The Fast Way to Component-Based Development Using SAP NetWeaver CE 7.1 – Part II: Local Development Configurations

Applies to:

SAP NetWeaver Composition Environment 7.1.

Summary

SAP NetWeaver Composition Environment 7.1 (CE), SAP is JEE 5.0 certified. SAP has added several enhancements of concepts and object types to the Java development. These can be used optionally. The enhancements address certain development needs, such as a client-independent UI, the handling of database tables, or the creation of composite applications. Conceptual enhancements ensure an improvement in the structuring of applications and the development process.

The development according to the component model is organized by development configurations, which define the development landscape for a specific software development project. Development configurations can be created locally (instead of using the SAP NetWeaver Development Infrastructure).

This second part of the article focuses on creating and using local development configurations. These processes are described in a “hello world” example. All development steps of the component-based development with optional infrastructure require only an SAP NetWeaver Developer Studio and a runtime system.

If you want to use local development configurations for team development additionally, central processes can be set up with the command line tool delivered with the Developer Studio for building the application centrally and packing it as a software component archive (SCA) for delivery. Optionally you can enhance these processes with external (non-SAP) infrastructure. Team development will be the topic for the third part of this article.

Author: Wolf Hengevoss

Company: SAP

Created on: 10 January 2008

Author Bio

Wolf Hengevoss graduated in natural sciences from the University of Kaiserslautern. In 1999, he joined SAP as a member of the product management. He has worked in the Basis group focusing on topics such as Computer-Aided Test Tool and Business Address Services. Since the early stages of SAP Exchange Infrastructure, he has been working on the Java environment. Today, his focus is on SAP’s Software Change Management of non-ABAP applications.

Table of Contents

Introduction

SAP is – apart from SUN, of course – the first certified vendor of a ***JEE 5.0*** server implementation. In addition to pure standard J2EE/JEE applications, SAP provides you with additional types of Java applications and enhanced control of the development process if you use SAP's component model. Let me recall a few facts from part I of this article:

- **SAP's component model** adds metadata to projects to break down applications into reusable components that contain ***development objects*** like class files, and so on. Those components that have the granularity of Eclipse projects are called ***development components*** (DCs). Explicit declaration of dependencies between DCs allows for better reuse and a build process on the granularity of these (DCs) instead of a build process for a complete application. Development components are grouped into ***software components*** (SCs) for installation and upgrade.

Note: SAP's component model does not change Java/JEE coding. All objects, such as Java classes, interfaces and so on remain unchanged. What is affected is the reuse between projects: In addition to the import you used, you have to declare dependencies to the DCs you want to use. This is only possible for those parts of the used components that have been added to the public part of the component used.

- While all the standard Java and JEE projects can still optionally be used without it in SAP NetWeaver, SAP's component model is used for many of the **SAP-specific types**. It is a prerequisite for *Composite Applications* and also recommended at least for *Web Dynpro* development, for example.
- SAP's concept for team development is what we call ***development configurations***, a description of a development environment written as an XML file and used to control the SAP NetWeaver Developer Studios (SAP's IDE) access to the relevant sources and artefacts. Development configurations are also a prerequisite for component-based development. They can be created centrally in the NWDI or, now, locally. Working with the same development configuration means you start with the same set of objects, which really helps to synchronize development efforts. Development results in a software component archive (SCA), which is the format in which Java applications are usually installed on the SAP NetWeaver platform.

In the first part I have described the development scenarios in the SAP NetWeaver Composition Environment 7.1:

- *Standard Development*: Develop standard JEE applications using the SAP NetWeaver Developer Studio.
- *Component-Based Development with Optional Development Infrastructure*: Development of both standard J2EE/JEE and all SAP-specific object types using the SAP NetWeaver Developer Studio, and, optionally, a tool to handle components in the central build and so on. It is also possible to use a versioning tool or other infrastructure tools.
- *Component-Based Development with Full-Blown SAP NetWeaver DI*: The development infrastructure of the SAP NetWeaver (NWDI) supports all steps from product definition to delivery and maintenance in centrally hosted systems for all types of components.

In this second part of the article I will describe the steps needed in component-based development with a local development configuration. A local development configuration has been an option since the very first version of the Developer Studio, but now there are three major improvements:

- You can create local development configurations yourself that are tailored to fit your needs
- You can create your software components - including their names, required SCs, and vendor information
- You can export both development configuration and software components you created

In the next part I will add information on how to integrate the optional steps in the command line tool for a central build, where you can integrate your source code versioning and how you can deliver and reuse the results of your work.

Prerequisites

There is only a small list of items you need to start with the SAP NetWeaver Composition Environment:

- SAP NetWeaver Developer Studio CE – your Java IDE in the SAP NetWeaver environment. It comes with a tool you can use to set up a simple process centrally to bundle the work of a team of developers.
- SAP NetWeaver Application Server Java – the runtime for your programs.

Note: To set the server according to your needs, you can use templates:
Open the Config Tool under the following path <installation drive>:\usr\sap\<SID>\J00\j2ee\configtool\configtool.bat.
Accept the default DB connection.
In the menu under *File*, choose *Change System Template*.
Choose a template that fits your needs from the list and follow the instructions.

If you are interested in the topic, you can get these items as downloads in a trial version from the SDN download area → (http://sdn.sap.com/ → Downloads → Software/SAP NetWeaver Composition Environment). If it comes to the question of licenses for productive use, check SDN for a new blog on the matter called "SAP NetWeaver, Composition Subscription Now Available", which deals with this question under https://www.sdn.sap.com/irj/sdn/weblogs?blog=/pub/wlg/8066.

In addition to the parts delivered by SAP, you can optionally use development infrastructure tools of your choice, for example, for the file versioning.

Component-Based Development with Optional Development Infrastructure: The Steps

Setting up a local development configuration is the fastest way to component-based development – apart from the runtime, everything you need is provided with the SAP NetWeaver Developer Studio.

In the following I will sketch out the tasks and procedure for developing component-based applications in the SAP NetWeaver CE environment with minimal effort – you can adopt a more comprehensive approach delivered by SAP later when your needs grow. This is what we did to make things easy:

- Integration of all component model related steps into one perspective of the SAP NetWeaver Developer Studio CE called the Development Infrastructure perspective
- Build and packaging tools as part of the Developer Studio

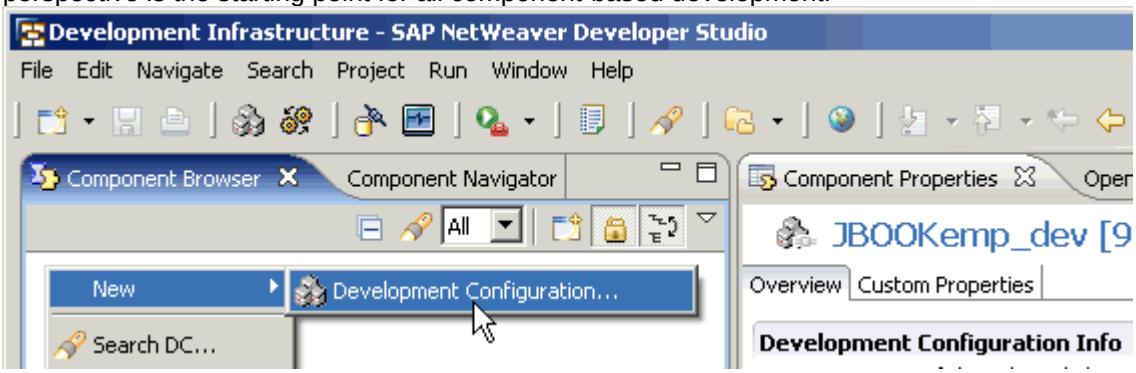
These functions can be accompanied by other tools to support team development (to be described in the next part of the article):

- Command-line tool for build and packaging as part of the Developer Studio installation. ANT tasks for build and packaging are delivered as part of the Developer Studio installation
- Source repository integration (team development support): You can integrate the work of developers in a team using the file system. I will deal with this subject in the next part of the article in detail. Direct integration of source code versioning into the development processes of the SAP NetWeaver Developer Studio is possible using SAP's Design Time Repository, which is delivered with usage type DI of SAP NetWeaver 7.0, and which fully supports development with the SAP NetWeaver CE – please compare part one of this article.

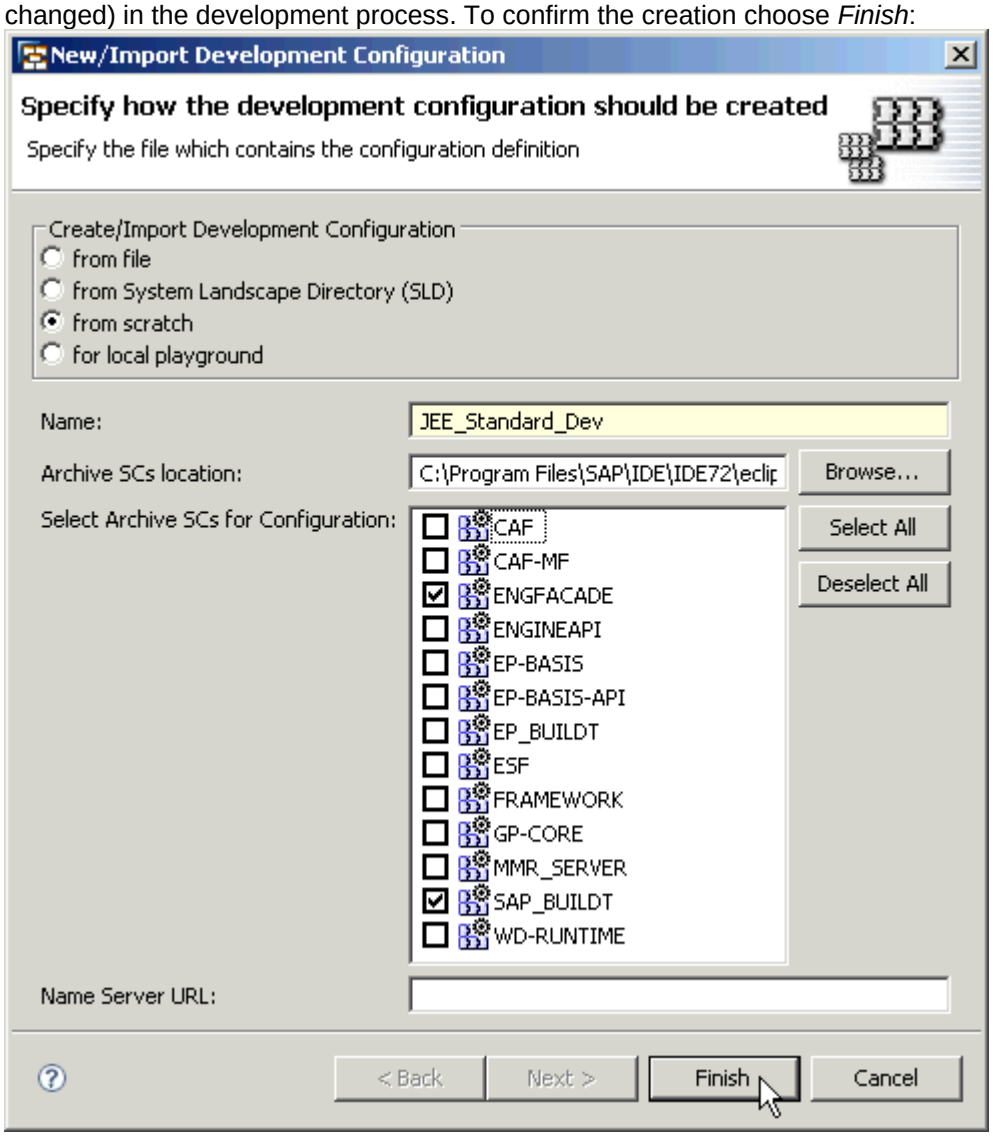
Setting up the Development Environment: Creating a Local Development Configuration

To set up your Developer Studio for component-based development, you create a local development configuration. It provides you with all the prerequisites you need to develop and deliver all project types that can be deployed on SAP NetWeaver – both the standard JEE type or SAP-specific types like Dictionary, Web Dynpro, and so on. The only difference is the selection of predefined libraries you decide to use in each development configuration. A set of such libraries is delivered as software components (SCs) by SAP with your Developer Studio. Each software component contains development components (DCs) in the granularity of Eclipse projects.

