

Using EnvEntries

Example `injection-of-env-entry` can be browsed at <https://github.com/apache/tomee/tree/master/examples/injection-of-env-entry>

The `@Resource` annotation can be used to inject several things including `DataSources`, `Topics`, `Queues`, etc. Most of these are container supplied objects.

It is possible, however, to supply your own values to be injected via an `<env-entry>` in your `ejb-jar.xml` or `web.xml` deployment descriptor. Java EE 6 supported `<env-entry>` types are limited to the following:

- `java.lang.String`
- `java.lang.Integer`
- `java.lang.Short`
- `java.lang.Float`
- `java.lang.Double`
- `java.lang.Byte`
- `java.lang.Character`
- `java.lang.Boolean`
- `java.lang.Class`
- `java.lang.Enum` (any enum)

See also the [Custom Injection](#) example for a TomEE and OpenEJB feature that will let you use more than just the above types as well as declare `<env-entry>` items with a plain properties file.

Using `@Resource` for basic properties

The use of the `@Resource` annotation isn't limited to setters. For example, this annotation could have been used on the corresponding **field** like so:

```
@Resource
```

```
private int maxLineItems;
```

A fuller example might look like this:

```
package org.superbiz.injection.enventry;
```

```
import javax.annotation.Resource;
```

```
import javax.ejb.Singleton;
```

```
import java.util.Date;
```

```
@Singleton
```

```
public class Configuration {
```

```
    @Resource
```

```
    private String color;
```

```
    @Resource
```

```
    private Shape shape;
```

```
    @Resource
```

```
    private Class strategy;
```

```
    @Resource(name = "date")
```

```
    private long date;
```

```
    public String getColor() {
```

```
        return color;
```

```
    }
```

```
    public Shape getShape() {
```

```
        return shape;
```

```
    }
```

```
    public Class getStrategy() {
```

```
        return strategy;
```

```
    }
```

```
    public Date getDate() {
```

```
        return new Date(date);
```

```
    }
```

```
}
```

Here we have an `@Singleton` bean called `Confuration` that has the following properties (`<env-entry>` items)

- String color
- Shape shape
- Class strategy

- long date

Supplying @Resource values for <env-entry> items in ejb-jar.xml

The values for our `color`, `shape`, `strategy` and `date` properties are supplied via `<env-entry>` elements in the `ejb-jar.xml` file or the `web.xml` file like so:

```
<ejb-jar xmlns="http://java.sun.com/xml/ns/javaee" version="3.0" metadata-complete="false">
  <enterprise-beans>
    <session>
      <ejb-name>Configuration</ejb-name>
      <env-entry>
        <env-entry-name>org.superbiz.injection.enventry.Configuration/color</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>orange</env-entry-value>
      </env-entry>
      <env-entry>
        <env-entry-name>org.superbiz.injection.enventry.Configuration/shape</env-entry-name>
        <env-entry-type>org.superbiz.injection.enventry.Shape</env-entry-type>
        <env-entry-value>TRIANGLE</env-entry-value>
      </env-entry>
      <env-entry>
        <env-entry-name>org.superbiz.injection.enventry.Configuration/strategy</env-entry-name>
        <env-entry-type>java.lang.Class</env-entry-type>
        <env-entry-value>org.superbiz.injection.enventry.Widget</env-entry-value>
      </env-entry>
      <env-entry>
        <description>The name was explicitly set in the annotation so the classname prefix isn't required</description>
        <env-entry-name>date</env-entry-name>
        <env-entry-type>java.lang.Long</env-entry-type>
        <env-entry-value>123456789</env-entry-value>
      </env-entry>
    </session>
  </enterprise-beans>
</ejb-jar>
```

Using the @Resource 'name' attribute

Note that `date` was referenced by `name` as:

```
@Resource(name = "date")
private long date;
```

When the `@Resource(name)` is used, you **do** not need to specify the full **class name** of the bean and can **do** it briefly like so:

```
<env-entry>
  <description>The name was explicitly set in the annotation so the classname prefix
  isn't required</description>
  <env-entry-name>date</env-entry-name>
  <env-entry-type>java.lang.Long</env-entry-type>
  <env-entry-value>123456789</env-entry-value>
</env-entry>
```

Conversly, `color` was not referenced by `name`

```
@Resource
private String color;
```

When something is not referenced by `name` in the `@Resource` annotation a default name is created. The format is essentially this:

```
bean.getClass() + "/" + field.getName()
```

