HowTo create J2EE Application using TogetherSoft ControlCenter

27.07.2001, mm

Autor	Michael Maretzke
eMail	michael@maretzke.com

Stand	31.07.2001
Status	preliminary finished released
	x intern x extern
Version	1.0
Druckdatum	17.11.03 22:26
Verwendungszweck	Knowledge Management

1. Overview 3

2. Creation phase 5

2.1. Create Enterprise Java Beans 5

2.1.1 General Beans properties in TCC 6

2.1.2 Entity Beans 7

2.1.3 Session Beans 7

2.2. Create Servlets 8

2.3. Create WWW resources 8

3. Assembly phase 8

3.1. Create EJB Assembler diagram 8

3.1.1 Create diagram 8

3.1.2 Configure diagram 8

3.1.3 Populate diagram 8

3.2. Create Web Application Diagram 9

3.2.1 Create diagram 9

3.2.2 Configure diagram 9

3.2.3 Populate diagram 10

3.3. Create Enterprise Application diagram 10

3.3.1 Create diagram 10

3.3.2 Configure diagram 11

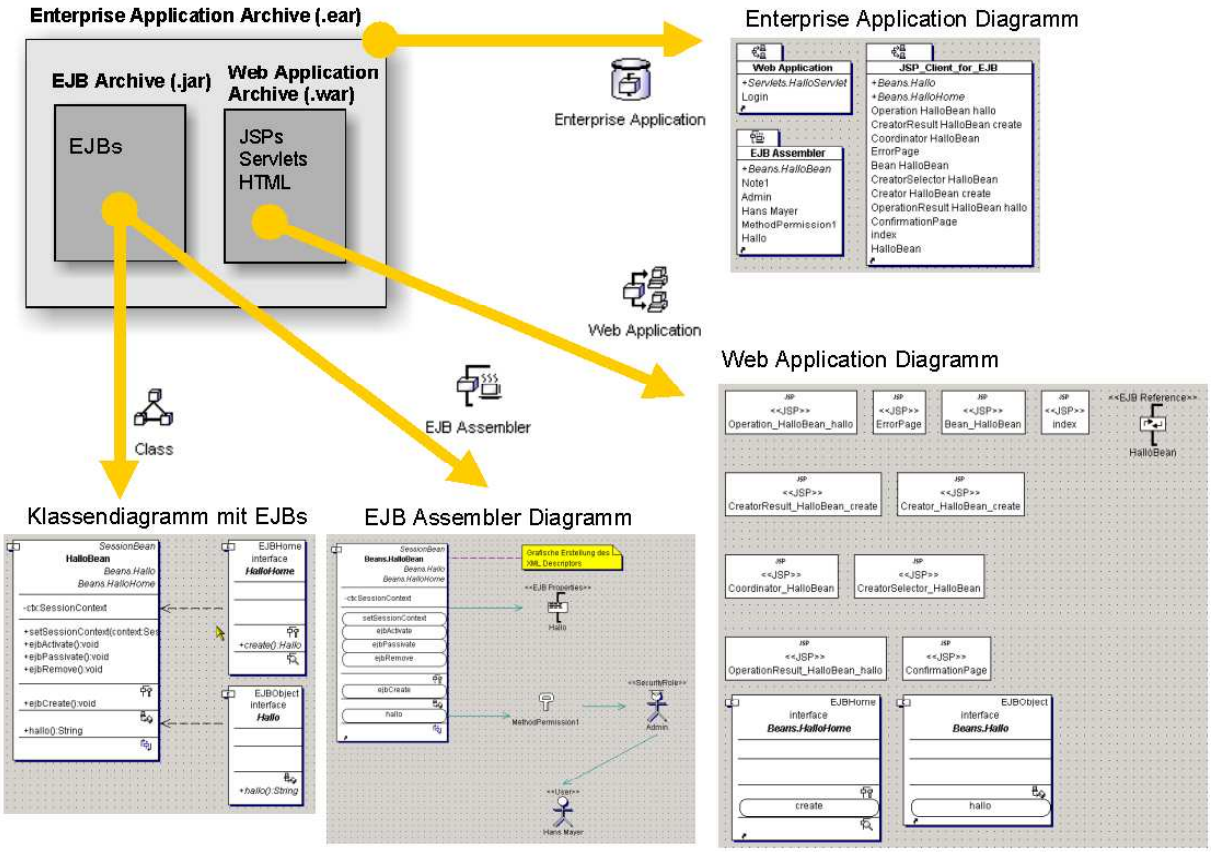
3.3.3 Populate diagram 11

4. EJB Deployment Expert 11

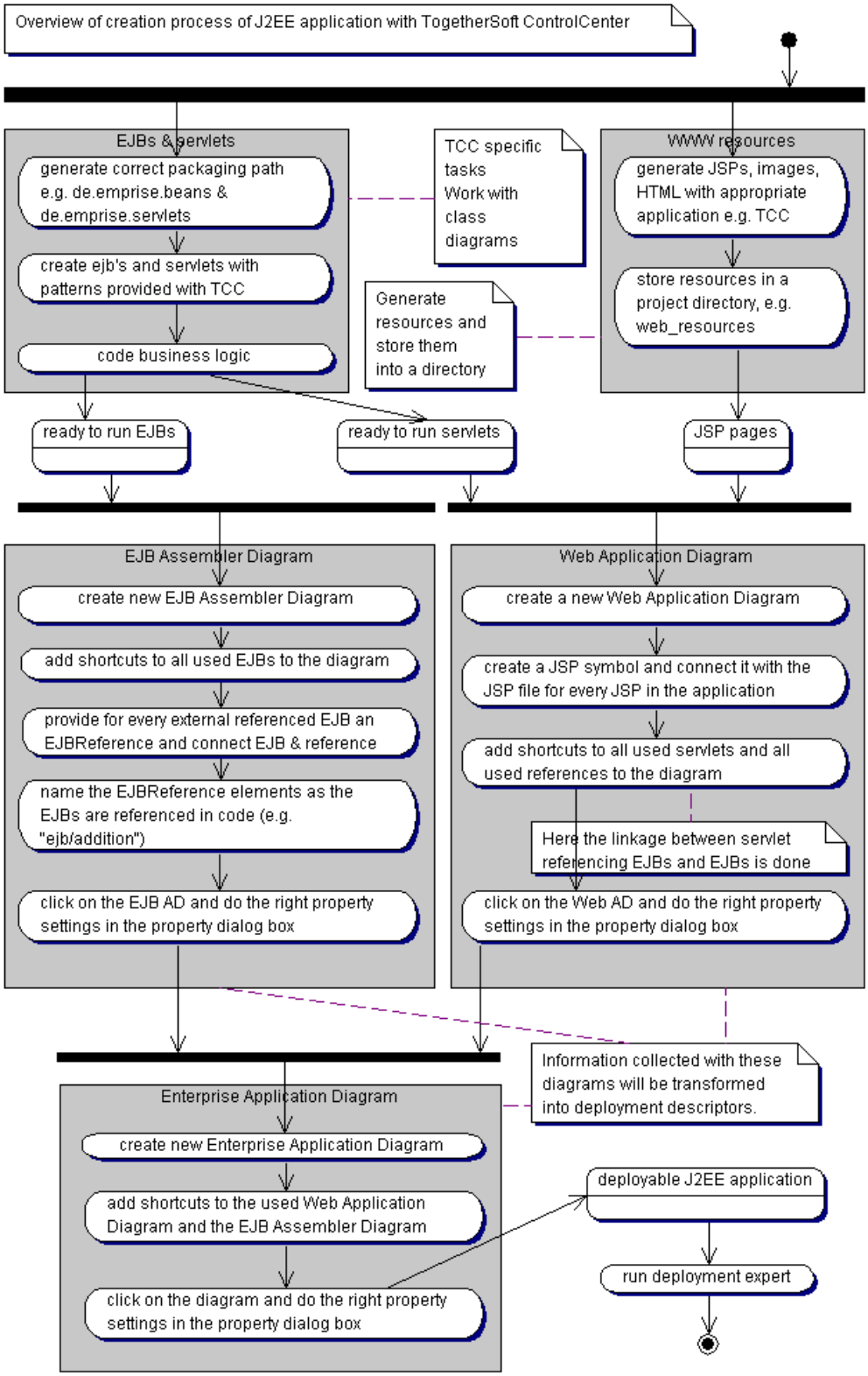
5. Life cycle of J2EE application development 12