1. Open the SAP NetWeaver CE Developer Studio and navigate to the Development Infrastructure Perspective under Window → Open Perspective → Other → Development Infrastructure. This perspective is the starting point for all component-based development:



2. Create a development configuration "*from scratch*"; then select "*required*" SCs to be used (not changed) in the development process. To confirm the creation choose *Finish*:

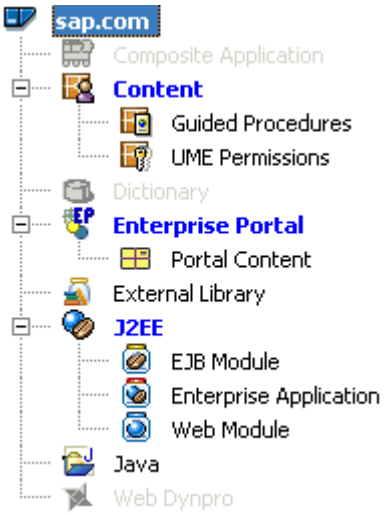
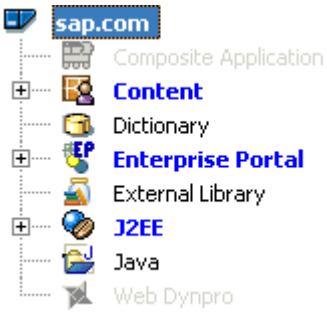
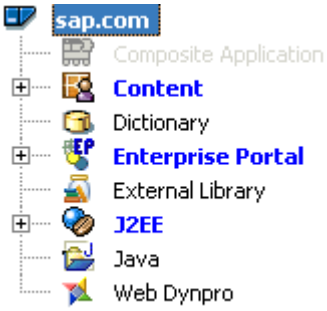


Note: All the "libraries" SAP provides are part of these Archive SCs and are delivered ready to be used with the Developer Studio.

Note: To guarantee the **uniqueness of names** of DCs, packages,and so on, you can optionally assign a Name Service running in SAP's System Landscape Directory (SLD), which is part of the SAP NetWeaver Application Server Java. We will deal with that in a later part of the article.

The object types that you can create in a specific development configuration depend on the required SCs that you chose from the list. Required SCs in dependency of project types you want to create are listed in the following table.

Table of *required SCs* according to different object types you need to create:

DC Project Types to Create	Required Software Component (Release 7.1)	Possible DC Types in the Developer Studio
Standard JEE **+ Enterprise Portal** **+ Guided Procedures**	• SAP_BUILDT • ENGFACADE There is a change of dependencies SAP NetWeaver 7.10 SP3 – update of existing development configurations is required in any case.	
All types listed before **+ Dictionary**	• SAP_BUILDT • ENGFACADE • ENGINEAPI • FRAMEWORK	
All types listed before **+ Web Dynpro**	• ENGFACADE • ENGINEAPI • FRAMEWORK • WD-RUNTIME • SAP_BUILDT	

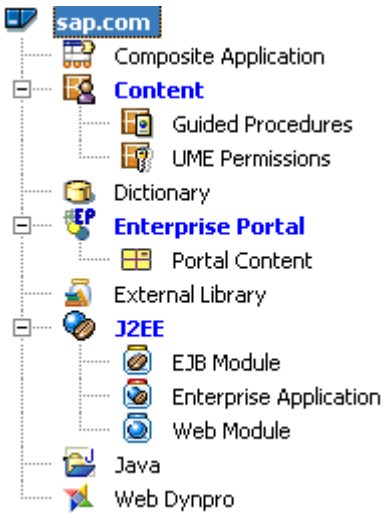
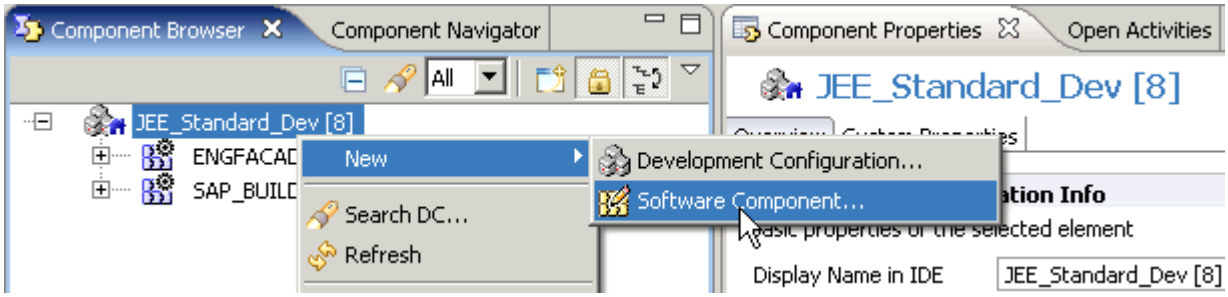
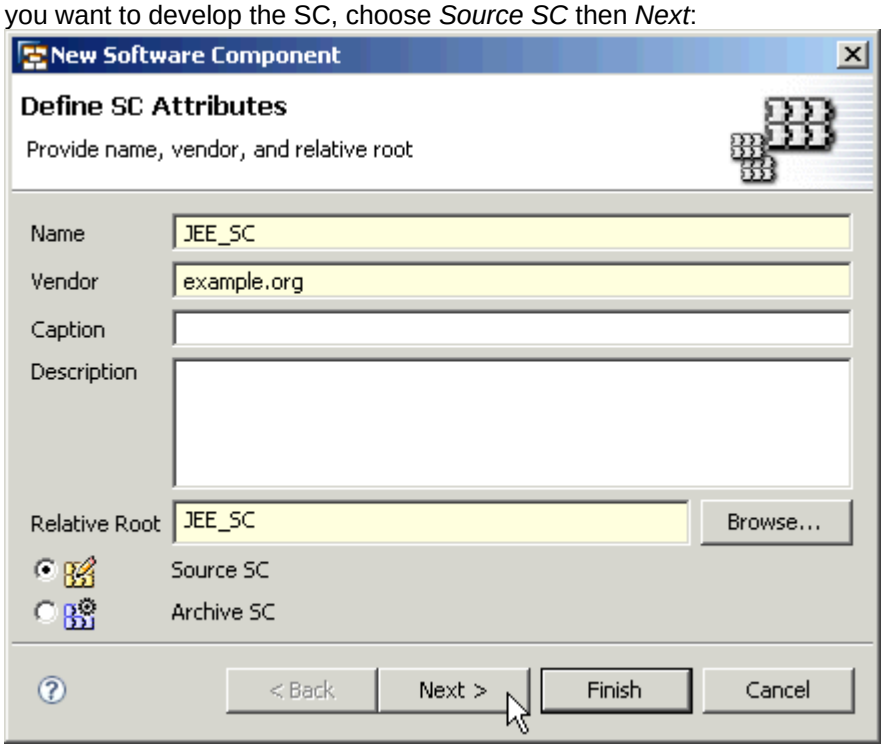
All types listed before **+ Composite Applications**	• CAF • CAF-MF • ENGFACADE • ENGINEAPI • EP-BASIS • EP-BASIS-API • EP_BUILDT • ESF • FRAMEWORK • GP-CORE • MMR_SERVER • SAP_BUILDT • VCFRAMEWORK • WD-RUNTIME	

Table 1: List of required software components derived from types of development projects needed – DC types shown in color are available when selecting the required SCs listed.

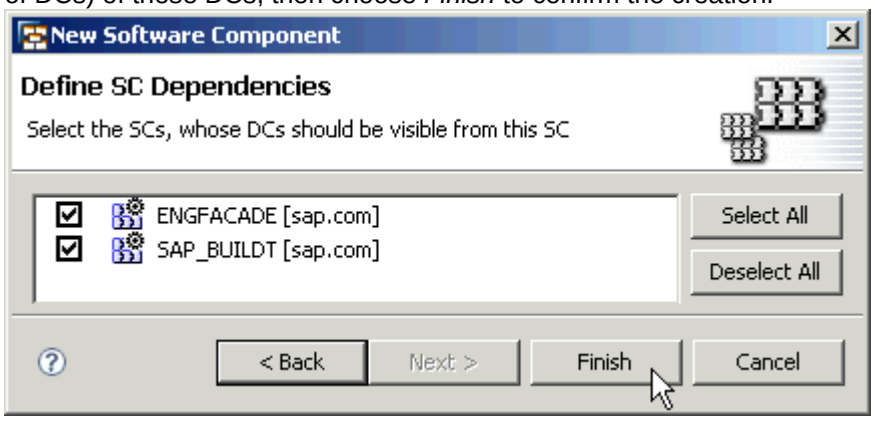
3. Create a new software component (SC), you want to develop and select which of the required SCs the new one is allowed to use:



4. Enter an SC *Name* and the *internet domain* of your company to indicate the SC's *Vendor* or owner; if you want to develop the SC, choose *Source SC* then *Next*:



5. Choose the SCs you have chosen as the required SCs – the new SC you are just creating will then depend on these – in SAP's component model this means it may use all the libraries (runtime objects of DCs) of these DCs, then choose *Finish* to confirm the creation.



Note: If you are only creating one SC for development, then it should use all the SCs you have previously selected. If you are, for example, developing one SC for basic functions and another for the UI, they might be using an overlapping but not identical subset of the required SCs.
In the third part of this article I will describe how to use an SC you developed in another development configuration.

Note: Any *software you create in layers* – a basic layer with business functions and a UI layer, for example – will need the SCs shown in the table above for the basic layer. This is also true for your UI SC, but additionally it will use the basic SC also. Both SCs can be developed in separate development configurations or in the same.

Now the development environment is completely defined for your project. The best thing is perhaps that this configuration can be exported to let other developers – usually your team colleagues – set up their PC in an identical manner with a few mouse clicks. I will describe this later.

Developing a Software Component: Creating DC-Projects

A software component is the object that you use to install new functionality. To develop an SC means that you fill it with projects. In component-based development, these projects are called development components (DCs). DCs are provided for every type of Java-based project, both standard and SAP-specific. What makes a DC different from a non-DC Eclipse-project is the metadata, which additionally contains information on DC public parts (a DC's interface) and dependencies (other DCs required for a DC). Also, for every type of DC (Java, Enterprise Application, Web Dynpro, and so on) the correct build script has to be used. Therefore, these parameters define how an application is built.

