -= -'org.hsqldb.jdbcDriver'.
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -JDBCPoolComponent.start -: -DS -'jdbc_2', -URL -'jdbc:hsqldb:hsql://
localhost:9002/jdbc_2', -Driver -= -'org.hsqldb.jdbcDriver'.
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -SmartClientEndPointComponent.start -: -SmartClient -Endpoint -listening -on -port -'2503'.
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -QuartzScheduler.<init> -: -Quartz -Scheduler -v.1.6.0 -created.
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -RAMJobStore.initialize -: -RAMJobStore -initialized.
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -StdSchedulerFactory.instantiate -: -Quartz -scheduler -'EasyBeans' -initialized -from -an -
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -StdSchedulerFactory.instantiate -: -Quartz -scheduler -version: -1.6.0
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -QuartzScheduler.start -: -Scheduler -EasyBeans_$_NON_CLUSTERED -started.
```

```
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -ComponentManager.startComponents -: -[ -Component(s) -started -: -Carol -JOTM -Joram -HSQLD
 - - - - -[java] -9/29/
07 -3:45:22 -PM -(I) -JMXRemoteHelper.init -: -Creating -JMXRemote -connector -with -URL -'service:jmx:rmi:/
//jndi//EasyBeansConnector'
 - - - - -[java] -9/29/07 -3:45:22 -PM -(I) -Embedded.start -: -Startup -of -EasyBeans -'1.0.0-
SNAPSHOT' -was -done -in -'1,954' -ms.
 - - - - -[java] -9/29/07 -3:45:22 -PM -(I) -Embedded.start -: -Waiting -requests...
```

EasyBeans is now launched and it is ready to handle EJBs.

Chapter 4. Using the Examples

4.1. Compiling the Examples

4.1.1. Requirements

Before running the examples, be sure to follow the requirements for compiling and running these EasyBeans examples.

4.1.2. Compile

The ant tool is used to build the examples. To compile the examples, use the `build.xml` file that is located in the `examples` directory.

The command **ant install_all_examples** must be launched in the `examples` directory:

```
$ -ant -install_all_examples
Buildfile: -build.xml

install_all_examples:

init-maven-task:

init:
 - - - -[mkdir] -Created -dir: -/home/benoitf/workspace/easybeans/output/example-classes
 - - - -[mkdir] -Created -dir: -/home/benoitf/workspace/easybeans/clients
 - - - -[mkdir] -Created -dir: -/home/benoitf/workspace/easybeans/webapps

compile:
 - - - -[javac] -Compiling -4 -source -files -to -/home/benoitf/workspace/easybeans/output/
example-classes

ejb:

ejb-standalone:
[easybeans:ejb] -Building -Ejb -in -'/home/benoitf/workspace/easybeans/easybeans-deploy/
entitybean.jar'.
[easybeans:ejb] -Building -jar: -/home/benoitf/workspace/easybeans/easybeans-deploy/
entitybean.jar

war:

ear:

client:

client-standalone:
[easybeans:client] -Building -Client -in -'/home/benoitf/workspace/easybeans/clients/client-
entitybean.jar'.
[easybeans:client] -Building -jar: -/home/benoitf/workspace/easybeans/clients/client-
entitybean.jar

install:

init-maven-task:

init:

compile:
 - - - -[javac] -Compiling -3 -source -files -to -/home/benoitf/workspace/easybeans/output/
example-classes

ejb:

ejb-standalone:
[easybeans:ejb] -Building -Ejb -in -'/home/benoitf/workspace/easybeans/easybeans-deploy/
mdb.jar'.
[easybeans:ejb] -Building -jar: -/home/benoitf/workspace/easybeans/easybeans-deploy/mdb.jar

war:

ear:
```

```
client:

client-standalone:
[easybeans:client] -Building -Client -in -'/home/benoitf/workspace/easybeans/clients/client-
mdb.jar'.
[easybeans:client] -Building -jar: -/home/benoitf/workspace/easybeans/clients/client-mdb.jar

install:

init-maven-task:

init:

compile:
 - - - -[javac] -Compiling -7 -source -files -to -/home/benoitf/workspace/easybeans/output/
example-classes

ejb:

ejb-standalone:
[easybeans:ejb] -Building -Ejb -in -'/home/benoitf/workspace/easybeans/easybeans-deploy/
migration21.jar'.
[easybeans:ejb] -Building -jar: -/home/benoitf/workspace/easybeans/easybeans-deploy/
migration21.jar

war:

ear:

client:

client-standalone:
[easybeans:client] -Building -Client -in -'/home/benoitf/workspace/easybeans/clients/client-
migration21.jar'.
[easybeans:client] -Building -jar: -/home/benoitf/workspace/easybeans/clients/client-
migration21.jar

install:

init-maven-task:

init:

compile:
 - - - -[javac] -Compiling -5 -source -files -to -/home/benoitf/workspace/easybeans/output/
example-classes

ejb:

ejb-standalone:
[easybeans:ejb] -Building -Ejb -in -'/home/benoitf/workspace/easybeans/easybeans-deploy/
security.jar'.
[easybeans:ejb] -Building -jar: -/home/benoitf/workspace/easybeans/easybeans-deploy/security.jar

war:

ear:

client:

client-standalone:
[easybeans:client] -Building -Client -in -'/home/benoitf/workspace/easybeans/clients/client-
security.jar'.
[easybeans:client] -Building -jar: -/home/benoitf/workspace/easybeans/clients/client-
security.jar

install:

init-maven-task:

init:

compile:
 - - - -[javac] -Compiling -7 -source -files -to -/home/benoitf/workspace/easybeans/output/
example-classes

ejb:

ejb-standalone:
[easybeans:ejb] -Building -Ejb -in -'/home/benoitf/workspace/easybeans/easybeans-deploy/
stateless.jar'.
[easybeans:ejb] -Copying -5 -files -to -/home/benoitf/workspace/easybeans/easybeans-deploy/
stateless.jar
```