6. References 13

1 Overview



Graphic 1 Mapping of J2EE elements to TCC elements [TOS01]



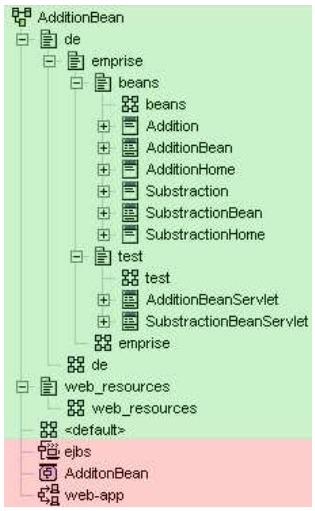
Graphic 2 Overview of creation process

This short HowTo focusses on the deployment process concerning the deployment expert for „generic 1.1“ deployment. Main focus lies on design and deploy process.

The picture in Graphic 1 shows a mapping between TCC elements and J2EE elements. With this mapping in the back of mind some of the following explanations might be clearer.

First of all EJB's and Servlets should be designed and JSP's should be generated. These will be enriched with J2EE needed information concerning deployment descriptors. After providing needed information the application will be plugged together. The last step will be the deployment expert and some specific settings need to be done for deployment on ATG Dynamo.

The complete process can be divided in two phases. The creation phase and the assembly phase. In the creation phase beans and servlets will be designed and coded. The WWW relevant resources will be created. After this phase the application will be plugged together using EJB Assembler diagrams, Web Application diagrams and last Enterprise Application diagrams.



*Graphic 3 Division of process in **creation phase** (green) and **assembly phase** (red)*

Possible products of each phase are shown in Graphic 3.

2 Creation phase

2.1 Create Enterprise Java Beans

With TCC it's no problem to create EJB's. TCC differenciates between

- Entity Beans
- Container Managed Persistenc Beans – CMP
- Bean Managed Persistence Beans – BMP
- Session Beans
- Stateful Session Beans
- Stateless Session Beans
- Message Driven Beans (EJB 2.0 and above)

Construct a bean by clicking the correct symbol above and set it into your class diagram. The class diagram you place the bean should be in the right package hierarchy. If you want to create a bean in package “de.emprise.beans” you have to construct a package  “de” in the default class diagram. Inside “de” construct a package “emprise” and so on.

The next sections will have a closer look on some hints concerning the beans.

1.1.1 General Beans properties in TCC

Selecting an ejb in TCC and typing <ALT> + <ENTER> will show up the property of this bean. Before going in deeper detail let us look on the common properties. The properties dialog is separated into different cards. We concentrate on the card „Session EJB“ and „Entity EJB“.

2.1.1.1 General

The card „General“ is about some general settings. Decide which names to set for home- and remoteinterface, also for primarykey class and implementation class of this bean.

1.1.1.1. Business Methods

Buisness Methods manages the public business methods to use the bean.

1.1.1.2. References1 & References2

„References1“ and „References2“ manages all reference related topics. EJB references used in this bean could be resolved. Also „EJB environment resources“, „EJB resource references“ and „EJB security role references“. Most important seem to be usage of „EJB references“.

Graphic 4 EJB References - an example

Here the name „storage_ejb_ref“ (which could be used in code !) will be preinitialized with the String value „ejb/storage“. This could be used to narrow to the home interface of the referenced ejb.

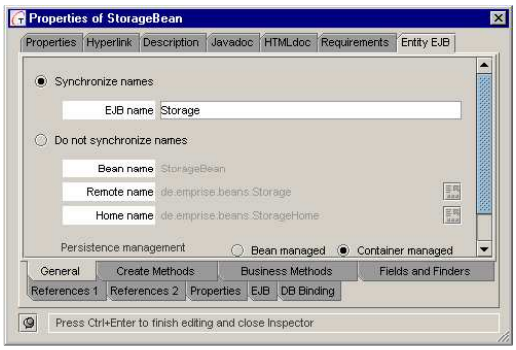
```
private String ejb_ref_prefix = "java:comp/env/";
StorageHome sh;

Context initCtx = new InitialContext();
Object obj = initCtx.lookup(ejb_ref_prefix + storage_ejb_ref);
sh = (StorageHome)PortableRemoteObject.narrow(obj, StorageHome.class);
```

The rest of the columns (Remote Interface, Home Interface, ...) will be set automatically after setting e.g. remote interface with the correct value. The remote interface may be selected via class select box.