Note: If a DC used by another has been changed and built, any dependent DC has to be rebuilt as well. In this local scenario you have to take care of that: DC needing rebuild are marked with a "*" in the Component Browser. If you are using SAP's Component Build Service (CBS) by reading the metadata of all DCs the CBS can take care of that – we'll deal with that in a later part of the article

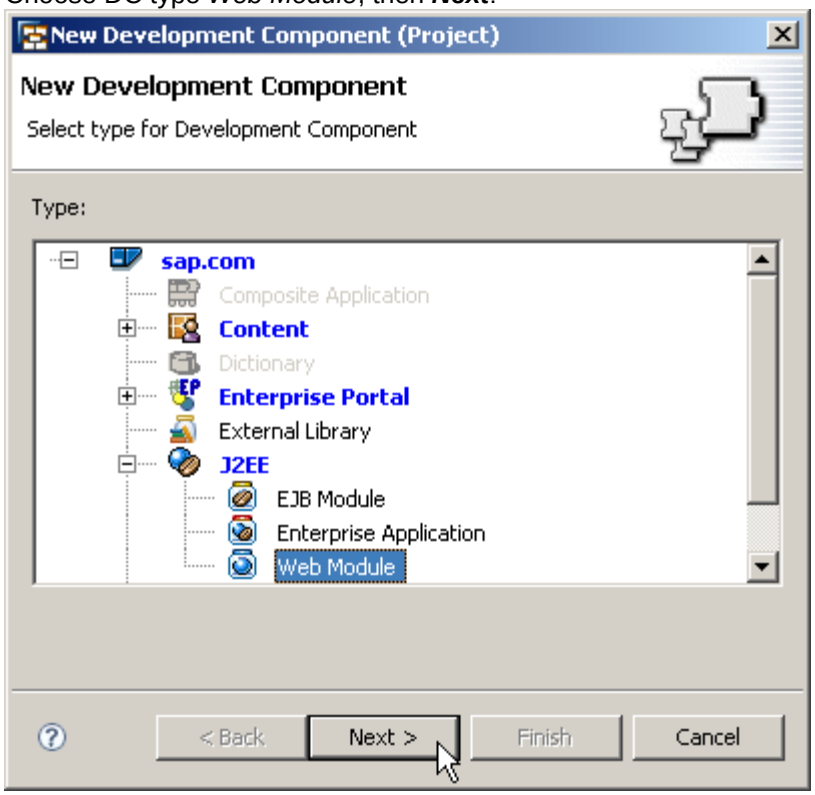
- Is to be used by defining its public part as those development objects (Java classes, interfaces, and so on) that can be used by other DCs.
- Depends on other DCs be declaring use dependencies to all other DCs that are needed. If these dependencies are used by default for a specific usage type it's added automatically. Only DCs from the required SCs can be used, all the others are not available.

This metadata is used by the build process to build DCs. Having heard about this add-on to standard development – all the other properties of the project stay the same – we can start with development. I will use the simplest example I can find – as I said, JEE development itself is not changed by the component model, so I just want to point out what has been added:

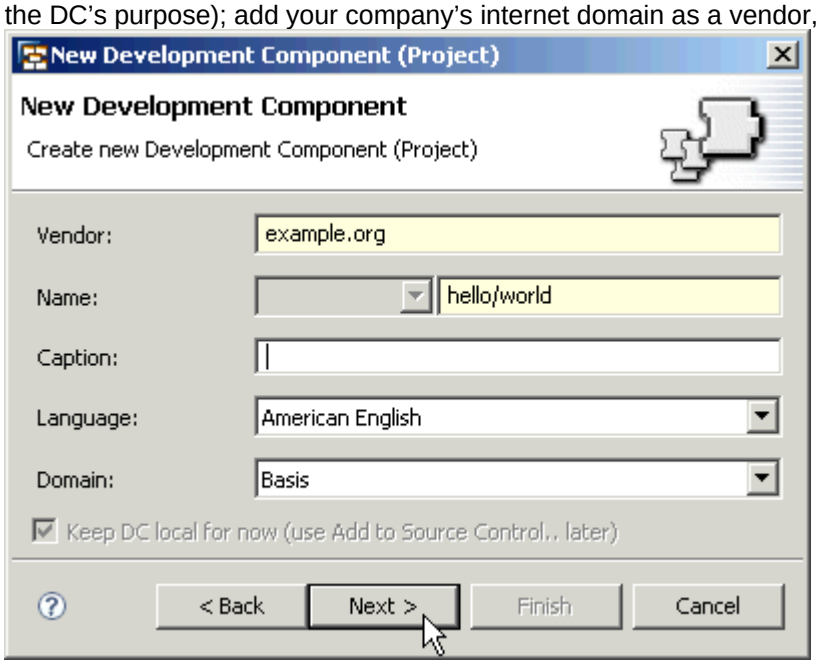
Note: SAP NetWeaver CE 7.1 Developer Studio SP1 was used to develop the first DC of the example. Then the Developer Studio version was updated to SP3 for you to compare and view screenshots. In such an update it is important that the required SCs and the version of the runtime system are also updated: ***Versions of Developer Studio, required SCs and runtime should be identical*** *– of course, the same versions must be used in the team.* This is especially important if you are using Web Dynpro Java. If you are using local development configurations, versions of required SCs will have the new version automatically.

1. Now, in the context menu of your SC create a new ***Development Component*** (DC):

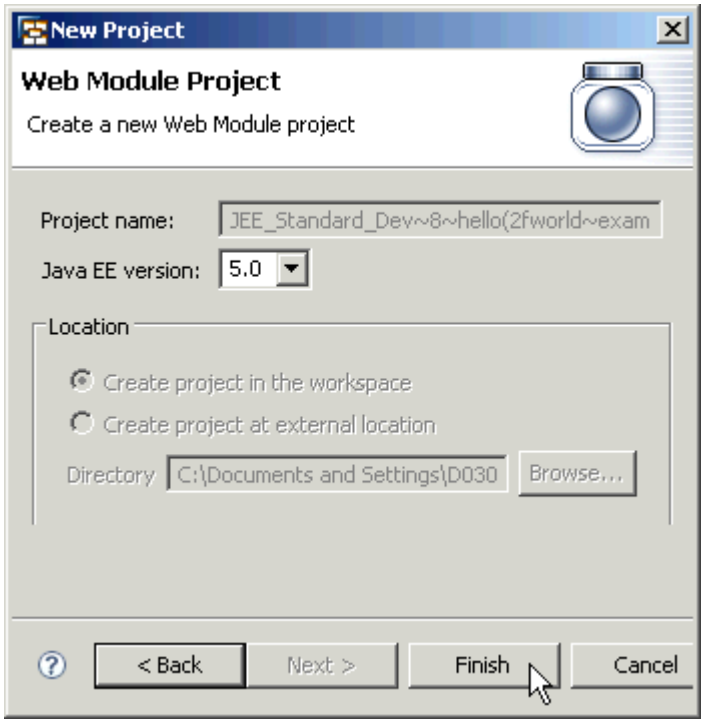
2. Choose DC type *Web Module*, then ***Next***:



3. Enter a DC *Name* (strings in lower case separated be "/", maximum length 40 characters, indicating the DC's purpose); add your company's internet domain as a vendor, then choose ***Next***:

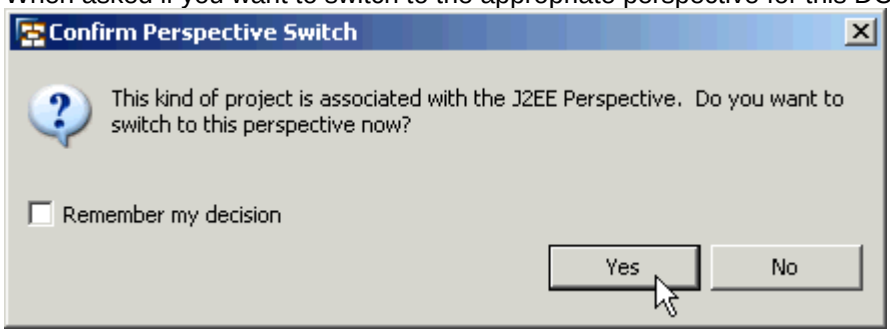


4. Choose the *JDK version* used by the project (1.4 or 5.0) and then ***Finish*** to start the creation of the DC:

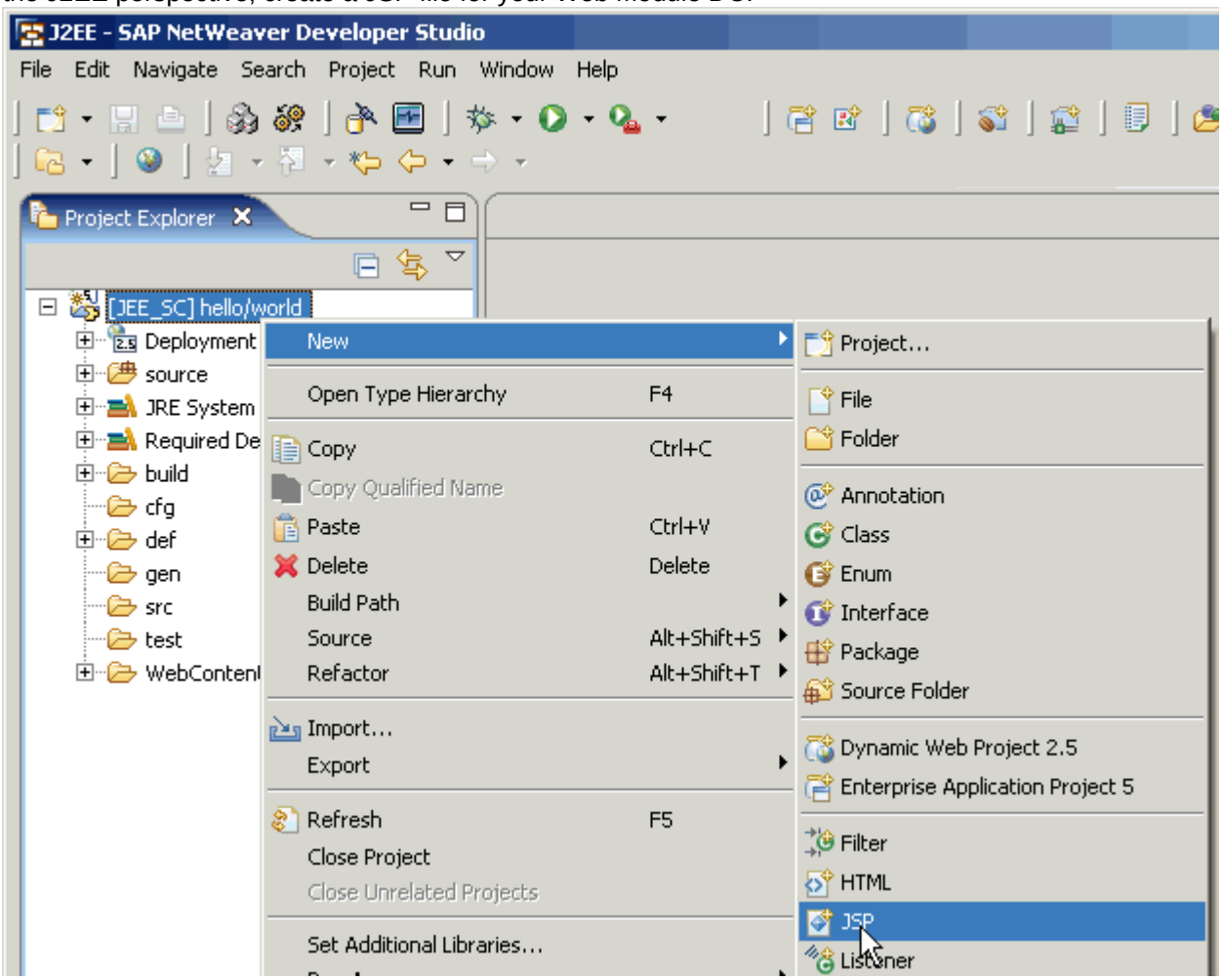


Note: If using a versioning repository, you might be asked to add the files and folders created now.

