



From EJB 1.0 to EJB 2.0

A Migration Project

Hermod Opstvedt

Chief Architect

Den norske Bank

From EJB 1.0 to EJB 2.0

A Migration Project

- First step

- Understand the specifications.

- It is very important to know what the specifications require of the developer. Otherwise it is easy to fall into a pit (Container specific dependencies).

- Note the changes between the different specs.

- Make a table identifying what has changed from one version to another.

From EJB 1.0 to EJB 2.0

A Migration Project

- Decide if you want to go complete 2.0.
 - You have the option of sticking around with deprecated stuff, but sooner or later you have to deal with it.
- Decide how you want to do the migration work
 - Manually - < 20 EJB's.
 - Using a tool (automated change) - > 20 EJB's.
- Don't even think about doing any functional change during the migration project – It will hurt you!!

From EJB 1.0 to EJB 2.0

A Migration Project

- Build a list of all your EJBs, and order them by complexity and type (Session/Entity).
 - Plan on doing the easiest/least complex ones first.
- Develop a migration plan.
 - Spec by spec, or one step.
 - Identify which beans can go all the way in one move.

From EJB 1.0 to EJB 2.0

A Migration Project

- Specifications

- EJB 1.0 – EJB-1.0.pdf.
- EJB 1.1 – EJB 1_1-spec.pdf.
- EJB 2.0 - EJB-2_0-fr2-spec.pdf.
- EJB 2.1 - EJB-2_1-pfd-spec.pdf (current).

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1

- `java.rmi.RemoteException` has been deprecated in business methods, `ejbCreate`, `ejbPostCreate`, `ejbFind<METHOD>`, `ejbRemove`, and the container-invoked callbacks (*i.e.*, the methods defined in the `EntityBean`, `SessionBean`, and `SessionSynchronization` interfaces)

- Use `javax.ejb.EJBException` instead.

Ex :

```
public void ejbCreate(String key) throws java.rmi.RemoteException
```

Should now be :

```
public void ejbCreate(String key) throws javax.ejb.EJBException
```

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1
 - `javax.jts.UserTransaction` has changed package to `javax.transaction.UserTransaction`.
 - This only means that you have to change your import statements, or if you do it the ugly way, wherever you refer to it.

```
import javax.jts.UserTransaction
```

Should now be :

```
import javax.transaction.UserTransaction
```

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1
 - `EJBContext.getEnvironment()` has been deprecated.
 - 1.1 introduces an environment naming context (JNDI) instead of the 1.0 environment properties (`java.util.Properties`).
 - This means that you have to manually go through this conversion.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1

- `EJBContext.getEnvironment()` has been deprecated.

- If you do not change the code, you have to define your 1.0 properties in an `ejb10-properties` sub-context of the environment naming section of the deployment descriptor (`ejb.xml`). Ex:

```
<env-entry>
  <description>My Applications name</description>
  <env-entry-name>ejb10-properties/appname</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>MegaApp</env-entry-value>
</env-entry>
```

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1

- `java.security.Identity` has been deprecated. It is replaced by `java.security.Principal`.

- This affects the followings methods of the `javax.ejb.EJBContext` interface

- ✓ `getCallerIdentity()` which has been removed and replaced by `getCallerPrincipal()`.
- ✓ `isCallerInRole(Identity identity)` which is changed to `isCallerInRole(String role)`.
- ✓ and any place that you deal with security in your code.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1

- `java.security.Identity` has been deprecated. It is replaced by `java.security.Principal`.

- Ex:

- `Identity identity=mySessionCtx.getCallerIdentity();`

- Should now be :

- `Principal principal= mySessionCtx. getCallerPrincipal();`

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1
 - `ejbCreate` of Container managed persistence Entity beans now allows null as return value.
 - Only requires recompilation of bean.
 - All Finder methods of entity beans must throw `javax.ejb.FinderException`.
 - Add the exception to all finder methods.
 - Ex:

```
public Person findByPrimaryKey(String firstName, string LastName)
    throws EJBException, FinderException;
```

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1
 - Not allowed to use TX_BEAN_MANAGED for Entity Beans.
 - Do not use it in the deployment descriptor.
 - Not allowed to use SessionSynchronization for TX_BEAN_MANAGED Session Beans.
 - Manually go through and remove the implementation.
 - Or change transaction scope.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1
 - Not allowed to implement the `javax.ejb.SessionSynchronization` interface when using the `javax.transaction.UserTransaction` interface.
 - Remove either the transactional handling, or don't implement the `SessionSynchronization` interface.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1
 - `EJBObject.getPrimaryKey()`,
`EJBMetaData.getPrimaryKeyClass()`,
`EJBHome.remove(Object primaryKey,)` and
`isIdentical(Object other)` for Session Beans now
must throw Exception.
 - New interface `HomeHandle`.
 - `EJBHome` has a new method: `getHomeHandle`.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1
 - EJBMetaData has a new method: `isStatelessSession()`.
 - New class `NoSuchEntityException`.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.0 to 1.1
 - `Java.lang.String` is now allowed as a primary key.
 - And last but not least: A completely new Deployment descriptor scheme. It is now an XML file instead of a serialized object tree (.ser file)!
 - The `manifest.mf` file is no longer used for identifying which beans are in a `ejb.jar` file. They are now defined in the deployment descriptor file (`ejb.xml`).