1.1.1.3. Properties & EJB

Cards „Properties“ and „EJB“ aren’t so important in this context of deploying applications for ATG Dynamo following the J2EE standard.

2.1.2 Entity Beans



Graphic 5Property Dialog Entity Bean

2.1.2.1 General

Here the decision is made between Container Managed Persistance (CMP) and Bean Managed Persistance (BMP). If CMP is selected some methods needed for BMP will be removed. If „Simple Primary Key“ is selected the need for a primarykey class is obsolete.

1.1.1.4. Create Methods

With the card „Create Methods“ it's possible to manage all `ejbCreate()` methods for this bean.

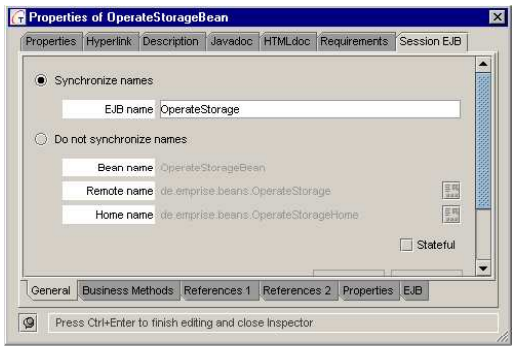
2.1.2.2 Fields and Finder

„Fields and Finder“ manages the fields of the bean and their finder methods. Fields in entity beans will be mapped to get persistent in any way. Finder methods are useful to find persistent data form the persistance store.

1.1.1.5. DB Binding

„DB Binding“ isn't so important in this context of deploying applications for ATG Dynamo following the J2EE standard.

2.1.3 Session Beans



Graphic 6 Property Dialog Session Bean

2.1.3.1 General

Here's the most important decision to take which type of Session Bean will be created. A stateful or a stateless session bean. This decision is made by clicking a check box.

2.2 Create Servlets

Creating servlets is as easy as creating EJB's in TCC. Click on  to create a class by pattern. Choose "Servlet" and make the right settings for name and choose the methods you wanted to be pregenerated.

After this step coding must be done to get the wanted functionality.

2.3 Create WWW resources

WWW resources as HTML, images, JSP's ... will be created using some special applications. Afterwards the products will be copied into project hierarchy (e.g. `/projectdir/web-resources`). They will be linked together in assembly phase.

3 Assembly phase

After the creation phase you should have coded all beans and servlets. Coding phase should be successfully ended for this iteration. Pressing `<SHIFT> + <F7>` in TCC should generate no errors compiling the sources.

Now let's enter the assembly phase. Information about how to construct the J2EE compliant application will be collected in this phase. The information stored graphically and in some property dialogs will be transformed into the deployment descriptors.

3.1 Create EJB Assembler diagram

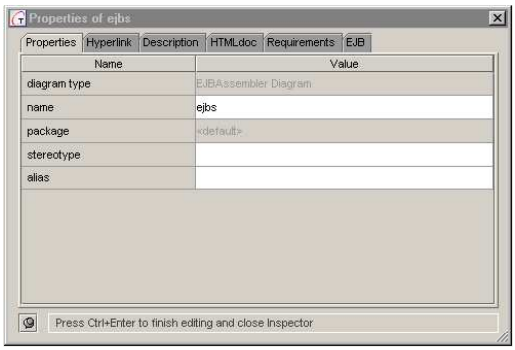
3.1.1 Create diagram

First of all create the EJB Assembler Diagram (AD) which ends in the creation of the EJB-JAR Archive. All business logic will be stored here.

Create an AD by pressing `<CTRL> + <n>` select card „Together" and choose „EJB Assembler Diagram".

3.1.2 Configure diagram

Only the property „name" on card „Properties" is important in this context in the property dialog (`<ALT> + <ENTER>`). This property sets the resulting EJB-JAR filename, e.g. „ejbs".