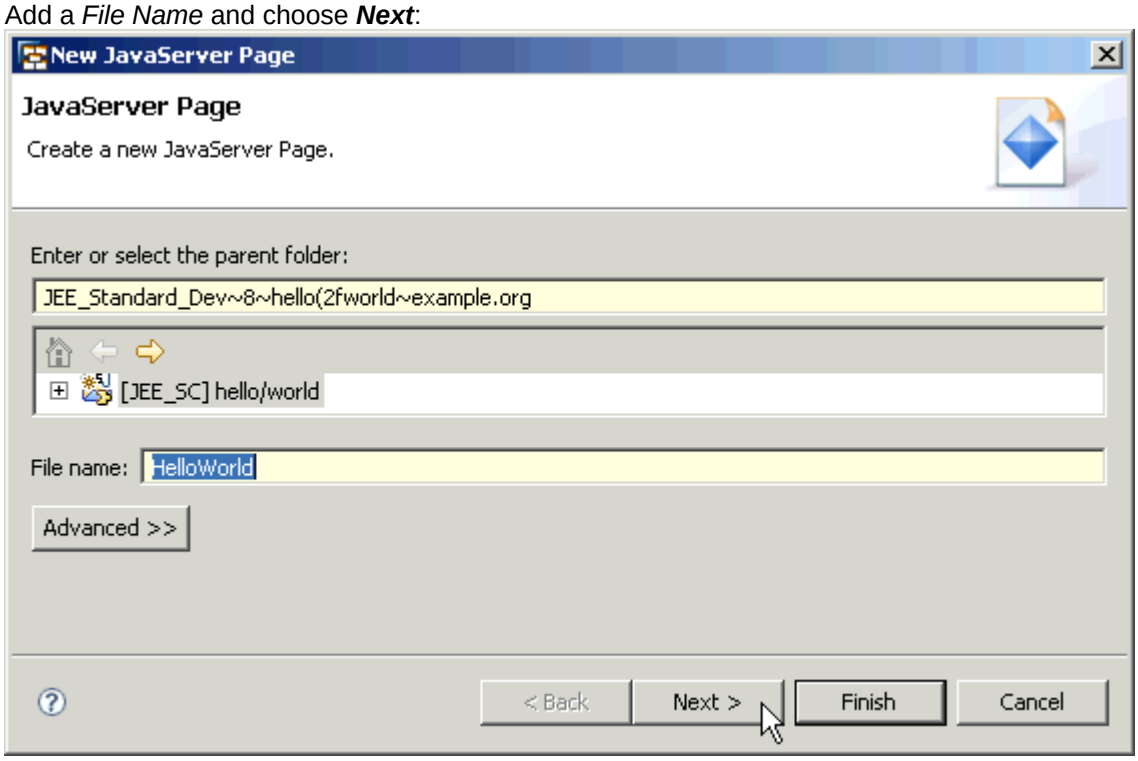
5. When asked if you want to switch to the appropriate perspective for this DC type, confirm with ***Yes***:



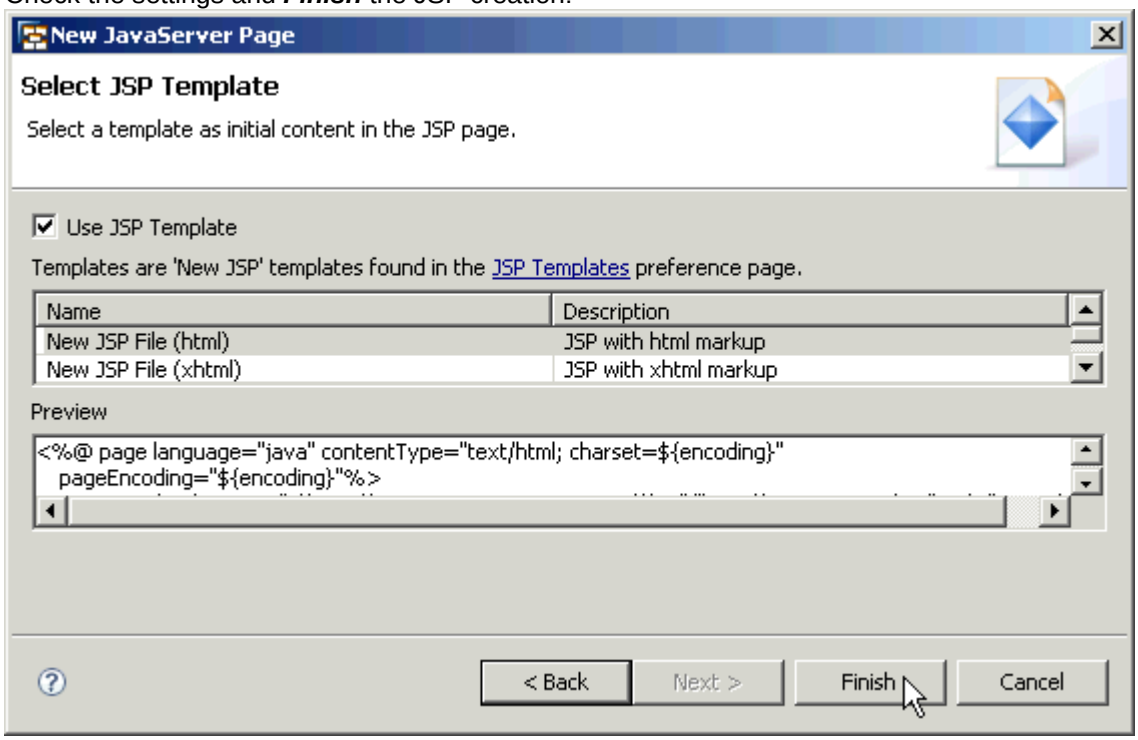
6. Once you have created a DC, you can develop the content in a specific perspective of the Developer Studio (for example, the *J2EE* or *Web Dynpro perspective*) as usual. Create development objects. In the J2EE perspective, create a JSP file for your Web Module DC:



7. Add a *File Name* and choose ***Next***:

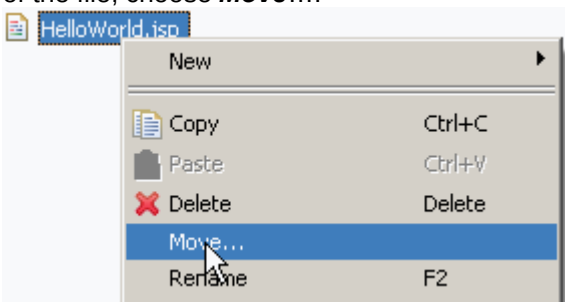


8. Check the settings and ***Finish*** the JSP creation:

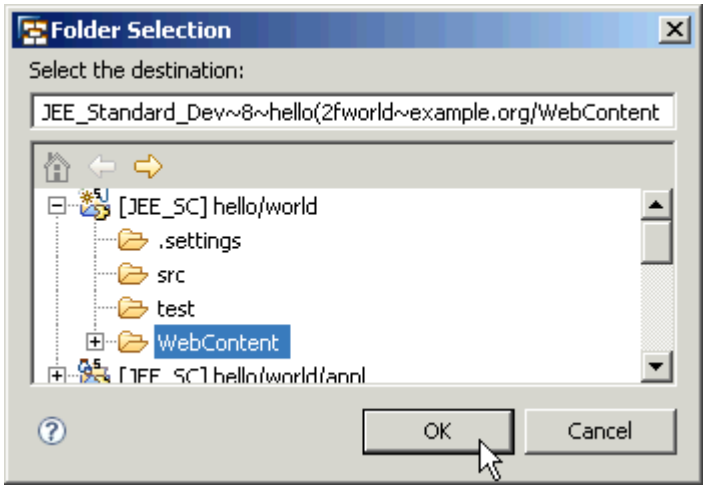


9. Edit the development object (your JSP file) and save your changes:

   a. The JSP file must be in the *WebContent* folder. If it is located elsewhere in the context menu of the file, choose ***Move***…:

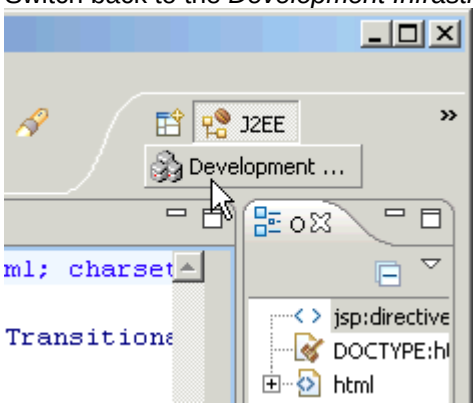


   b. Choose folder *WebContent* as the destination and confirm the move with ***OK***:

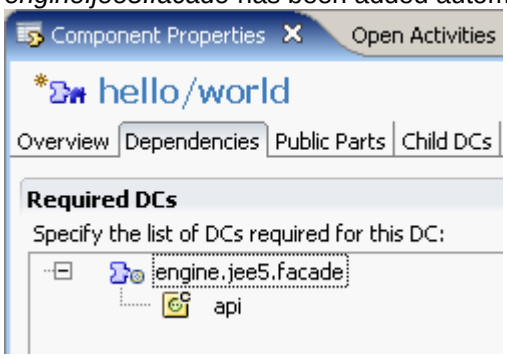


10. Now check the component properties:

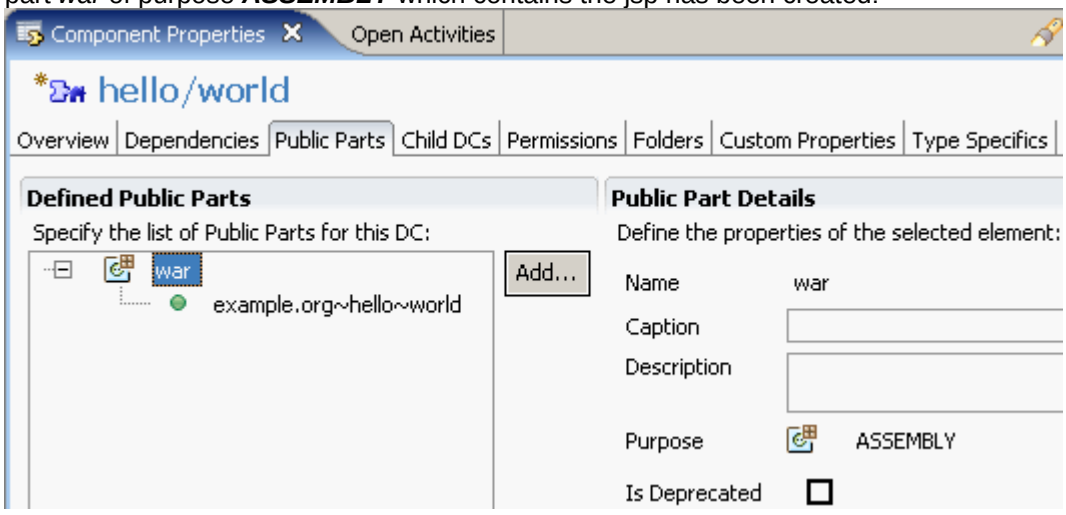
    a. Switch back to the *Development Infrastructure* perspective:



    b. In the *Component Properties* view, which is part of this perspective by default, open the tab ***Dependencies*** – you will see that a dependency to a public part api of DC *engine.jee5.facade* has been added automatically:

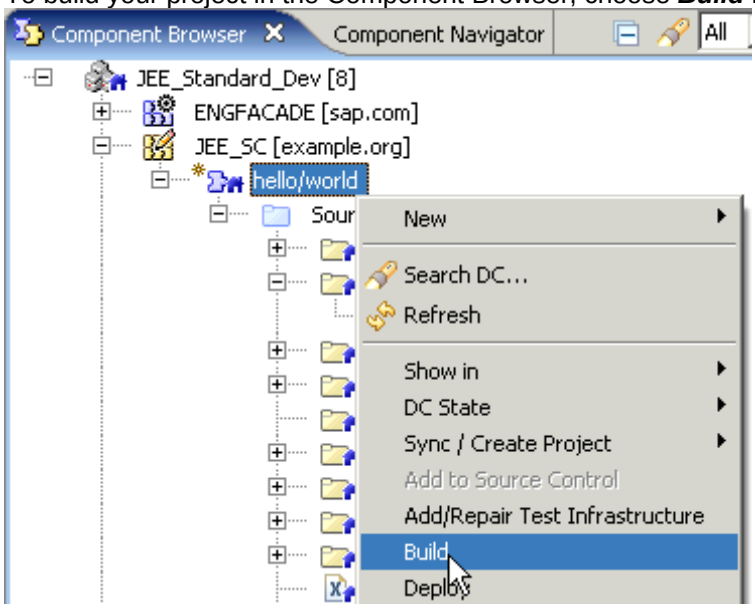


    c. To check the public part(s) of your DC, choose the tab *Public Parts* – you see that a public part *war* of purpose ***ASSEMBLY*** which contains the jsp has been created:

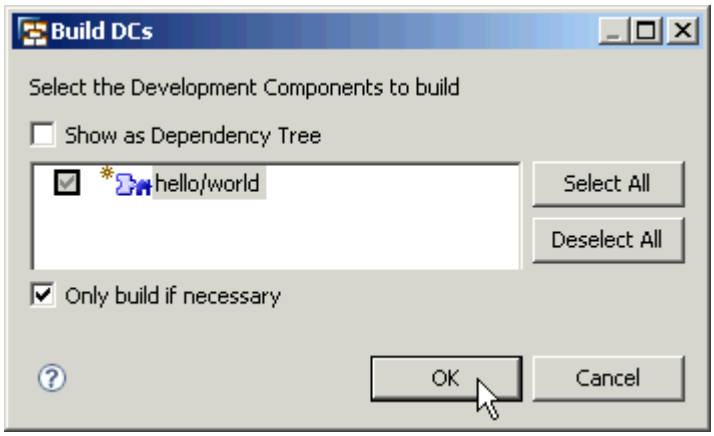


Note: A *public part of purpose* ***Assembly*** is used to package the content of a DC into another DC (for example *Java* or *Web Module* that is **not deployable by itself**, such as types to an Enterprise Application).
A *public part of purpose* ***Compilation*** is used if you need to compile against parts of another DC.
Both types of public parts can be used during *Design Time* or *Build Time*. Also, you can use a complete DC during *Deploy Time* or *Runtime* . Obviously the purpose needs to be clear when you create public parts yourself (public parts needed by default are created by the system automatically).
For a discussion of the purposes of public parts, see the SAP Help Portal under http://help.sap.com/saphelp_nw70/helpdata/en/02/6755bd296ade42931646f869b1fd15/frameset.htm.

11. To build your project in the Component Browser, choose ***Build*** in the context menu of your DC:



12. Confirm the build with *OK*:

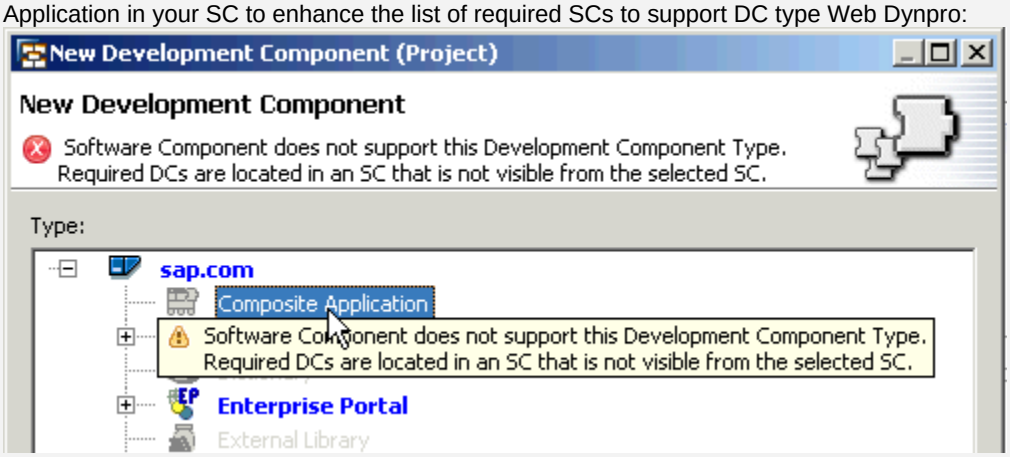


The DC is built. The resulting archive will not be deployed directly, but wrapped in an Enterprise Application.

Updating a Local Development Configuration

You will remember that in between the Developer Studio was updated from SP1 to SP3. Now, an update of the development configuration is required, since default dependencies have been changed for JEE applications. Therefore, you need to add the new SC version to the development configuration – and when updating already used SCs you need to remove the older versions first.

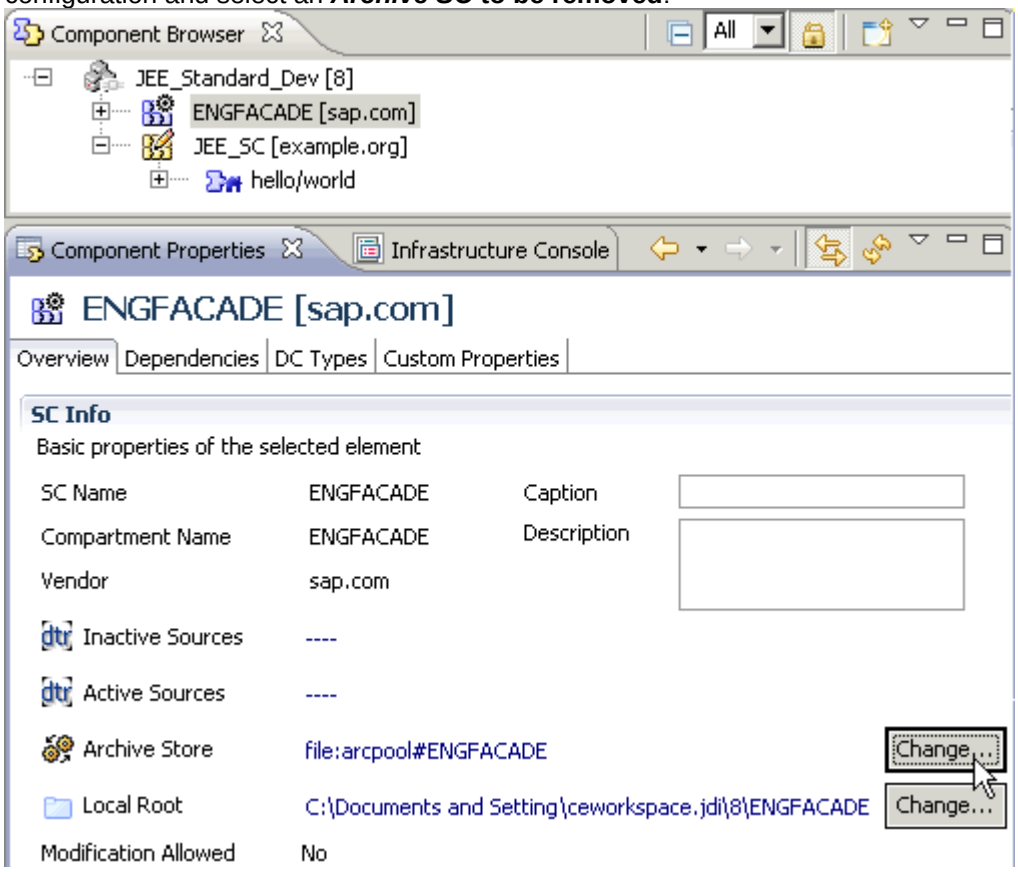
Note: Adding SCs would be used if you decide later to add a Web Dynpro UI to your application or to create a Composite Application in your SC to enhance the list of required SCs to support DC type Web Dynpro:



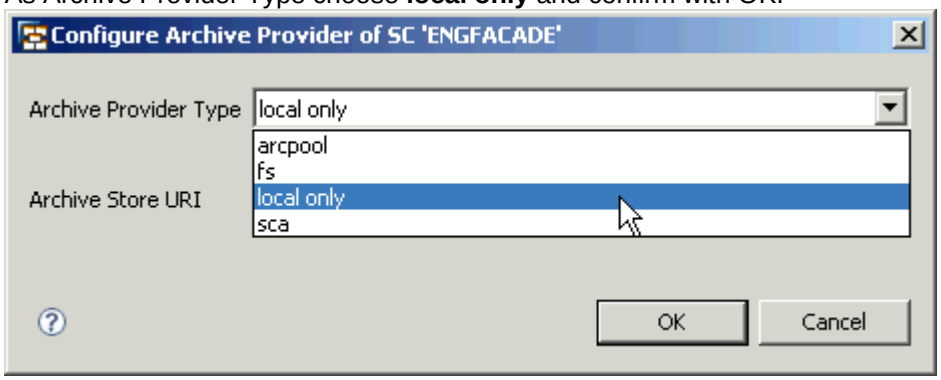
(For a list of required SCs for specific DC types, see table 1).

Removing Archive SCs

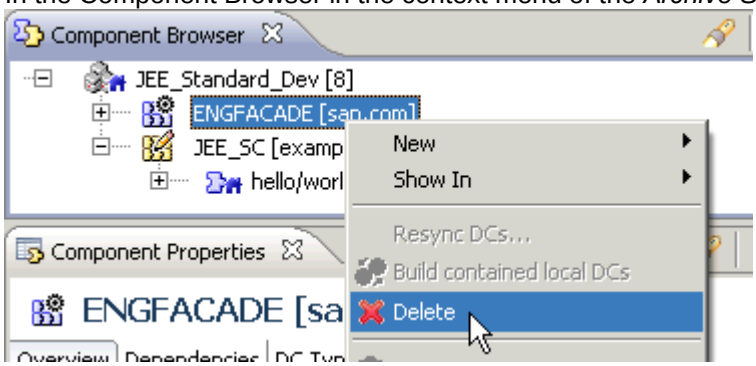
1. In the *Development Infrastructure* perspective → *Component Browser*, open your development configuration and select an ***Archive SC* to be removed**:



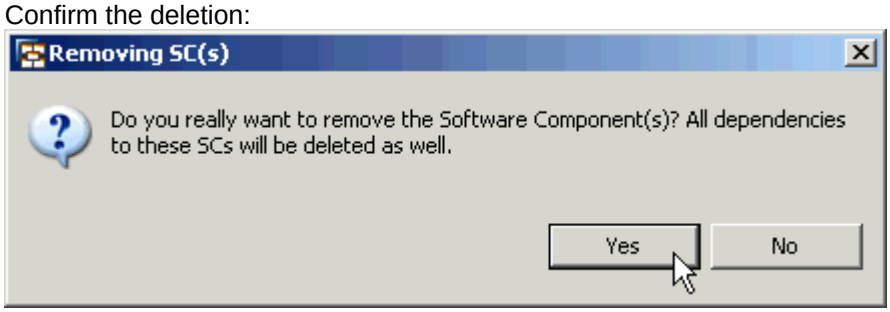
2. SCs delivered cannot be deleted in the *arcpool*, so you need to change the Archive Store:
   a. In the *Component Properties* view, choose Archive Store/Change (see above).
   b. As Archive Provider Type choose **local only** and confirm with OK.



   c. In the Component Browser in the context menu of the *Archive SC* choose delete:



d. Confirm the deletion:



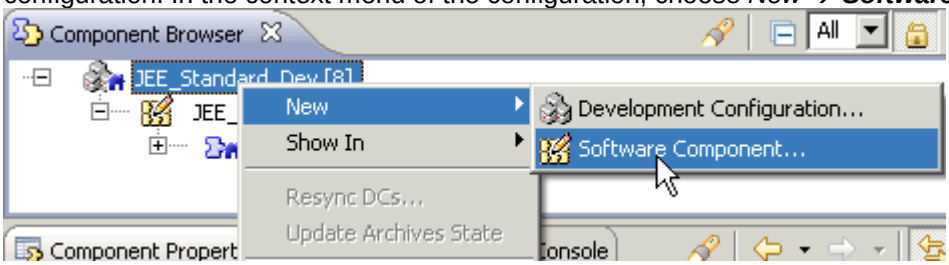
Note: As a result, the dependencies of your DCs will be invalid indicated by the red color of the Required DC:

We will fix that in the next step.