From EJB 1.0 to EJB 2.0

A Migration Project

- Going from 1.0 to 1.1
 - Step 1 – Identify
 - Identify all beans that are hit by the changes.
 - ✓ Search for files containing one of the changes.
 - ✓ Or
 - ✓ Use a tool to identify them.
 - Step 2 – Apply the changes
 - ✓ Manually
 - ✓ And/or
 - ✓ Use a tool (Vendor/Selfmade/other)

From EJB 1.0 to EJB 2.0

A Migration Project

- Going from 1.0 to 1.1
 - Step 3 – Assemble a the new jar file(s).
 - Remember to pay attention to the deployment descriptor with respect to environment properties.
 - Step 4 – Test your changes thoroughly.
 - Use a 1.1 compliant container (ex. Sun reference implementation).
 - Test, test and test again.

From EJB 1.0 to EJB 2.0

A Migration Project

- Going from 1.0 to 1.1
 - After completing first migration you are ready to move on to the next phase.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.1 to 2.0
 - `java.rmi.RemoteException` in business methods, `ejbCreate`, `ejbPostCreate`, `ejbFind<METHOD>`, `ejbRemove`, and the container-invoked callbacks (*i.e.*, the methods defined in the `EntityBean`, `SessionBean`, and `SessionSynchronization` interfaces) is no longer permitted.
 - MUST use `javax.ejb.EJBException` instead.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.1 to 2.0
 - Radical change in Container Managed Persistence (CMP) Entity beans.
 - Entity beans are now declared as abstract. 1.1 beans are still supported. Change to the new way of doing entity beans should be done as a separate phase.
 - Although, because the persistence mechanism is separated from the implementation, the new way of doing things makes CMP Entity beans a whole lot simpler to code.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.1 to 2.0
 - Mapping between logical and physical attributes is now done at deployment time, and hence is now the responsibility of the deployer.
 - EJB Query Language introduced for CMP Entity beans.
 - The old way of doing it is still supported.
 - A conversion into the new way should be done as a step by it's own, to avoid introducing errors.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.1 to 2.0
 - Container managed relationships between Entity beans (CMR)
 - One-to-one, one-to-many, and many-to-many. Included as a part of the abstract persistence schema.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.1 to 2.0

- New methods/constructors added

- `javax.ejb.EJBContext.getEJBLocalHome()`
- `javax.ejb.EJBException(java.lang.String, java.lang.Exception)`
- `javax.ejb.EJBException.getMessage()`
- `javax.ejb.EJBException.printStackTrace()`
- `javax.ejb.EJBException.printStackTrace(java.io.PrintStream)`

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.1 to 2.0
 - New methods/constructors added
 - `javax.ejb.EJBException.printStackTrace(java.io.PrintWriter)`
 - `javax.ejb.EntityContext.getEJBLocalObject()`
 - `javax.ejb.SessionContext.getEJBLocalObject()`

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.1 to 2.0

- New Features added

- Message driven beans.
- Local beans.
 - ✓ Accessed as normal beans instead of going through the network layer as for remote beans.
 - ✓ A lot faster to access.
 - ✓ Means container affinity between beans.

From EJB 1.0 to EJB 2.0

A Migration Project

- Changes from 1.1 to 2.0
 - New Classes added.
 - javax.ejb.AccessLocalException
 - javax.ejb.EJBLocalHome
 - javax.ejb.EJBLocalObject
 - javax.ejb.MessageDrivenBean
 - javax.ejb.MessageDrivenContext
 - javax.ejb.NoSuchObjectLocalException
 - javax.ejb.TransactionRequiredLocalException
 - javax.ejb.TransactionRolledbackLocalException

From EJB 1.0 to EJB 2.0

A Migration Project

- Going from 1.1 to 2.0
 - Step 1 – Identify
 - Identify all beans that are hit by the changes.
 - ✓ Search for files containing one of the changes.
 - ✓ Or
 - ✓ Use a tool to identify them.
 - Step 2 – Apply the changes
 - ✓ Manually
 - ✓ And/or
 - ✓ Use a tool (Vendor/Selfmade/other)

From EJB 1.0 to EJB 2.0

A Migration Project

- Going from 1.1 to 2.0
 - Step 3 – Assemble a the new jar file(s).
 - Step 4 – Test your changes thoroughly.
 - Use a 2.0 compliant container (ex. Sun reference implementation).
 - Test, test and test again.

From EJB 1.0 to EJB 2.0

A Migration Project

- Automating the migration.
 - Ensures repeatability.
 - Saves time (=money).
 - Saves YOU from boring tasks.

- And now how we do it.

