CDI field producer

Example cdi-produces-field can be browsed at https://github.com/apache/tomee/tree/master/examples/cdi-produces-field

This example shows the usage of the @Produces annotation. @Produces is a CDI mechanism which allows defining a source for injection. This example shows one of two ways of declaring a producer. Instead of a producer method (see CDI-produces-disposes example) a producer field can be used. A producer field can be used instead of a simple getter method. It could be used to inject resources, such as persistence contexts. One caveat to using producer fields over producer methods is that a @Disposes method cannot be used in conjunction with @Produces field.

ConsoleHandler

```
package org.superbiz.cdi.produces.field;

public class ConsoleHandler implements LogHandler {

    private String name;

    public ConsoleHandler(String name) {
        this.name = name;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public void writeLog(String s) {
        System.out.printf("##### Handler: %s, Writing to the console!\n", getName());
    }
}
```

DatabaseHandler

```
package org.superbiz.cdi.produces.field;

public class DatabaseHandler implements LogHandler {

    private String name;

    public DatabaseHandler(String name) {
        this.name = name;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public void writeLog(String s) {
        System.out.printf("##### Handler: %s, Writing to the database!\n", getName());
        // Use connection to write log to database
    }
}
```

FileHandler

```
package org.superbiz.cdi.produces.field;

public class FileHandler implements LogHandler {

    private String name;

    public FileHandler(String name) {
        this.name = name;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public void writeLog(String s) {
        System.out.printf("##### Handler: %s, Writing to the file!\n", getName());
        // Write to log file
    }
}
```

LogFactory

```
package org.superbiz.cdi.produces.field;
```

```
import javax.enterprise.inject.Produces;
```

```
public class LogFactory {
```

```
private int type = 2;
```

```
@Produces
LogHandler handler;
```

```
public LogFactory(){
    handler = getLogHandler();
}
```

```
public LogHandler getLogHandler() {
    switch (type) {
        case 1:
            return new FileHandler("@Produces created FileHandler!");
        case 2:
            return new DatabaseHandler("@Produces created DatabaseHandler!");
        case 3:
        default:
            return new ConsoleHandler("@Produces created ConsoleHandler!");
    }
```

```
    }
}
```

Logger

```
package org.superbiz.cdi.produces.field;

public interface Logger {

    public void log(String s);

    public LogHandler getHandler();
}
```

LoggerImpl

```
package org.superbiz.cdi.produces.field;

import javax.inject.Inject;
import javax.inject.Named;

@Named("logger")
public class LoggerImpl implements Logger {

    @Inject
    private LogHandler handler;

    @Override
    public void log(String s) {
        getHandler().writeLog(s);
    }

    public LogHandler getHandler() {
        return handler;
    }
}
```

LogHandler

```
package org.superbiz.cdi.produces.field;

public interface LogHandler {

    public String getName();

    public void writeLog(String s);
}
```

beans.xml

```
<beans 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/beans_1_0.xsd">
```

```
</beans>
```

LoggerTest

```
package org.superbiz.cdi.produces.field;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import javax.ejb.embeddable.EJBContainer;
import javax.inject.Inject;
import javax.naming.Context;

import static junit.framework.Assert.assertNotNull;
import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

public class LoggerTest {

    @Inject
    Logger logger;

    private Context ctxt;

    @Before
    public void setUp() {
        try {
            ctxt = EJBContainer.createEJBContainer().getContext();
            ctxt.bind("inject", this);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @After
    public void cleanUp() {
        try {
            ctxt.unbind("inject");
```

```
            ctxt.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    @Test
    public void testLogHandler() {
        assertNotNull(logger);
        assertFalse("Handler should not be a ConsoleHandler", logger.getHandler()
instanceof ConsoleHandler);
        assertFalse("Handler should not be a FileHandler", logger.getHandler()
instanceof FileHandler);
        assertTrue("Handler should be a DatabaseHandler", logger.getHandler()
instanceof DatabaseHandler);
        logger.log("##### Testing write\n");
        logger = null;
    }

}
```

Running

```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.superbiz.cdi.produces.field.LoggerTest
INFO -
********************************************************************************
INFO - OpenEJB http://tomee.apache.org/
INFO - Startup: Thu May 10 01:28:19 CDT 2012
INFO - Copyright 1999-2012 (C) Apache OpenEJB Project, All Rights Reserved.
INFO - Version: 7.0.0-SNAPSHOT
INFO - Build date: 20120510
INFO - Build time: 04:06
INFO -
********************************************************************************
INFO - openejb.home = /home/daniel/projects/openejb/source/openejb/examples/cdi-
produces-field
INFO - openejb.base = /home/daniel/projects/openejb/source/openejb/examples/cdi-
produces-field
INFO - Created new singletonService
org.apache.openejb.cdi.ThreadSingletonServiceImpl@a81b1fb
INFO - succeeded in installing singleton service
INFO - Using 'javax.ejb.embeddable.EJBContainer=true'
INFO - Cannot find the configuration file [conf/openejb.xml].  Will attempt to
create one for the beans deployed.
INFO - Configuring Service(id=Default Security Service, type=SecurityService,
```

```
provider-id=Default Security Service)
    INFO - Configuring Service(id=Default Transaction Manager,
type=TransactionManager, provider-id=Default Transaction Manager)
    INFO - Creating TransactionManager(id=Default Transaction Manager)
    INFO - Creating SecurityService(id=Default Security Service)
    INFO - Inspecting classpath for applications: 26 urls. Consider adjusting your
exclude/include.  Current settings: openejb.deployments.classpath.exclude='',
openejb.deployments.classpath.include='.*'
    INFO - Searched 26 classpath urls in 2015 milliseconds.  Average 77 milliseconds
per url.
    INFO - Beginning load: /home/daniel/projects/openejb/source/openejb/examples/cdi-
produces-field/target/classes
    INFO - Configuring enterprise application:
/home/daniel/projects/openejb/source/openejb/examples/cdi-produces-field
    INFO - Auto-deploying ejb cdi-produces-field.Comp: EjbDeployment(deployment-
id=cdi-produces-field.Comp)
    INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
id=Default Managed Container)
    INFO - Auto-creating a container for bean cdi-produces-field.Comp:
Container(type=MANAGED, id=Default Managed Container)
    INFO - Creating Container(id=Default Managed Container)
    INFO - Using directory /tmp for stateful session passivation
    INFO - Enterprise application
"/home/daniel/projects/openejb/source/openejb/examples/cdi-produces-field" loaded.
    INFO - Assembling app: /home/daniel/projects/openejb/source/openejb/examples/cdi-
produces-field
    INFO - ignoreXmlConfiguration == true
    INFO - ignoreXmlConfiguration == true
    INFO - existing thread singleton service in SystemInstance()
org.apache.openejb.cdi.ThreadSingletonServiceImpl@a81b1fb
    INFO - OpenWebBeans Container is starting...
    INFO - Adding OpenWebBeansPlugin : [CdiPlugin]
    INFO - All injection points were validated successfully.
    INFO - OpenWebBeans Container has started, it took [69] ms.
    INFO - Deployed
Application(path=/home/daniel/projects/openejb/source/openejb/examples/cdi-produces-
field)
    ##### Handler: @Produces created DatabaseHandler!, Writing to the database!
    INFO - Undeploying app: /home/daniel/projects/openejb/source/openejb/examples/cdi-
produces-field
    Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.79 sec

    Results :

    Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```