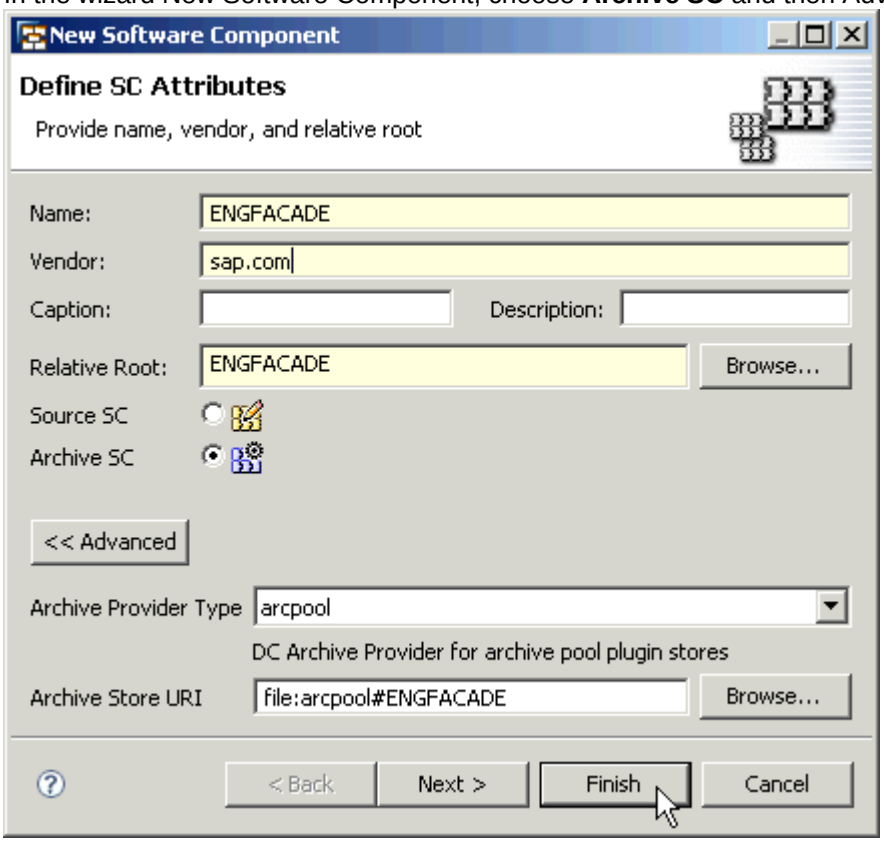
Adding Archive SCs

To fix the dependencies, you first have to re-add the required Archive SCs.

1. In the *Development Infrastructure* perspective → *Component Browser*, open your development configuration. In the context menu of the configuration, choose *New* → ***Software Component***:



2. In the wizard New Software Component, choose **Archive SC** and then Advanced.

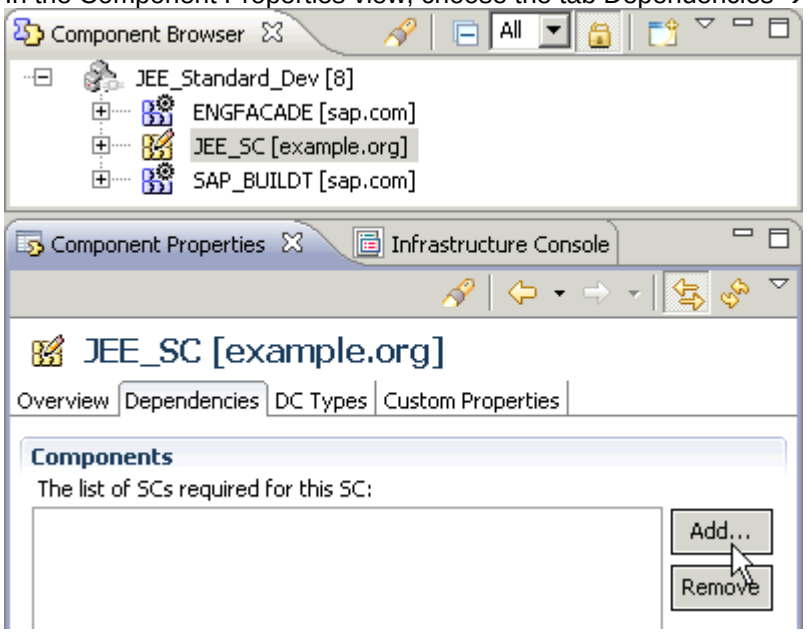


3. Enter the following data if you need an additional standard SC of SAP:
   a. Name = name of the Archive SC = **ENGFACADE**
   b. Vendor = **sap.com**
   c. Archive Provider Type = **arcpool**
   d. *Archive Store URI* = as **file:arcpool#<SC name>**
   e. Confirm with *Finish*.
4. Repeat the last steps to add *Archive SC* ***SAP_BUILDT***.

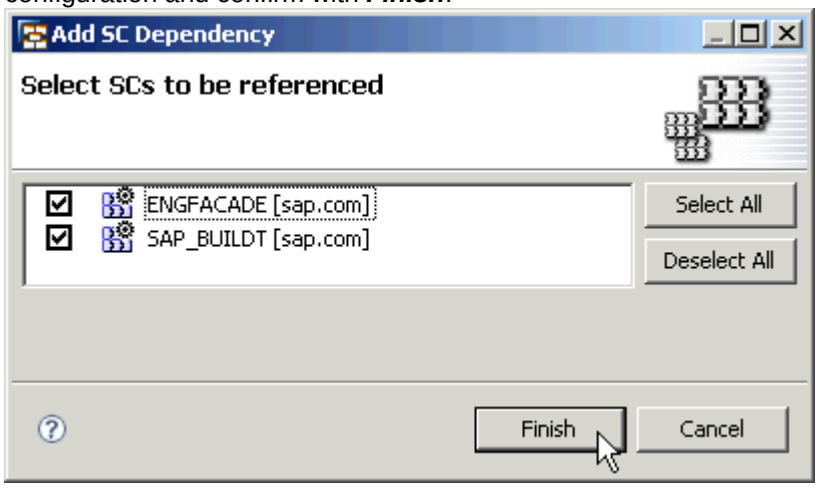
Adding SC Dependencies

Now, you need to allow your *Source SC* to use the *Archive SCs*.

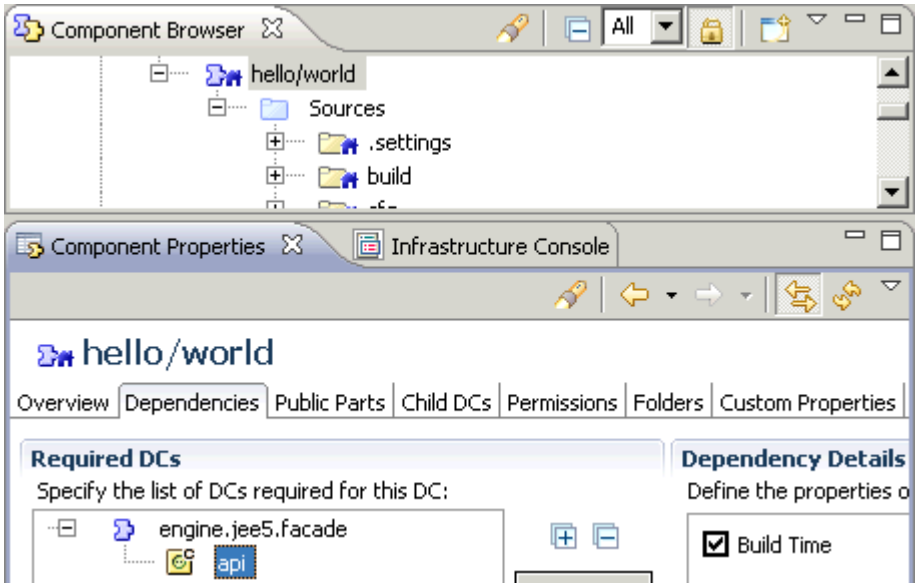
5. To make the content of the Archive SCs visible to your *Source SC*, select it in the *Component Browser*.
6. In the Component Properties view, choose the tab Dependencies → **Add**.



7. In the dialog window *Add SC Dependency*, choose the *Archive SCs* you added to your development configuration and confirm with ***Finish***:



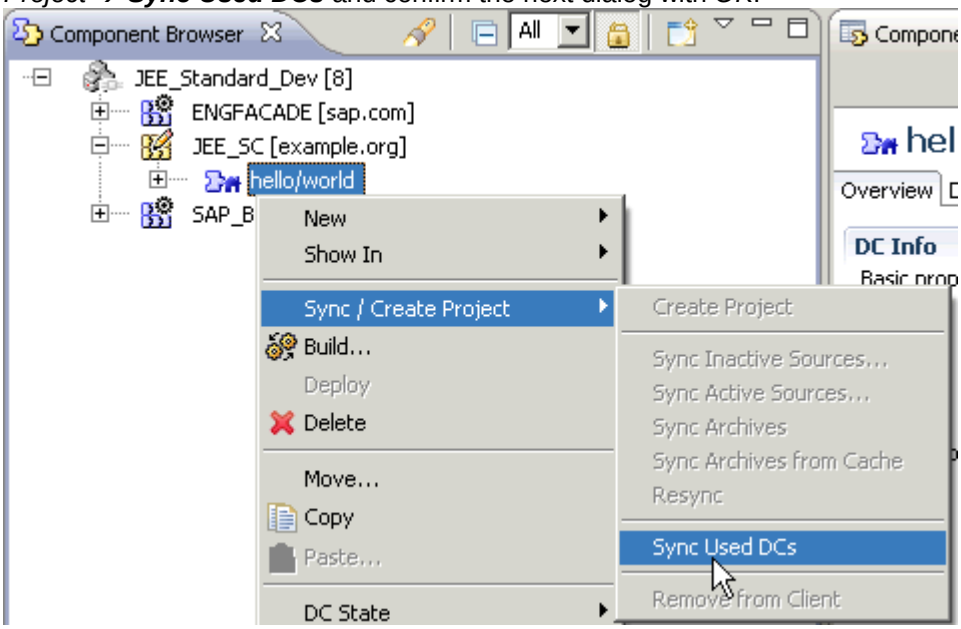
As a result, DCs from the *Source SC* now can use all DCs of the Archive SCs; the dependencies of the Source SC's DCs are valid again:



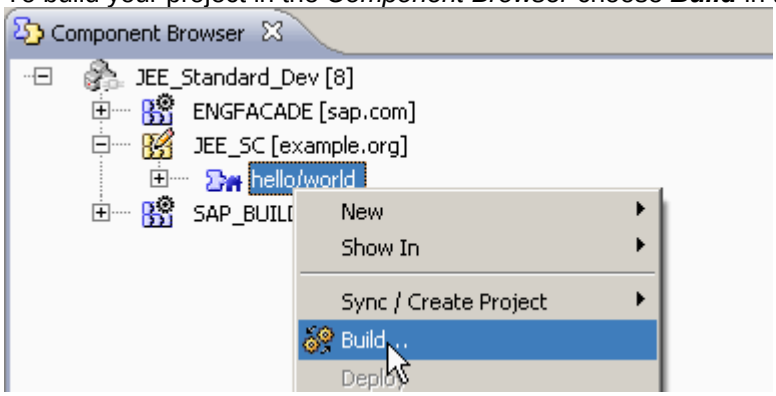
Updating the Development Components

Now that the development configuration is updated, we must build the DC again to test it against the updated archives.