From EJB 1.0 to EJB 2.0

A Migration Project

- EJBMigrator – A Web based migration tool.
- Challenge :
 - Should be Web-based.
 - Should be a wizard style application.
 - Java source files do not easily lend themselves to analysis and/or change programmatically.
 - Create a customizable rules engine.
 - Should be able to generate complete 2.0 compliant ejb.jar files.

From EJB 1.0 to EJB 2.0

A Migration Project

- Solution :

- Web based and Wizard style application –

- Application built using Struts/Tiles in a Workflow type manner.

From EJB 1.0 to EJB 2.0

A Migration Project

- Solution :

- Java source files do not easily lend themselves to analysis and/or change programmatically.
 - Obvious solution was to convert them into XML.
 - Needed something that would parse the source files and generate an XML representation of them.
 - After some research, JavaCC seemed to be what would do the trick.

From EJB 1.0 to EJB 2.0

A Migration Project

- Solution :

- Java source files do not easily lend themselves to analysis and/or change programmatically.
 - Further search uncovered a open source project called BeautyJ (<http://beautyj.berlios.de/>). BeautyJ will read a .java file and transform it into an XML representation and back again.
 - BeautyJ needed some rework and additional JavaCC definitions to work as needed (revised jar file is supplied).

From EJB 1.0 to EJB 2.0

A Migration Project

- Java source files do not easily lend themselves to analysis and/or change programmatically.
 - So now we had an XML representation of it.
 - Working with XML directly is cumbersome.
 - Needed to create a Java object hierarchy that mapped the XML structure.
 - JAXB was the obvious choice.
 - Changing code is just a matter of presenting/getting it from a Web page and then calling setXxxx on a Java object.
 - So finally we had what we wanted – a simple way of navigating, analyzing and altering the Java code.

From EJB 1.0 to EJB 2.0

A Migration Project

- Create a customizable rules engine.
 - Should be easy to extend/add/alter rules.
 - The famous factory pattern was chosen.
 - ✓ Factory reads in all available rules from a property file.
 - ❖ RuleName=Rule class
 - ✓ Has a method for returning a list of all available rules by name.
 - ✓ When a rule is needed, you request it by name, and it is created dynamically using `Class.forName`.

From EJB 1.0 to EJB 2.0

A Migration Project

- Create a customizable rules engine.
 - The rules themselves use another factory build in a similar fashion to get the message to display for the outcome of an analysis (messages defined in a property file – supports internationalization).

From EJB 1.0 to EJB 2.0

A Migration Project

- Should be able to generate complete 2.0 compliant `ejb.jar` files.
 - To avoid as much manual work as possible, we wanted to be able to generate as much as possible of the classes that are needed to build a complete `ejb.jar` file using the application.
 - Further research ended up at another opensource project : XDoclet (<http://xdoclet.sourceforge.net>).

From EJB 1.0 to EJB 2.0

A Migration Project

- Should be able to generate complete 2.0 compliant `ejb.jar` files.
 - XDoclet works by Attribute-Oriented Programming.
 - By adding special Javadoc tags to the `Javasource`, it will generate the files that are necessary to complete an `ejb.jar` file.
 - It contains a number of Ant tasks for doing the `generate/build`.

From EJB 1.0 to EJB 2.0

A Migration Project

- Should be able to generate complete 2.0 compliant `ejb.jar` files.
 - It supports several EJB containers including JBoss, BEA WebLogic, IBM WebSphere, Oracle IAS, Orion, Borland, MacroMedia JRun, Jonas, Pramati, Sybase EAServer.
 - It supports some IDEs.

From EJB 1.0 to EJB 2.0

A Migration Project

- Should be able to generate complete 2.0 compliant ejb.jar files.

Xdoclet tags

```
/**
 * @ejb.bean name="Stateful" type="Stateful"
 *
 * @jonas.bean ejb-name="Stateful" jndi-name="StatefulHome"
 */
public abstract class StatefulBean implements javax.ejb.SessionBean {
    private String x;

    /**
     * @ejb.interface-method
     */
    public String foobar() {
        return "Foobar";
    }

    /**
     * @ejb.create-method
     */
    public void ejbCreateWithParam(String x) {
        this.x = x;
    }
}
```

From EJB 1.0 to EJB 2.0

A Migration Project

- EJBMigrator – an example run.

From EJB 1.0 to EJB 2.0

A Migration Project

- Some of the other tools/help available
 - From Sun
 - Sun ONE Migration Tool for Application Servers.
 - ✓ <http://java.sun.com/j2ee/tools/migdocs/index.html>
 - From IBM
 - Some support in tooling (WSAD/VAJ).
 - Several Redbooks.
 - From Weblogic
 - Some support in tooling (DDConverter)
 - <http://edocs.bea.com/wls/docs61/notes/migrate.html#1071232>

From EJB 1.0 to EJB 2.0

A Migration Project

- Questions



From EJB 1.0 to EJB 2.0

A Migration Project

- References and resources :

- J2EE: <http://java.sun.com/j2ee/>