So the default `name` of the above `color` property ends up being `org.superbiz.injection.enventry.Configuration/color`. This is the name we must use when we attempt to declare a value for it in xml.

```
<env-entry>
  <env-entry-name>org.superbiz.injection.enventry.Configuration/color</env-entry-
  name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>orange</env-entry-value>
</env-entry>
```

@Resource and Enum (Enumerations)

The `shape` field is actually a custom Java Enum type

```
package org.superbiz.injection.enventry;

public enum Shape {

    CIRCLE,
    TRIANGLE,
    SQUARE
}
```

As of Java EE 6, `java.lang.Enum` types are allowed as `<env-entry>` items. Declaring one in xml is done using the actual enum's class name like so:

```
<env-entry>
  <env-entry-name>org.superbiz.injection.enventry.Configuration/shape</env-entry-name>
  <env-entry-type>org.superbiz.injection.enventry.Shape</env-entry-type>
  <env-entry-value>TRIANGLE</env-entry-value>
</env-entry>
```

Do not use `<env-entry-type>java.lang.Enum</env-entry-type>` or it will not work!

ConfigurationTest

```
package org.superbiz.injection.enventry;

import junit.framework.TestCase;

import javax.ejb.embeddable.EJBContainer;
import javax.naming.Context;
import java.util.Date;

public class ConfigurationTest extends TestCase {

    public void test() throws Exception {
        final Context context = EJBContainer.createEJBContainer().getContext();

        final Configuration configuration = (Configuration) context.lookup(
            "java:global/injection-of-env-entry/Configuration");

        assertEquals("orange", configuration.getColor());

        assertEquals(Shape.TRIANGLE, configuration.getShape());

        assertEquals(Widget.class, configuration.getStrategy());

        assertEquals(new Date(123456789), configuration.getDate());
    }
}
```

Running

```
-----
T E S T S
-----
```

```
Running org.superbiz.injection.enventry.ConfigurationTest
Apache OpenEJB 4.0.0-beta-1    build: 20111002-04:06
http://tomee.apache.org/
```

```
INFO - openejb.home = /Users/dblevins/examples/injection-of-env-entry
INFO - openejb.base = /Users/dblevins/examples/injection-of-env-entry
INFO - Using 'javax.ejb.embeddable.EJBContainer=true'
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 - Found EjbModule in classpath: /Users/dblevins/examples/injection-of-env-
entry/target/classes
INFO - Beginning load: /Users/dblevins/examples/injection-of-env-entry/target/classes
INFO - Configuring enterprise application: /Users/dblevins/examples/injection-of-env-
entry
WARN - Method 'lookup' is not available for 'javax.annotation.Resource'. Probably
using an older Runtime.
INFO - Configuring Service(id=Default Singleton Container, type=Container, provider-
id=Default Singleton Container)
INFO - Auto-creating a container for bean Configuration: Container(type=SINGLETON,
id=Default Singleton Container)
INFO - Configuring Service(id=Default Managed Container, type=Container, provider-
id=Default Managed Container)
INFO - Auto-creating a container for bean
org.superbiz.injection.enventry.ConfigurationTest: Container(type=MANAGED, id=Default
Managed Container)
INFO - Enterprise application "/Users/dblevins/examples/injection-of-env-entry"
loaded.
INFO - Assembling app: /Users/dblevins/examples/injection-of-env-entry
INFO - Jndi(name="java:global/injection-of-env-
entry/Configuration!org.superbiz.injection.enventry.Configuration")
INFO - Jndi(name="java:global/injection-of-env-entry/Configuration")
INFO -
Jndi(name="java:global/EjbModule1355224018/org.superbiz.injection.enventry.Configurati
onTest!org.superbiz.injection.enventry.ConfigurationTest")
INFO -
Jndi(name="java:global/EjbModule1355224018/org.superbiz.injection.enventry.Configurati
onTest")
INFO - Created Ejb(deployment-id=org.superbiz.injection.enventry.ConfigurationTest,
ejb-name=org.superbiz.injection.enventry.ConfigurationTest, container=Default Managed
Container)
INFO - Created Ejb(deployment-id=Configuration, ejb-name=Configuration,
container=Default Singleton Container)
INFO - Started Ejb(deployment-id=org.superbiz.injection.enventry.ConfigurationTest,
ejb-name=org.superbiz.injection.enventry.ConfigurationTest, container=Default Managed
Container)
INFO - Started Ejb(deployment-id=Configuration, ejb-name=Configuration,
container=Default Singleton Container)
INFO - Deployed Application(path=/Users/dblevins/examples/injection-of-env-entry)
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.664 sec
```

Results :

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