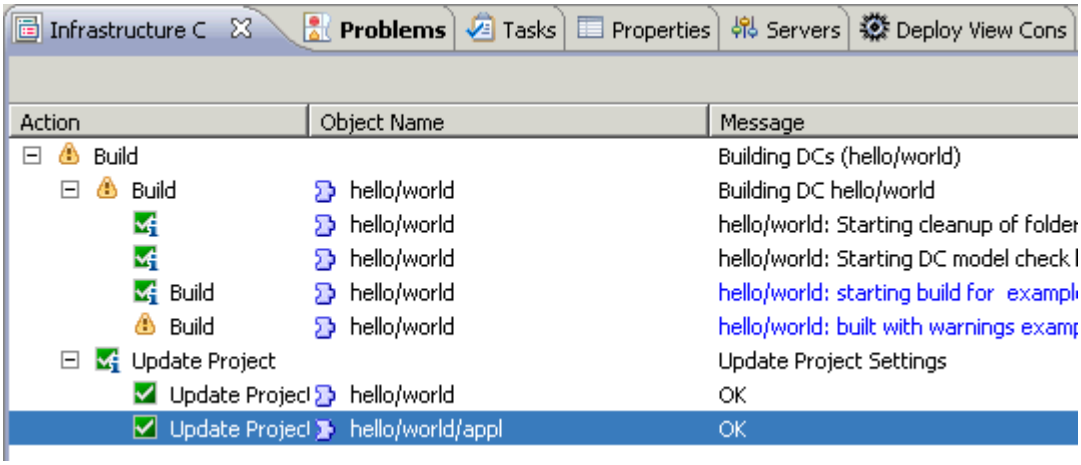
8. Open the *Component Browser*. In the context menu of the Web Module DC, choose *Sync/Create Project* → ***Sync Used DCs*** and confirm the next dialog with *OK*:



9. To build your project in the *Component Browser* choose ***Build*** in the context menu of your DC:



You can check the result of the build in the *Infrastructure Console*:



The build is successful; therefore you can (re-)use the DC from now on.

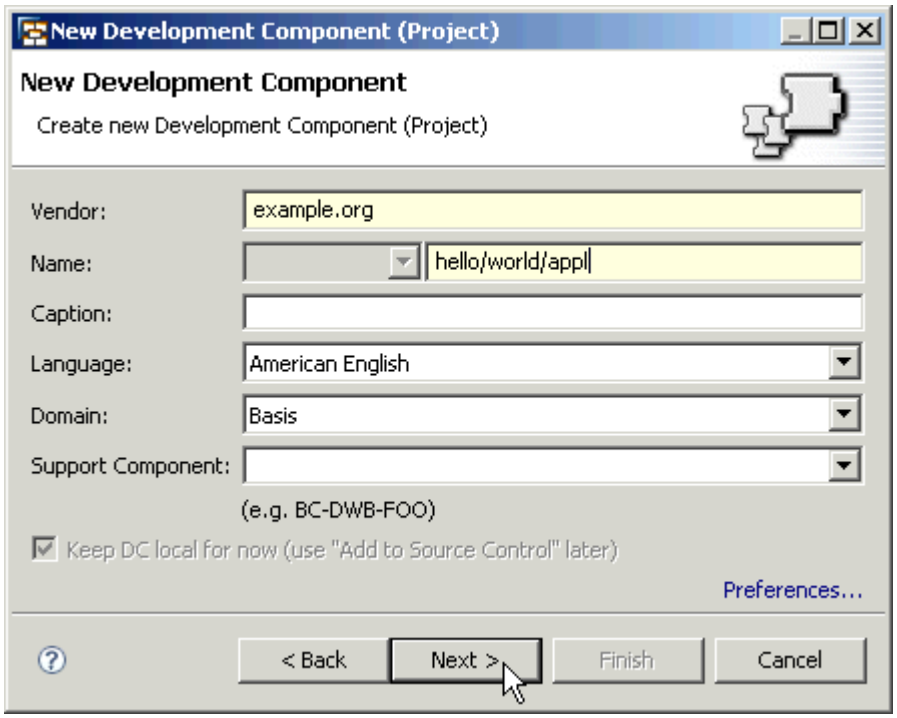
Note: The warning you see indicates an empty packages folder. Clicking a *message* in the list opens the complete build log.

Reusing a Development Component

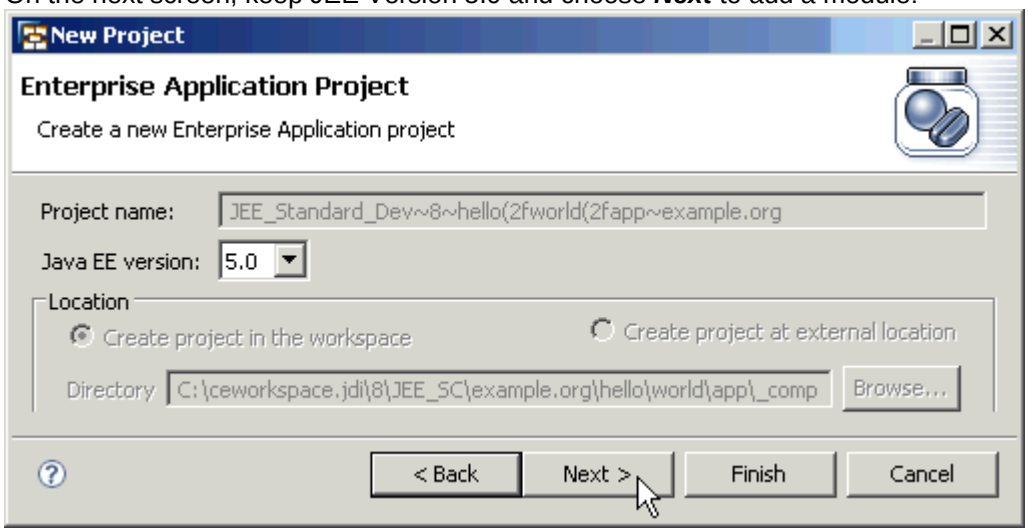
You can now use the DC you just created in other DC projects. To deploy and use a Web Module on a server, create an Enterprise Application:

1. Create a DC type *J2EE/Enterprise Application* similarly to steps 1 to 4 with the same JDK version. Use the following settings:

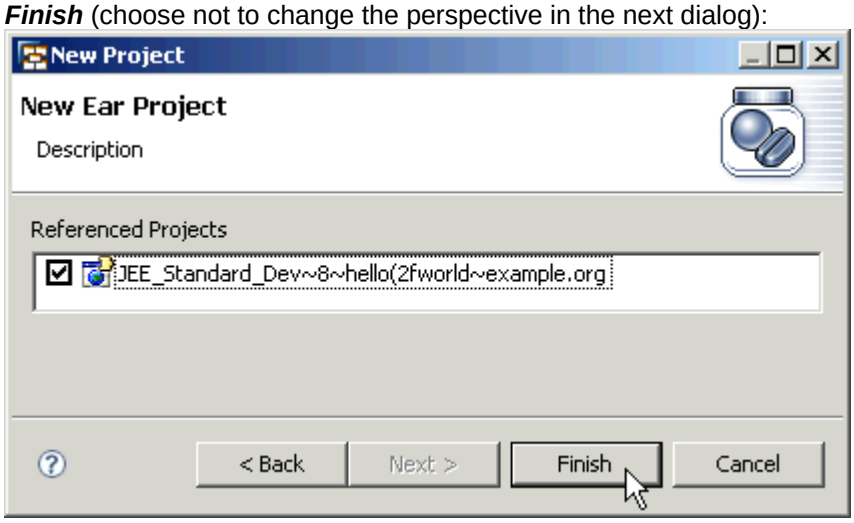
   a. *Vendor* and *Name* = same as used before then choose ***Next***:



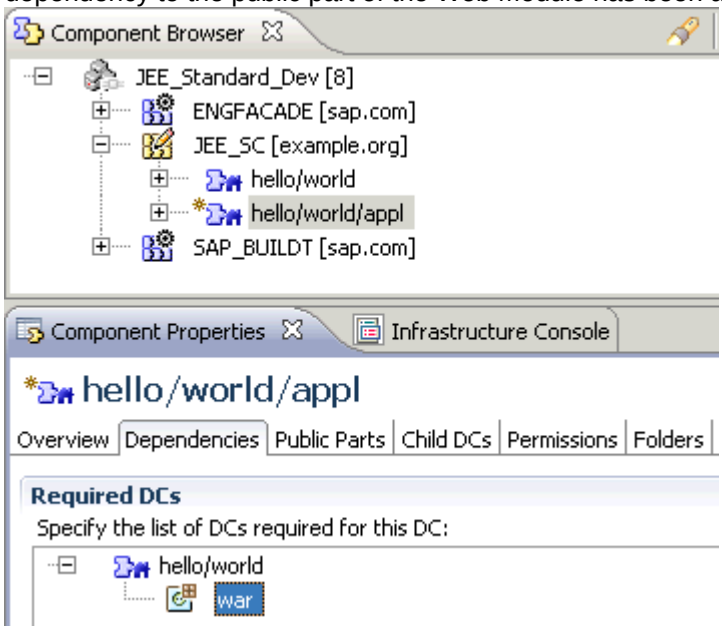
   b. On the next screen, keep JEE Version 5.0 and choose ***Next*** to add a module:



c. To add a module, choose the *Web Module DC*, which means to reuse it, and confirm with ***Finish*** (choose not to change the perspective in the next dialog):



2. Check the dependencies of your Enterprise Application in the Component Properties – a dependency to the public part of the Web Module has been added:



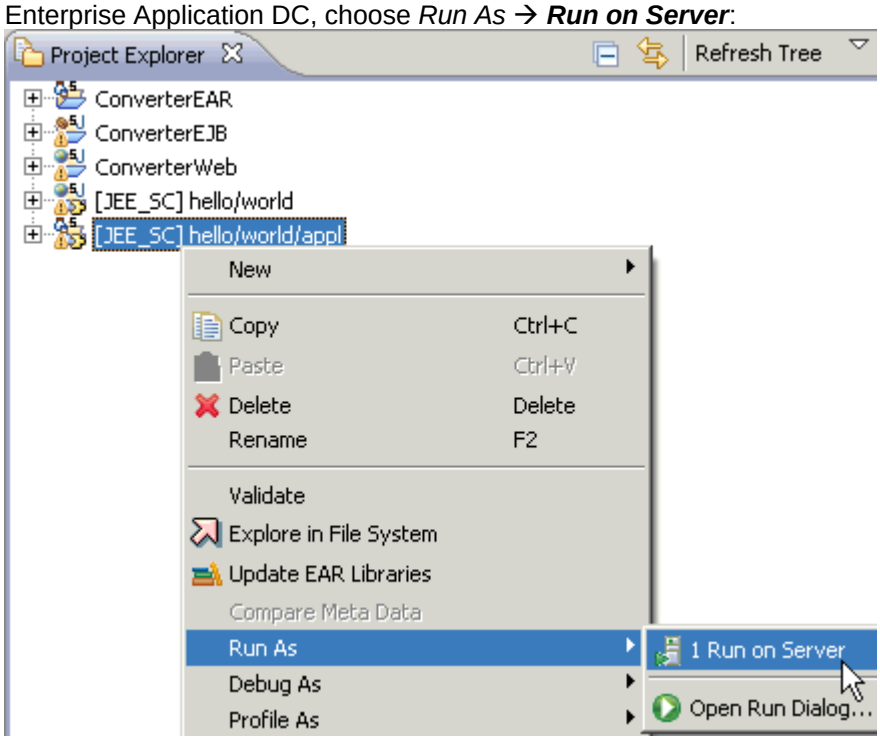
3. Build the Enterprise application DC.