```
war:

war-standalone:
[easybeans:war] -Building -War -in -'/home/benoitf/workspace/easybeans/webapps/web.war'.
[easybeans:war] -Copying -6 -files -to -/home/benoitf/workspace/easybeans/webapps/web.war/WEB-
INF/classes
[easybeans:war] -Copying -1 -file -to -/home/benoitf/workspace/easybeans/webapps/web.war/WEB-INF

ear:

client:

client-standalone:
[easybeans:client] -Building -Client -in -'/home/benoitf/workspace/easybeans/clients/client-
stateless.jar'.
[easybeans:client] -Building -jar: -/home/benoitf/workspace/easybeans/clients/client-
stateless.jar

install:

init-maven-task:

init:

compile:
 - - - -[javac] -Compiling -3 -source -files -to -/home/benoitf/workspace/easybeans/output/
example-classes

ejb:

ejb-standalone:
[easybeans:ejb] -Building -Ejb -in -'/home/benoitf/workspace/easybeans/easybeans-deploy/
stateful.jar'.
[easybeans:ejb] -Building -jar: -/home/benoitf/workspace/easybeans/easybeans-deploy/stateful.jar

war:

ear:

client:

client-standalone:
[easybeans:client] -Building -Client -in -'/home/benoitf/workspace/easybeans/clients/client-
stateful.jar'.
[easybeans:client] -Building -jar: -/home/benoitf/workspace/easybeans/clients/client-
stateful.jar

install:

init-maven-task:

init:

compile:
 - - - -[javac] -Compiling -6 -source -files -to -/home/benoitf/workspace/easybeans/output/
example-classes

ejb:

ejb-standalone:
[easybeans:ejb] -Building -Ejb -in -'/home/benoitf/workspace/easybeans/easybeans-deploy/
timer.jar'.
[easybeans:ejb] -Building -jar: -/home/benoitf/workspace/easybeans/easybeans-deploy/timer.jar

war:

ear:

client:

client-standalone:
[easybeans:client] -Building -Client -in -'/home/benoitf/workspace/easybeans/clients/client-
timer.jar'.
[easybeans:client] -Building -jar: -/home/benoitf/workspace/easybeans/clients/client-timer.jar

install:

init-maven-task:

init:

compile:
```

```
 - - - -[javac] -Compiling -5 -source -files -to -/home/benoitf/workspace/easybeans/output/
example-classes

ejb:

ejb-standalone:

war:

war-standalone:

ear:
[easybeans:ear] -Building -Ear -in -'/home/benoitf/workspace/easybeans/easybeans-deploy/
ear3.ear'.
 - - - - - -[ejb] -Building -Ejb -in -'/tmp/easybeans-ant33717.tmp'.
 - - - - - -[ejb] -Building -jar: -/tmp/easybeans-ant33717.tmp
 - - - - - -[war] -Building -War -in -'/tmp/easybeans-ant33718.tmp'.
 - - - - - -[war] -Building -war: -/tmp/easybeans-ant33718.tmp
[easybeans:ear] -Building -jar: -/home/benoitf/workspace/easybeans/easybeans-deploy/ear3.ear

client:

install:

BUILD -SUCCESSFUL
Total -time: -22 -seconds
```

The examples are copied under the `easybeans-deploy/` folder of the project and are available for the deployment.

Note

If the EasyBeans server is running, it will detect these new applications and deploy them automatically.

4.2. Running Examples

Each example has its own `build.xml` file; this allows each example to be run independently.

4.2.1. Stateless Session Bean

The `build.xml` file for this example is located in the `examples/statelessbean` folder.

4.2.1.1. Description

This example is a stateless session bean. It contains a `helloWorld()` method that displays text on the server side. Additionally, it demonstrates the use of EJB3 annotation, such as `@Stateless`.

The `trace()` method is annotated with `@AroundInvoke` EJB3 annotation. This method will be called at each call on a business method. The business methods are defined in the interface implemented by the SessionBean class.

The signature of the method annotated by `@AroundInvoke` when it is defined in the bean class, must follow this signature:

```
  (private|protected|public) Object methodName(
            InvocationContext

            invocationContext
          )
    throws Exception;
```

Note

As a new feature of EJB3, the bean's interface does not need to extend the Remote interface.

4.2.1.2. Running the Server

If the server is not available, it must be run by following the steps described in Chapter 3, "Running the EasyBeans Server" of the developer's guide.

4.2.1.3. Deploying the Bean

The stateless session bean must be deployed. If the bean has been installed in the `easybeans-deploy` folder, this is done automatically.

On the server side, the following output should display:

```
 - - - - -[java] -5/16/
07 -10:59:32 -AM -(I) -AbsDeployer.deployEJB -: -Deploying -EJB3DeployableImpl[archive=easybeans-
deploy/stateless.jar]
 - - - - -[java] -5/16/
07 -10:59:32 -AM -(I) -JContainer3.start -: -Container -started -in -: -408 -ms
```

Once this information is displayed on the screen, the container is ready to receive client calls.

4.2.1.4. Running the Client

Once the container has been started, the client can be launched.

Run the client with the following ant command: **ant run.client**

If the client runs successfully, the following output is displayed:

```
 - - - - -[java] -Calling -helloWorld -method...
 - - - - -[java] -Add -1 -+ -2...
 - - - - -[java] -Sum -= -'3'.
```

Note

In the client's code, the use of the PortableRemoteObject.narrow() call is no longer required.

4.2.2. Stateful Session Bean

The `build.xml` file for this example is located in the `examples/statefulbean` folder.

4.2.2.1. Description

This is an example of a stateful session bean using the `SessionSynchronization` interface.

It uses the `@Stateful` annotation and uses the default transaction model, which is REQUIRED.

4.2.2.2. Running the Server

If the server is not available, it must be run by following the steps described in Chapter 3, "Running the EasyBeans Server" of the developer's guide.

4.2.2.3. Deploying the Bean

The stateful session bean must be deployed. It is done automatically if the bean has been installed in the `easybeans-deploy` folder.

On the server side, the following output should be seen:

```
 - - - - -[java] -5/16/
07 -10:59:37 -AM -(I) -AbsDeployer.deployEJB -: -Deploying -EJB3DeployableImpl[archive=easybeans-
deploy/stateful.jar]
```

```
 - - - - -[java] -5/16/
07 -10:59:37 -AM -(I) -JContainer3.start -: -Container -started -in -: -94 -ms
```

Once this information is displayed on the screen, the container is ready to receive client calls.

4.2.2.4. Running the Client

Once the container has been started, the client can be launched.

Run the client with the following ant command: **ant run.client**

If the client runs successfully, the following output is displayed:

```
 - - - - -[java] -Start -a -first -transaction
 - - - - -[java] -First -request -on -the -new -bean
 - - - - -[java] -Second -request -on -the -bean
 - - - - -[java] -Commit -the -transaction
 - - - - -[java] -Start -a -second -transaction
 - - - - -[java] -Buy -50 -amount.
 - - - - -[java] -Rollback -the -transaction
 - - - - -[java] -after -rollback, -value -= -30
 - - - - -[java] -Request -outside -any -transaction
 - - - - -[java] -Check -that -value -= -30
 - - - - -[java] -ClientStateful -OK. -Exiting.
```

4.2.3. Entity Bean

The `build.xml` file for this example is located in the `examples/entitybean` folder.

4.2.3.1. Description

This is an example of an entity bean. It describes how to use the new Java Persistence Model of an EJB3 persistence provider. To access EJB3 entities that are POJO, a stateless session bean is used. It is a facade bean.

The Entity class is a POJO class annotated with `@Entity`. The entities class is managed by the persistence provider.

Currently, the persistence provider is supplied by the Hibernate product, but the ObjectWeb Speedo product should be available soon. Users will have the choice between providers.

This example uses the `@Stateful` annotation and uses the default transaction model, which is REQUIRED.

The example shows an entity bean using EJB3 Hibernate-prototype persistence provider.

4.2.3.2. Running the Server

If the server is not available, it must be run following the steps described in Chapter 3, "Running the EasyBeans Server" of the developer's guide.

4.2.3.3. Deploying the Bean

The entity bean must be deployed. It is done automatically if the bean has been installed in the `easybeans-deploy` folder.

On the server side, the following output should be seen:

```
 - - - - -[java] -5/16/
07 -10:59:36 -AM -(I) -AbsDeployer.deployEJB -: -Deploying -EJB3DeployableImpl[archive=easybeans-
deploy/entitybean.jar]
 - - - - -[java] -5/16/
07 -10:59:36 -AM -(I) -JPersistenceUnitInfoHelper.getPersistenceUnitInfo -: -No -persistence -provider -was -set,
 - - - - -[java] -5/16/
07 -10:59:36 -AM -(I) -JPersistenceUnitInfoHelper.getPersistenceUnitInfo -: -Found -a -default -configuration -fo
```

```
 - - - - -[java] -5/16/
07 -10:59:36 -AM -(I) -JPersistenceUnitInfoHelper.getPersistenceUnitInfo -: -Setting -the -property -hibernate.tr
 - - - - -[java] -5/16/
07 -10:59:36 -AM -(I) -JPersistenceUnitInfoHelper.getPersistenceUnitInfo -: -Setting -the -property -hibernate.ca
 - - - - -[java] -5/16/
07 -10:59:36 -AM -(I) -Ejb3Configuration.configure -: -Processing -PersistenceUnitInfo -[
 - - - - -[java] -name: -entity
 - - - - -[java] -...]
 - - - - -[java] -5/16/
07 -10:59:36 -AM -(I) -Ejb3Configuration.scanForClasses -: -found -EJB3 -Entity -bean: -org.objectweb.easybeans.e
 - - - - -[java] -5/16/
07 -10:59:36 -AM -(I) -Ejb3Configuration.scanForClasses -: -found -EJB3 -Entity -bean: -org.objectweb.easybeans.e
 - -...
 - - - - -[java] -5/16/
07 -10:59:36 -AM -(I) -JContainer3.start -: -Container -started -in -: -412 -ms
```

Once this information is displayed on the screen, the container is ready to receive client calls.

4.2.3.4. Running the Client

Once the container has been started, the client can be launched.

The client is run with the following ant command: **ant run.client**

If the client runs successfully, the following output is displayed:

```
 - - - - -[java] -Employee -with -id -1 -= -Florent
 - - - - -[java] -Employee -with -id -2 -= -Whale
```

4.2.3.5. Properties for the persistence

These properties are defined in the `META-INF/persistence.xml` file.

4.2.3.5.1. JDBC Dialect

By default, the dialect used to communicate with the database is set to HSQL, as it is embedded in EasyBeans.

This dialect configuration is done with the following properties:

```
 - - - - - -<property -name="hibernate.dialect" -value="org.hibernate.dialect.HSQLDialect" -/>
 - - - - - -<property -name="toplink.target-database" -value="HSQL"/>
 - - - - - -<property -name="openjpa.jdbc.DBDictionary" -value="hsql"/>
```

These properties are for Hibernate, Apache OpenJPA and Oracle TopLink Essentials.

4.2.3.5.2. Database (tables)

By default, the tables are created and the database is empty after loading the entity beans.

This configuration is done with the following properties:

```
 - - - - - -<property -name="hibernate.hbm2ddl.auto" -value="create-drop"/>
 - - - - - -<property -name="toplink.ddl-generation" -value="drop-and-create-tables"/>
 - - - - - -<property -name="toplink.ddl-generation.output-mode" -value="database"/>
 - - - - - -<property -name="openjpa.jdbc.SynchronizeMappings" -value="buildSchema(ForeignKeys=true)"/
>
```

In order to keep data in the database, this property should be changed.

4.2.4. Message Driven Bean

The `build.xml` file for this example is located in the `examples/messagedrivenbean` folder.

4.2.4.1. Description

This is an example of a message driven bean. It describes how to use a JMS message driven bean.

The class is a class annotated with `@MessageDriven`. Then, it is mapped to a JMS queue through the properties of this annotation.

```
@MessageDriven(activationConfig -= -{
 - - - - - - - -@ActivationConfigProperty(propertyName -= -"destination", -propertyValue -= -"SampleQueue"),
 - - - - - - - -@ActivationConfigProperty(propertyName -= -"destinationType", -propertyValue -= -"javax.jms.Queue
 - - - - - - - -}
)
```

The Message Driven Bean will receive message from the SampleQueue queue.

4.2.4.2. Running the Server

If the server is not available, it must be run following the steps described in Chapter 3, "Running the EasyBeans Server" of the developer's guide.

4.2.4.3. Deploying the Bean

The entity bean must be deployed. It is done automatically if the bean has been installed in the `easybeans-deploy` folder.

On the server side, the following output should be seen:

```
5/16/
07 -2:42:24 -PM -(I) -AbsDeployer.deployEJB -: -Deploying -EJB3DeployableImpl[archive=easybeans-
deploy/mdb.jar]
5/16/07 -2:42:24 -PM -(I) -JContainer3.start -: -Container -started -in -: -267 -ms
```

Once this information is displayed on the screen, the container is ready to receive client calls.

4.2.4.4. Running the Client

Once the container has been started, the client can be launched.

The client is run with the following ant command: **ant run.client**

If the client runs successfully, the following output is displayed:

```
run.client:
 - - - - -[java] -May -16, -2007 -3:39:08 -PM -org.objectweb.carol.util.configuration.ConfigurationRepository -in
 - - - - -[java] -INFO: -No -protocols -were -defined -for -property -'carol.protocols', -trying -with -default -
 - - - - -[java] -May -16, -2007 -3:39:08 -PM -org.objectweb.util.monolog.wrapper.javaLog.Logger -log
 - - - - -[java] -INFO: -Debug.initialize() -- -a3debug.cfg
 - - - - -[java] -May -16, -2007 -3:39:09 -PM -org.objectweb.util.monolog.wrapper.javaLog.Logger -log
 - - - - -[java] -INFO: -ReliableTcpConnection.windowSize=100
 - - - - -[java] -Message -[ID:0.0.1026c2m1, -text:Message_0] -sent
 - - - - -[java] -Message -[ID:0.0.1026c2m2, -text:Message_1] -sent
 - - - - -[java] -Message -[ID:0.0.1026c2m3, -text:Message_2] -sent
 - - - - -[java] -Message -[ID:0.0.1026c2m4, -text:Message_3] -sent
 - - - - -[java] -Message -[ID:0.0.1026c2m5, -text:Message_4] -sent
```

And on the server side, the messages have been received:

```
Receiving -a -message -named -'((org.objectweb.joram.client.jms.TextMessage@4391f0,messageID=ID:0.0.1026c2m1,dest
Receiving -a -message -named -'((org.objectweb.joram.client.jms.TextMessage@13e9934,messageID=ID:0.0.1026c2m4,des
Receiving -a -message -named -'((org.objectweb.joram.client.jms.TextMessage@1e064c,messageID=ID:0.0.1026c2m5,dest
Receiving -a -message -named -'((org.objectweb.joram.client.jms.TextMessage@95ef17,messageID=ID:0.0.1026c2m2,dest
Receiving -a -message -named -'((org.objectweb.joram.client.jms.TextMessage@17c4779,messageID=ID:0.0.1026c2m3,des
```

4.2.5. Timer example

The `build.xml` file for this example is located in the `examples/timerservice` folder.

4.2.5.1. Description

This example shows the use of the @Timeout annotation on a method. The client invokes the TimerBean that will launch a timer. This timer will send a message to an MDB and then calls another bean which implements `javax.ejb.TimedObject` interface.

4.2.5.2. Running the server

If the server is not available, it must be run following the steps described in Chapter 3, "Running the EasyBeans Server" of the developer's guide.

4.2.5.3. Deploying the Bean

The timer bean example must be deployed. It is done automatically if the bean has been installed in the `easybeans-deploy` folder.

On the server side, the following output should display:

```
 - - - - -[java] -9/29/
07 -3:52:50 -PM -(I) -AbsDeployer.deployEJB -: -Deploying -EJB3DeployableImpl[archive=easybeans-
deploy/timer.jar]
 - - - - -[java] -9/29/
07 -3:52:50 -PM -(I) -JContainer3.start -: -Container -started -in -: -104 -ms
```

Once this information is displayed on the screen, the container is ready to receive client calls.

4.2.5.4. Running the Client

Once the container has been started, the client can be launched.

The client is run with the following ant command: **ant run.client**

If the client runs successfully, the following output is displayed on the client side:

```
run.client:
 - - - - -[java] -Sep -29, -2007 -4:16:45 -PM -org.objectweb.carol.util.configuration.ConfigurationRepository -in
 - - - - -[java] -INFO: -No -protocols -were -defined -for -property -'carol.protocols', -trying -with -default -
 - - - - -[java] -Calling -init -method -that -will -fire -a -new -timer...
```

The following output is displayed on the server side:

```
 - - - - -[java] - -SLSB --> -Timer -method -called -by -the -Timer -Service.
 - - - - -[java] - -SLSB --
> -Timer -received -= -'org.ow2.easybeans.component.quartz.EasyBeansTimer@6e7d3050'.
 - - - - -[java] - -SLSB --
> -Info -object -inside -the -timer -object -is -'Simple -Serializable -object'.
 - - - - -[java] - -SLSB --> -Sending -a -message -to -a -MDB -which -will -start -a -timer.
 - - - - -[java] - -SLSB --> -Message -sent
 - - - - -[java] - -SLSB --> -Call -a -local -bean -in -order -to -start -a -new -timer.
 - - - - -[java] - - -MDB --> -Timer -method -called -by -the -Timer -Service.
 - - - - -[java] - - -MDB --
> -Timer -received -= -'org.ow2.easybeans.component.quartz.EasyBeansTimer@59d794d'.
 - - - - -[java] - - -MDB --
> -Info -object -inside -the -timer -object -is -'Timer -started -by -the -onMessage() -method'.
 - - - - -[java] - -TimedBean --
> -Got -a -timer -with -value -'org.ow2.easybeans.component.quartz.EasyBeansTimer@2dd5b883'.
```

4.2.6. Security example

The `build.xml` file for this example is located in the `examples/security` folder.

4.2.6.1. Description

This example illustrates the use of different Java EE 5 annotations which are linked to the security part.

The annotations used by the example are:

- `@DeclareRoles`, which is used to declare the roles used by an EJB component
- `@RolesAllowed`, which lists the authorized roles in order to call a method
- `@DenyAll`, which denies the call to the method (for every role)

- `@RunAs`, which sets a new identity when calling other EJBs

4.2.6.2. Running the Server

If the server is not available, it must be run following the steps described in Chapter 3, "Running the EasyBeans Server" of the developer's guide.

4.2.6.3. Deploying the Bean

The security bean example must be deployed. It is done automatically if the bean has been installed in the `easybeans-deploy` folder.

On the server side, the following output should display:

```
 - - - - -[java] -5/16/
07 -10:59:37 -AM -(I) -AbsDeployer.deployEJB -: -Deploying -EJB3DeployableImpl[archive=easybeans-
deploy/security.jar]
 - - - - -[java] -5/16/
07 -10:59:37 -AM -(I) -JContainer3.start -: -Container -started -in -: -115 -ms
```

Once this information is displayed on the screen, the container is ready to receive client calls.

4.2.6.4. Running the Client

Once the container has been started, the client can be launched.

The client is run with the following ant command: **ant run.client**

If the client runs successfully, the following output is displayed on the client side:

```
 - - - - -run.client:
 - - - - -[java] -Oct -16, -2006 -5:27:03 -PM -org.objectweb.carol.util.configuration.ConfigurationRepository -in
 - - - - -[java] -INFO: -No -protocols -were -defined -for -property -'carol.protocols', -trying -with -default -
 - - - - -[java] -Calling -methods -that -everybody -can -call...
 - - - - -[java] -Call -a -bean -with -run-as -in -order -to -have -'admin' -role...
 - - - - -[java] -Access -denied -as -expected -(method -is -denied)
```

The following output is displayed on the server side:

```
 - - - - -[java] -someRolesAllowed() -called
 - - - - -[java] --> -Caller -is -'Principal[EasyBeans/Anonymous]'.
 - - - - -[java] -for -run-as -bean, -caller -is -Caller -is -'Principal[EasyBeans/Anonymous]
 - - - - -[java] -onlyAdminAllowed() -called
 - - - - -[java] --> -Caller -is -'Principal[admin]'.
 - - - - -[java] -someRolesAllowed() -called
 - - - - -[java] --> -Caller -is -'Principal[admin]'.
```

4.2.7. Pool example

The `build.xml` file for this example is located in the `examples/`pool folder.

4.2.7.1. Description

This example illustrates the definition of some values to limit the size of a pool. In the example, the pool size can be configured through the specific XML deployment descriptor or with annotations.

The example contains two kind of beans, Stateless beans and Message Driven beans.

The annotation used in the example is:

- `@Pool`, for configuring the pool.

By using annotation to configure the pool, the @Pool annotation needs to be put on the class of the bean. For example : @Pool(max = MyInterface.MAX_INSTANCE)

By using XML configuration, the settings are located in the META-INF/easybeans.xml entry of the EJB-JAR file.