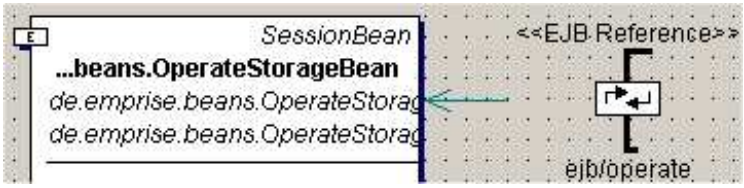
Graphic 7Property Dialog of Assembler Diagram

3.1.3 Populate diagram

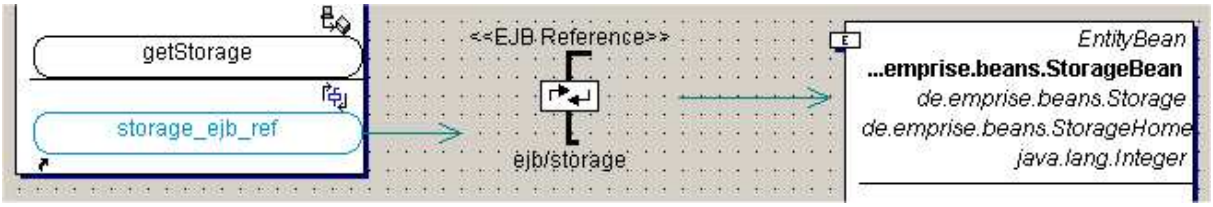
Afterwards add your beans and ejb-references. Right-click on your empty AD sheet and select „Add Shortcuts". Now select all the beans which should be included into your EJB-JAR archive. Shortcuts to the selected beans will be

placed onto your AD. For every bean referenced from somewhere (other ejb's or servlets) add a EJB-Reference ( ) to the AD. Name the ejb-reference as wanted, e.g. "ejb/storage". Connect the reference with the EJB it references using . Start at the reference and end at the EJB.

Here are some examples:



Graphic 8 An EJB Reference named "ejb/operate" pointing to a SessionBean



Graphic 9 An EJB Reference named "ejb/storage" resolving the reference from left bean pointing to right bean

EJB References are used to create `<ejb-ref>` entries in web.xml deployment descriptors.

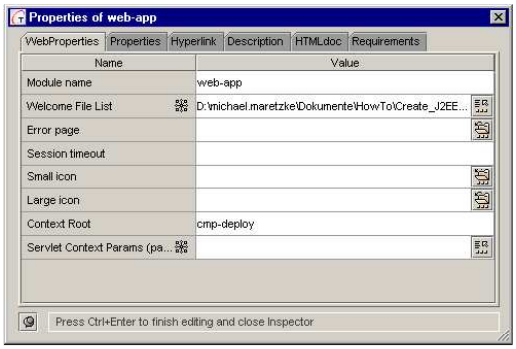
3.2 Create Web Application Diagram

3.2.1 Create diagram

Create a Web Application Diagram (WAD) by pressing `<CTRL> + <n>` select card „Together" and choose „Web Application Diagram".

3.2.2 Configure diagram

After creating the WAD show up the property dialog.



Graphic 10 Property Dialog of Web Application Diagram

Under the card called „WebProperties" some settings are necessary.

3.2.2.1 Module name

This name is the name provided for the WAR file.

3.2.2.2 Welcome File List

The file set in this property will be shown if the J2EE application will be started in a web context. It's very important to set the file to a really existing file – otherwise the entry `<web-file-list>` will not be produced by TCC. The absolute path will be extracted and will not be set in deployment descriptor.

3.2.2.3 Context Root

After the deployment process is complete and the application runs it should be somehow adressed. A kind of URL should be provided for the application. This is done setting context root. If set as shown in Graphic 10 the J2EE application is adressed as follows:

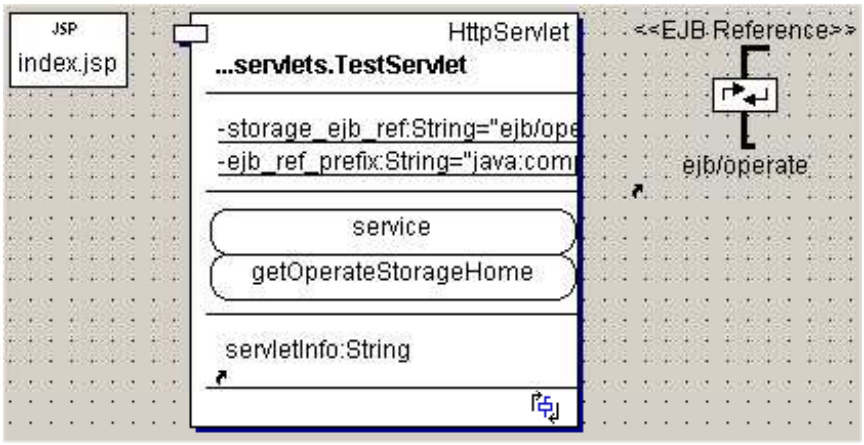
`http://<host>:<port>/cmp-deploy`

Entering this URL into the web browser the J2EE container will deliver the file defined by Welcome File List property.

3.2.3 Populate diagram

Create a  symbol for every JSP page used in your application. Let this symbol be named as the page (e.g. "index.jsp") and let this symbol point to a real existent file. Following this rules your JSP will be packaged into your WAR archive.

Now create shortcuts (right click on the WAD and choose "Add Shortcuts") to the servlets used in your application. Create also shortcuts to your "EJB Reference" elements created in your Assembly Diagram. Import only these references used by your servlets.



Graphic 11 Web Application Diagram

In Graphic 11 the Servlet called „`TestServlet`" references an EJB via „`.../ejb/operate`". Therefor an EJB Reference shortcut for this reference in servlets' code is needed.

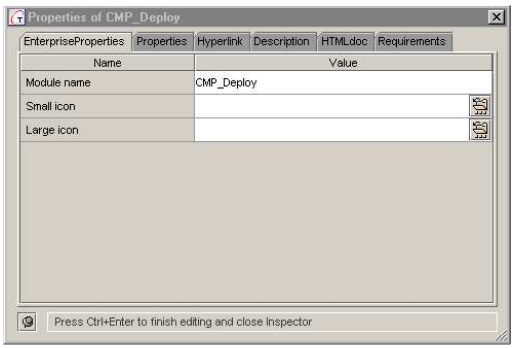
After placing all needed elements the next step is to assemble all subelements to an J2EE compliant application.

3.3 Create Enterprise Application diagram

3.3.1 Create diagram

Last step in assembly phase is the creation of Enterprise Application Diagram (EAD). Create an EAD by pressing `<CTRL> + <n>` select card „Together" and choose „Enterprise Application Diagram".

3.3.2 Configure diagram



Graphic 12 Property Dialog of Enterprise Application Diagram

The only interesting setting to configure is the „Module Name“ which sets the name of EAR-Archive to create.

3.3.3 Populate diagram

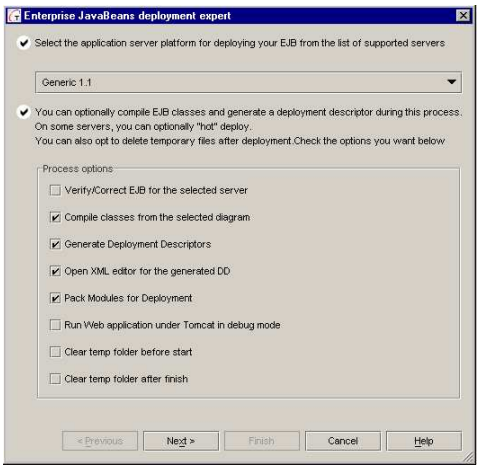
Right click on the empty EAD and select „Add Shortcut“. Select the created Web Application Diagram and the EJB Assembler Diagram.

The J2EE compliant application is ready to deploy.

4 EJB Deployment Expert

To deploy the complete J2EE compliant application it’s necessary to select and show the previously constructed EAD. If only the WAD is selected an shown only the WAR file archive will be generated.

Select „EJB Deployment Expert“ in TCC’s „Tools“ menu.