```
- - -...
- - - - - -<!-- -Configure -pool -element -with -pool -namespace --->
- - - - - -<pool:pool>
- - - - - - - -<!-- -Sets -the -max -value -to -2 --->
- - - - - - - -<pool:max>2</pool:max>
- - - - - -</pool:pool>
- -... - -
```

4.2.7.2. Running the Server

If the server is not available, it must be run following the steps described in Chapter 3, "Running the EasyBeans Server" of the developer's guide.

4.2.7.3. Deploying the Bean

The pool bean example must be deployed. It is done automatically if the bean has been installed in the `easybeans-deploy` folder.

On the server side, the following output should display:

```
3/7/
08 -5:26:26 -PM -(I) -AbsDeployer.deployEJB -: -Deploying -EJB3DeployableImpl[archive=easybeans-
deploy/pool.jar]
3/7/08 -5:26:28 -PM -(I) -JContainer3.start -: -Container -'easybeans-deploy/
pool.jar' -[2 -SLSB, -0 -SFSB, -2 -MDB] -started -in -1,388 -ms
```

Once this information is displayed on the screen, the container is ready to receive client calls.

4.2.7.4. Running the Client

Once the container has been started, the client can be launched.

The client is run with the following ant command: **ant run.client**

If the client runs successfully, the following output is displayed on the client side:

```
run.client:
 - - - - -[java] -Mar -7, -2008 -5:31:35 -PM -org.objectweb.carol.util.configuration.ConfigurationRepository -ini
 - - - - -[java] -Calling -bean's -methods...
 - - - - -[java] -Waiting -some -time -before -checking -the -number -of -instances...
 - - - - -[java] -Number -of -instances -Annotation -Bean -= -5
 - - - - -[java] - - ---> -This -value -is -OK, -pool -is -limited -to -5
 - - - - -[java] -Number -of -instances -XML -Bean -= -2
 - - - - -[java] - - ---> -This -value -is -OK, -pool -is -limited -to -2
 - - - - -[java] -3/7/08 -5:31:41 -PM -(I) -Logger.log -: -Debug.initialize() -- -a3debug.cfg
 - - - - -[java] -3/7/08 -5:31:42 -PM -(I) -Logger.log -: -ReliableTcpConnection.windowSize=100
 - - - - -[java] -Sending -messages -with -multiple -threads...
 - - - - -[java] -Waiting -some -time -to -ensure -that -all -messages -have -been -sent...
 - - - - -[java] -Look -at -the -server -side -console -to -check -pool -values -of -MDB -...
```

The following output is displayed on the server side:

```
 - - - - -[java] -MDBAnnotationBean: -Number -of -instances -= -'5', -max -= -'5'.
 - - - - -[java] -MDBAnnotationBean: -Number -of -instances -= -'5', -max -= -'5'.
 - - - - -[java] -MDBAnnotationBean: -Number -of -instances -= -'5', -max -= -'5'.
 - - - - -[java] -MDBXMLBean:Number -of -instances -= -'2', -max -= -'2'.
 - - - - -[java] -MDBAnnotationBean: -Number -of -instances -= -'5', -max -= -'5'.
 - - - - -[java] -MDBAnnotationBean: -Number -of -instances -= -'5', -max -= -'5'.
 - - - - -[java] -MDBXMLBean:Number -of -instances -= -'2', -max -= -'2'.
 - - - - -[java] -MDBAnnotationBean: -Number -of -instances -= -'5', -max -= -'5'.

...
```

The instances are not exceeding the limits fixed in the example then everything is working fine.

4.2.8. Migration EJB 2.1/3.0 example

The `build.xml` file for this example is located in the `examples/migrationejb21` folder.

4.2.8.1. Description

This example illustrates the use of annotations that provide Home and Remote interface for clients written for the EJB 2.1 specification.

The annotations used by the example are:

- `@Remote`, for the definition of the business interface.
- `@RemoteHome`, for defining the EJB 2.1 Remote Home interface.
- @LocalHome, for defining the EJB 2.1 Local Home interface.

An EJB that is using these annotations can be used by an EJB3 client and a EJB 2.1 client. These annotations can be used to do a migration of your beans on the server side while the clients are the same.

4.2.8.2. Running the Server

If the server is not available, it must be run following the steps described in Chapter 3, "Running the EasyBeans Server" of the developer's guide.

4.2.8.3. Deploying the Bean

The migration bean example must be deployed. It is done automatically if the bean has been installed in the `easybeans-deploy` folder.

On the server side, the following output should display:

```
5/16/
07 -2:42:24 -PM -(I) -AbsDeployer.deployEJB -: -Deploying -EJB3DeployableImpl[archive=easybeans-
deploy/migration21.jar]
5/16/07 -2:42:25 -PM -(I) -JContainer3.start -: -Container -started -in -: -166 -ms -
```

Once this information is displayed on the screen, the container is ready to receive client calls.

4.2.8.4. Running the Client

Once the container has been started, the client can be launched.

The client is run with the following ant command: **ant run.client**

If the client runs successfully, the following output is displayed on the client side:

```
run.client:
 - - - - -[java] -May -16, -2007 -2:43:18 -PM -org.objectweb.carol.util.configuration.ConfigurationRepository -in
 - - - - -[java] -INFO: -No -protocols -were -defined -for -property -'carol.protocols', -trying -with -default -
 - - - - -[java] -Calling -hello() -method -on -EJB -3.0 -view -of -the -Bean...
 - - - - -[java] -Calling -hello() -method -on -Remote -EJB -2.1 -view -of -the -Bean...
```

The following output is displayed on the server side:

```
Hello -world -EJB -3.0 -!
Hello -world -EJB -2.1 -Remote -View -!
Link -to -itself -remote -= -org.objectweb.easybeans.examples.migrationejb21.EJB2And3Bean_org.objectweb.easybeans
8414877
Link -to -itself -local -view -= -org.objectweb.easybeans.examples.migrationejb21.EJB2And3Bean_org.objectweb.easy
8414877
Calling -itself -on -the -local -view...
Hello -world -EJB -2.1 -Local -View -!
```

4.2.9. EAR example

The `build.xml` file for this example is located in the `examples/ear` folder.

Note

This example required the use of a web container, then it can work in EasyBeans/JOnAS, EasyBeans/Tomcat or EasyBeans/Jetty but not in standalone mode as the war file can't be deployed.

4.2.9.1. Description

This example will deploy the EJB3 included in the EAR file in EasyBeans EJB3 container while the .war file will be deployed in the web container This EAR example includes an EJB3 and a WAR file. This allows to use local interface between the WEB layer and the EJB layer. The EAR file has no entry named `META-INF/application.xml`, EasyBeans will detect the type of the given archives and use default values for the name of the web context. Due to the use of local interface, the Entities don't need to implement the Serializable interface. The interface is not annotated with `@Local` annotation as it is the default value. Each entity class provides a `@NamedQuery` query that allows to get all the objects There is a relationship between `Author` and `Book` entities. It is very simple: One `Author` can write several books, but a `Book` is written only by one `Author`. `@OneToMany` and `@ManyToOne` annotations are used to define the relationship

4.2.9.2. Running the Server

If the server is not available, it must be run following the steps described in Chapter 3, "Running the EasyBeans Server" of the developer's guide.

4.2.9.3. Deploying the EAR

The EAR application example must be deployed. It is done automatically if the EAR has been installed in the `easybeans-deploy` folder.

When the EAR is detected by EasyBeans, the following traces will be displayed :

```
JOnASDeployer.deployEAR -: -Deploying -EARDeployableImpl[archive=/tmp/EasyBeans-Deployer-
benoitf/EAR/ear3.ear]
ENCManager.getInterceptorClass -: -Detecting -JOnAS: -using -JOnAS -ENC -for -the -naming.
JPersistenceUnitInfoHelper.loadDefaultValues -: -Default -persistence -provider -set -to -value -org.hibernate.ej
...
Version.&lt;clinit&gt; -: -Hibernate -Annotations -3.3.0.GA
Environment.&lt;clinit&gt; -: -Hibernate -3.2.4
...
JContainer3.start -: -Container -started -in -: -5619 -ms
AbsJWebContainerServiceImpl.registerWar -: -War -/tmp/EasyBeans-Deployer-benoitf/EAR/ear3.ear/
ear-web.war -available -at -the -context -/ear-web.
JOnASDeployer.deployEAR -: -'EARDeployableImpl[archive=/tmp/EasyBeans-Deployer-benoitf/EAR/
ear3.ear]' -EAR -Deployable -is -now -deployed
```

Once this information is displayed on the screen, the application can be used by using an HTTP browser.

4.2.9.4. Using the Client

Once the container has been started, the client can be accessed.

The URL used to connect to the client is the following: http://localhost:9000/ear-web for JOnAS.

The following text should be displayed on the browser:

```
Initialize -authors -and -their -books...

Get -authors

List -of -books -with -author -'Honore -de -Balzac' -:

 - - - -* -Title -'Le -Pere -Goriot'.
 - - - -* -Title -'Les -Chouans'.
```

```
List -of -books -with -author -'Victor -Hugo' -:

 - - - -* -Title -'Les -Miserables'.
 - - - -* -Title -'Notre-Dame -de -Paris'.
```

There is no output on the server side.

Chapter 5. EasyBeans Code Convention

Contributions should follow the EasyBeans code convention. A good document to begin with is Java code convention [http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html]. Other conventions are also listed in this document.

In addition, EasyBeans uses tools to check the compliance: the checkstyle plugin [http://checkstyle.sourceforge.net/] and the eclipse checkstyle plugin [http://eclipse-cs.sourceforge.net/]. The configuration settings are available in the Eclipse settings of the project.

5.1. File Organization

5.1.1. Header

All files should have a header that contains the LGPL and the date.

If a file is modified, the modification year should be appended to the existing year, which is the year it was initially created. For example, if the create date is '1999' or '2004' it should be edited to '1999-2006' or '2004-2006', respectively.

Also, the tag $Id: code_convention.xml 314 2006-04-04 09:39:43Z pinheirg $ should be added. The following is a header example:

```
/**
 -* -EasyBeans
 -* -Copyright -(C) -2008 -Bull -S.A.S.
 -* -Contact: -easybeans@ow2.org
 -*
 -* -This -library -is -free -software; -you -can -redistribute -it -and/or
 -* -modify -it -under -the -terms -of -the -GNU -Lesser -General -Public
 -* -License -as -published -by -the -Free -Software -Foundation; -either
 -* -version -2.1 -of -the -License, -or -(at -your -option) -any -later -version.
 -*
 -* -This -library -is -distributed -in -the -hope -that -it -will -be -useful,
 -* -but -WITHOUT -ANY -WARRANTY; -without -even -the -implied -warranty -of
 -* -MERCHANTABILITY -or -FITNESS -FOR -A -PARTICULAR -PURPOSE. - -See -the -GNU
 -* -Lesser -General -Public -License -for -more -details.
 -*
 -* -You -should -have -received -a -copy -of -the -GNU -Lesser -General -Public
 -* -License -along -with -this -library; -if -not, -write -to -the -Free -Software
 -* -Foundation, -Inc., -51 -Franklin -Street, -Fifth -Floor, -Boston, -MA - -02110-1301, -USA
 -* -USA
 -*
 -* ------------------------------------------------------------------------
 -* -$Id: -code_convention.xml -3272 -2008-05-21 -09:53:35Z -benoitf -$
 -* ------------------------------------------------------------------------
 -*/
```

5.1.2. Imports

Imports should reference a valid class name, instead of using wildcard imports. Wildcard imports are not authorized.

For example, if the interface and class List and ArrayList are used, the imports should not be as follows:

```
import java.util.*;
```

The imports should have each class as follow:

```
- - - - - -import -java.util.List;
- - - - - -import -java.util.ArrayList;
```

The classes should not have an unused import.

Note

The Eclipse IDE provides facilities to do this job. There is the option Organize Imports (**Shift+Ctrl+O**) in the menu Source that correctly inserts the imports and removes the unused imports. However, this option does not work well with 'import static'.

5.1.3. Class and Interface Declarations

The class and interface names should begin with an uppercase letter. Also, each class and interface has an @author tag in the comment. For example:

```
/**
 -* -This -is -an -example -that -shows -a -class/interface -declaration.
 -* -@author -Gisele -Pinheiro -Souza
 -* -@author -Eduardo -Studzinski -Estima -de -Castro
 -*/
 -public -class -ClassExample -implements -InterfaceExample{
 -}
```

5.2. Indentation / WhiteSpace

5.2.1. Indentation

The space character is used instead of the tab character. The number of spaces for an indent is *4 spaces*.

Wrapping a single source line into multiple lines should follow the Java code convention [http://java.sun.com/docs/codeconv/html/CodeConventions.doc3.html#248].

5.2.2. WhiteSpace

Any trailing spaces should be removed. Eclipse provides a plugin that removes the trailing spaces and converts the tab into spaces. The plugin is AnyEdit [http://andrei.gmxhome.de/anyedit/].

Use whitespaces in for() loop, while(), when concatenating strings. One space should be added before the operator and another after the operator. For example, the correct syntax is:

```
for -(int -i -= -0; -i -< -arTest.length; -i++) -{
 - - - -String -strResult -= -"The -element -" -+ -i -+ -" -has -the -value -" -+ -arTest[i];
 -}
```

The following code does not adhere to the convention:

```
 -for -(int -i -= -0; -i< -arTest.length; -i++) -{
 - - - - -String -strResult -= -"The -element -"+ -i+" -has -the -value -"+arTest[i];
 -}
```

5.3. JavaDoc Comments

All methods and attributes (including protected and private) must have a comment. The parameters, the exceptions thrown, and the method return should have a comment in the method comment. For example:

```
/**
 -* -This -is -an -example -that -is -used -in -the -EasyBeans -Code -Convention.
 -*/
private -int -intValue;

/**
 -* -This -is -an -example -method -to -show -a -class -comment.
 -* -@param -a -an -example -of -parameter.
 -* -@param -b -other -example -of -parameter.
```

```
 -* -@return -the -method -result.
 -* -@throws -Exception -the -exception -thrown -by -the -method.
 -*/
public -int -add(final -int -a, -final -int -b) -throws -Exception -{
 - - - -return -a -+ -b;
}
```

5.4. Statements

5.4.1. If/else

Braces must be used in the if/else blocks, even if there is a single statement. To illustrate:

```
if -(true) -{
 - - - -doThis();
}
```

The following is not allowed:

```
 -if -(true)
 - - - - -doThis();
```

The position of the braces should be the same as in the first example. The following format is incorrect:

```
if -(true)
{
 - - - -test1();
 - - - -test2();
}
```

5.4.2. Try/catch

All exceptions require a statement; no silent catching is allowed. For example:

```
try -{
 - - -doThis();
} -catch -(Exception -e) -{
 - - - -// -should -not -occur
}
```

A logger can be used:

```
try -{
 - - - -doThis();
} -catch -(Exception -e) -{
 - - - -logger.logDebug("Exception -while -doing -.....", -e);
}
```

5.4.3. Inline Conditionals

Inline conditionals are not allowed. The following code is incorrect:

```
b -= -isOk() -? -true -: -false;
```

The correct way to write this is as follows:

```
if -(isOk()) -{
 - - - -b -= -true;
} -else -{
 - - - -b -= -false;
}
```

5.4.4. Naming Conventions

5.4.4.1. Static Final Attributes

Declarations are static final, not final static. This is a JLS recommendation.

5.4.4.2. Constants

Constants should be static and final, and should adhere to the following:

```
'^[A-Z][A-Z0-9]*(_[A-Z0-9]+)*$'
```

5.4.4.3. No Magic Numbers, Use Constants

Constants must be used in the code and magic number must be avoided. For example, the following is not allowed:

```
private -int -myAttribute -= -5;
```

The correct format is:

```
/**
 -* -Default -value
 -*/
private -static -final -int -DEFAULT_VALUE -= -5;

/**
 -* -This -attribute -is -initialized -with -the -default -value
 -*/
private -int -myAttribute -= -DEFAULT_VALUE;
```

5.4.4.4. Attribute Name

The attribute name should not have an underscore (_). The _ is valid for constants that are in uppercase.

Use pValue and mValue instead of p_Value and m_Value.

The pattern for attribute name is:

```
'^[a-z][a-zA-Z0-9]*$'
```

Chapter 6. Contributing to EasyBeans

6.1. Mailing Lists

Developers wanting to contribute information about EasyBeans can share their thoughts via the easybeans mailing list.

The steps necessary for subscribing to the list are described at the following url :http://www.objectweb.org/wws/info/easybeans

6.2. Ideas for Contributing

There are many ways to contribute to easybeans. New ideas are also welcome.

The following is a list of some of the ways to make contributions:

- Documentation: Improve or add to the existing documentation, create new chapters, translate, etc.
- Code: Some glue could be added so that EasyBeans could be integrated in other servers.
- Tests: Add new tests to the current test suite.