Graphic 13 First dialog in EJB Deployment Expert

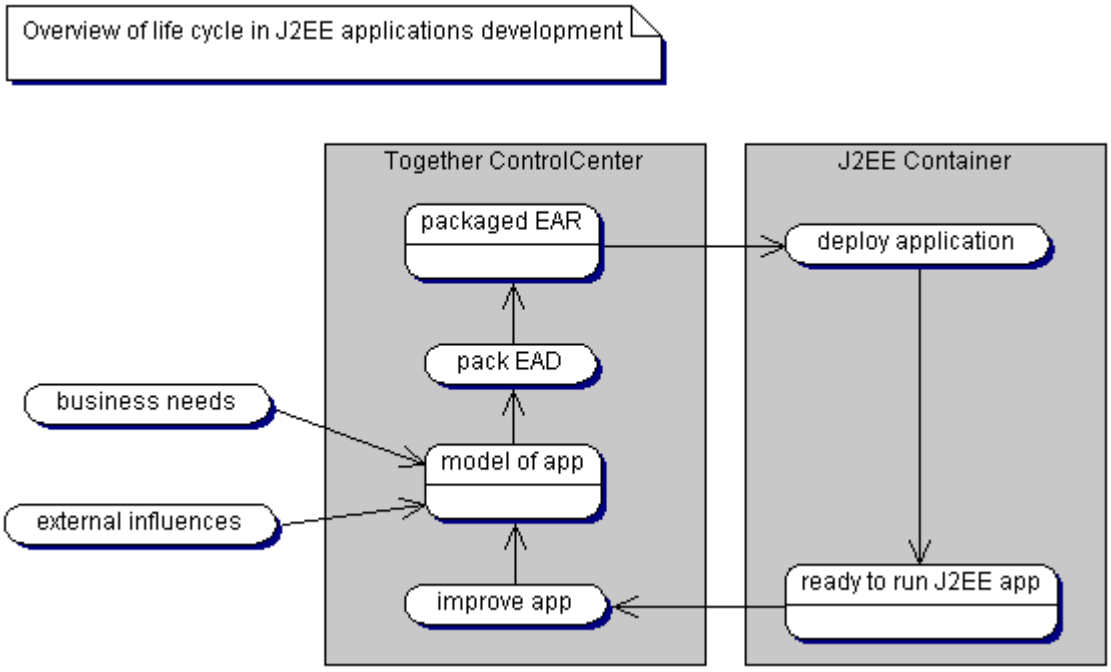
For deploying target ATG Dynamo choose „Generic 1.1“ as application server platform. Select „Compile classes from the selected diagram“ to ensure all of the latest changes will be considered in your EAR archive file. „Generate Deployment Descriptors“ should also be checked. „Open XML editor for the generated DD“ is an option which should also be checked to see if all deployment descriptors are created as wanted. Sometimes a wrong setting somewhere

in a TCC property field could result in a missing entry in one of these descriptors. Searching the error is a time-consuming process. „Pack Modules for Deployment“ is also necessary to get a EAR archive file.

On the next page the target directories and the directories for JDK and J2EE kits from SUN could be set.

During the process some steps (about 40) will be processed. Sometimes interaction will be needed if the generated deployment descriptors should be validated. After the process there's a EAR archive file in the selected target directory. The J2EE compliant application.

2. Life cycle of J2EE application development



Graphic 14 Overview of life cycle in J2EE applications development

Most of these arrows shown in Graphic 14 mean automatic processes. Nice, but some of these arrows stand for time-consuming processes. As shown, development is always a kind of iterative process. The process to get a „packaged EAR“ from the „model of app“ is a time-consuming process. Every time deployment is wanted the expert must step through its number of steps (about 40). Another one is the way from the „packaged EAR“ to the „ready to run J2EE app“. The J2EE container specific deployment tool has to unpack the EAR, compile all needed stubs and skeletons to integrate into the J2EE container and pack it again. Some syntactic checks are done also.

Regarded for its own the time to process is no problem. The sum of all process times make J2EE development a time wasting affair. One little error in one of these deployment descriptors makes a developer to run through the complete cycle. Time to deploy is sometimes much higher than developing the core functionality of software.

5 References

[TOS01] Presentation of from TogetherSoft about „J2EE with Together ControlCenter 5.0“, Oliver Böhm, TogetherSoft GmbH