Note: If problems occur, *sync the used archives* again in the context menu of the DC you want to build.

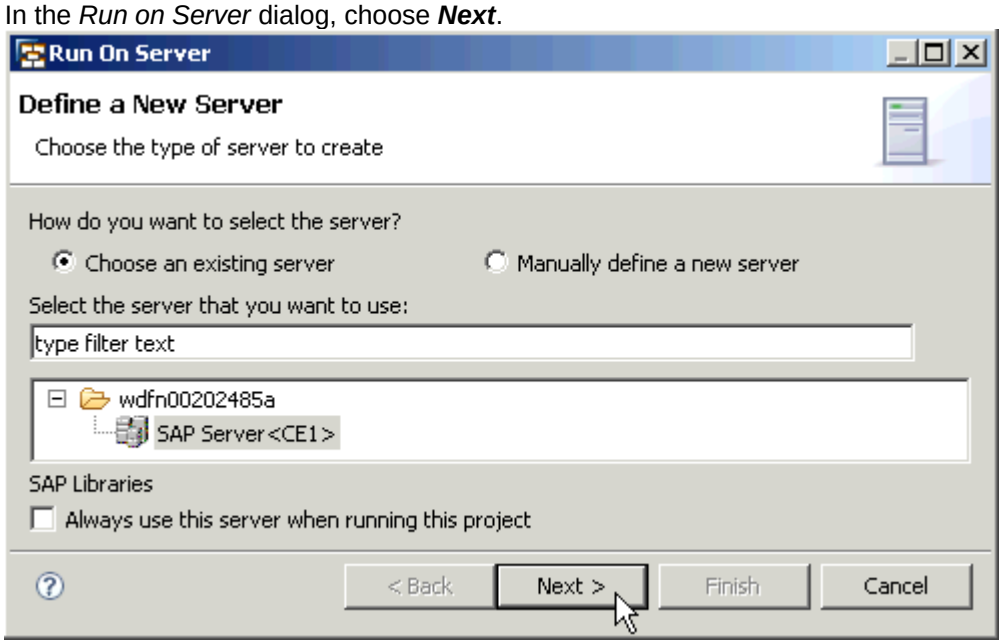
Testing an Application

After the successful build, you can now run and test your projects. (Debugging your projects if necessary is of course also possible in the Developer Studio).

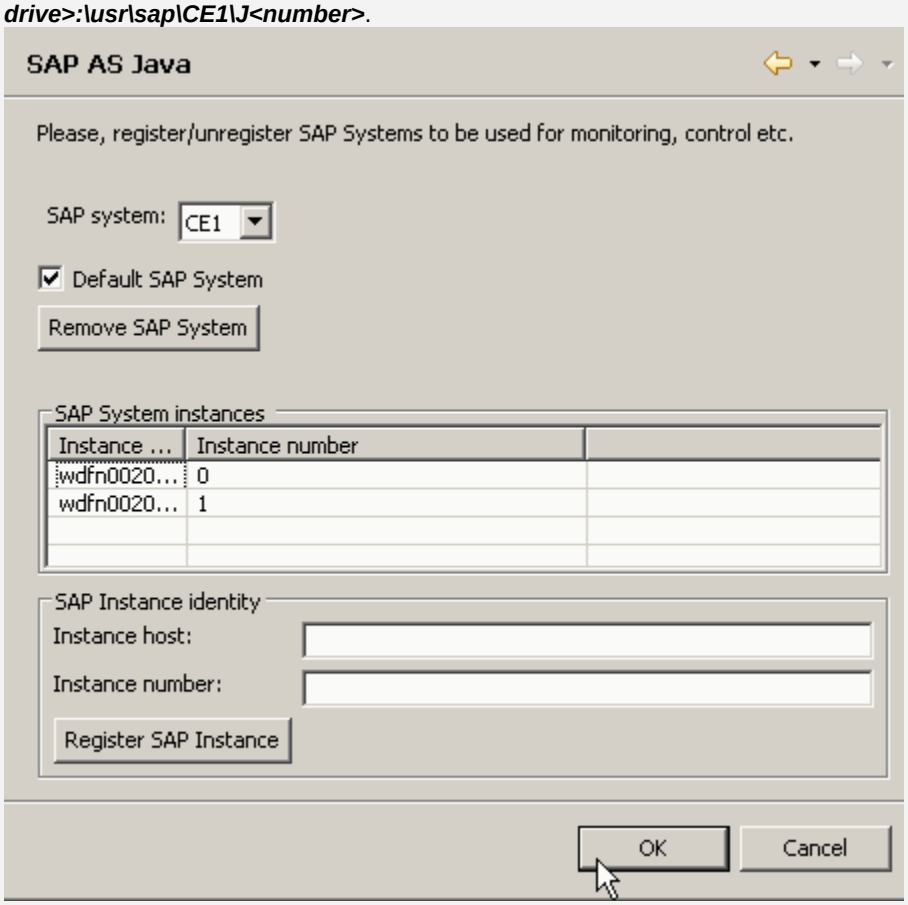
1. To deploy and run your application, go to the Java EE perspective again. In the *context menu* of your Enterprise Application DC, choose *Run As* → ***Run on Server***:



2. In the *Run on Server* dialog, choose ***Next***.

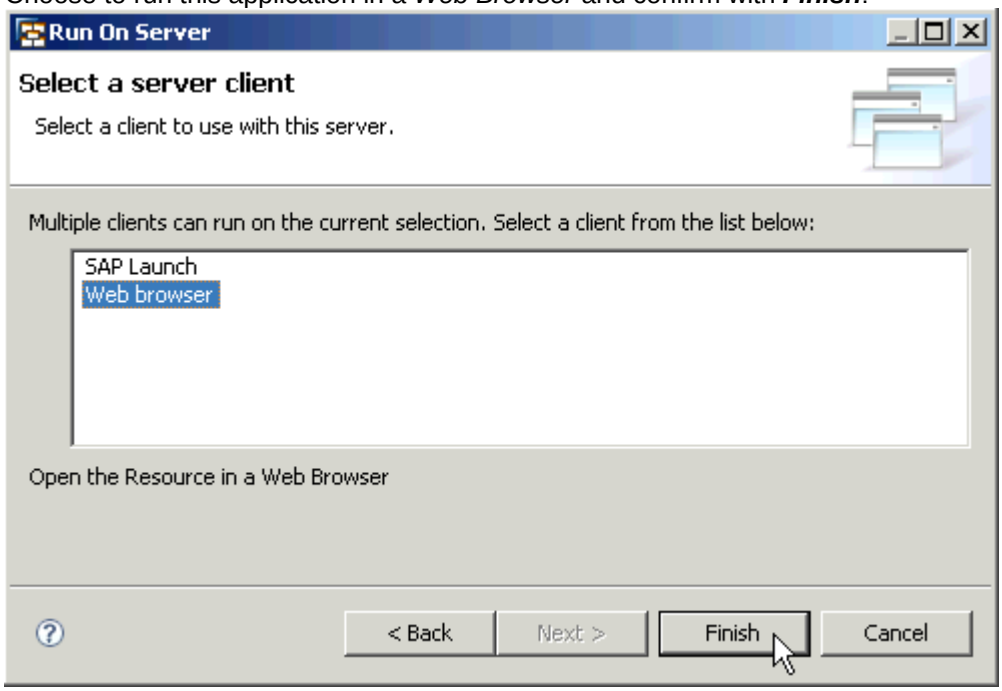


Note: In case a server has not been configured, a dialog will open where you can enter the name of your PC and instance number. Insert *Instance* = ***server name*** and *Instance number* = ***<CE installation drive>:\usr\sap\CE1\J<number>***.

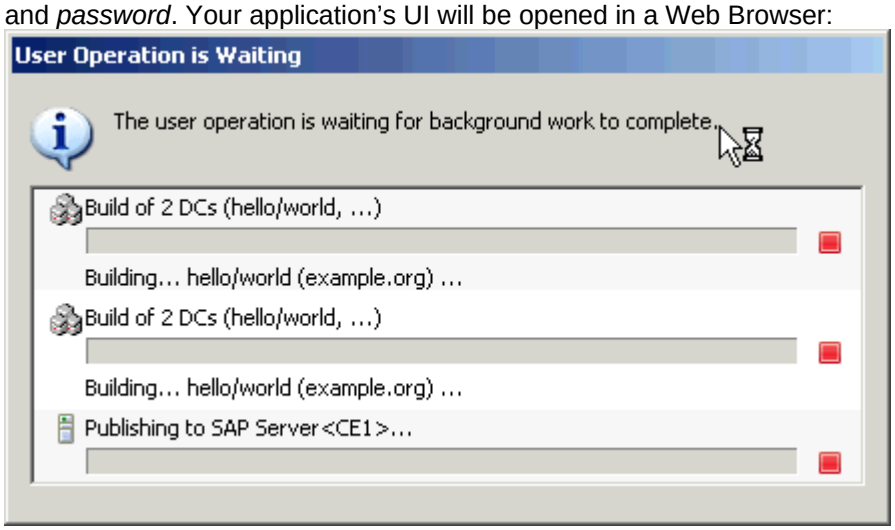


You can check the settings under *Windows* → *Preferences* → *SAP AS Java*.

3. Choose to run this application in a *Web Browser* and confirm with ***Finish***:



Build and deployment will be started automatically. If asked to do so, enter the *administrator's user* and *password*. Your application's UI will be opened in a Web Browser:



4. Pages such as *index.html* will open automatically. If you use another name, you will get an *Error 403*: In this case simply ***add the name of your JSP file to the URL*** and *go* to the new URL:

Your JSP will be displayed:

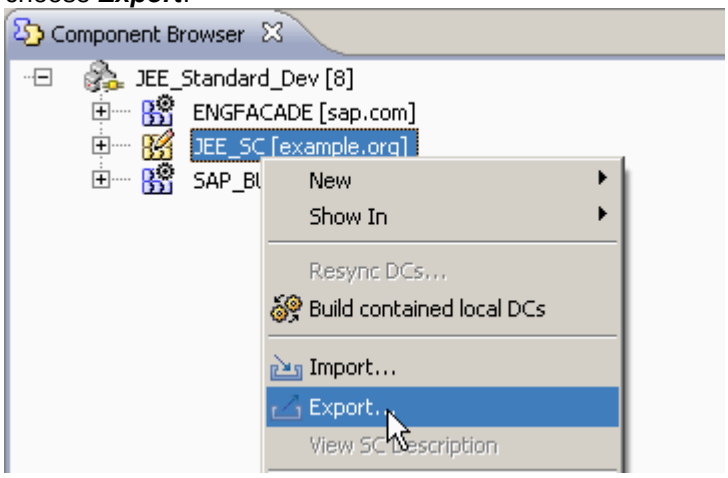
Your application is available now. The last step is to deliver it as a software component archive (SCA) file.

Delivering a Software Component

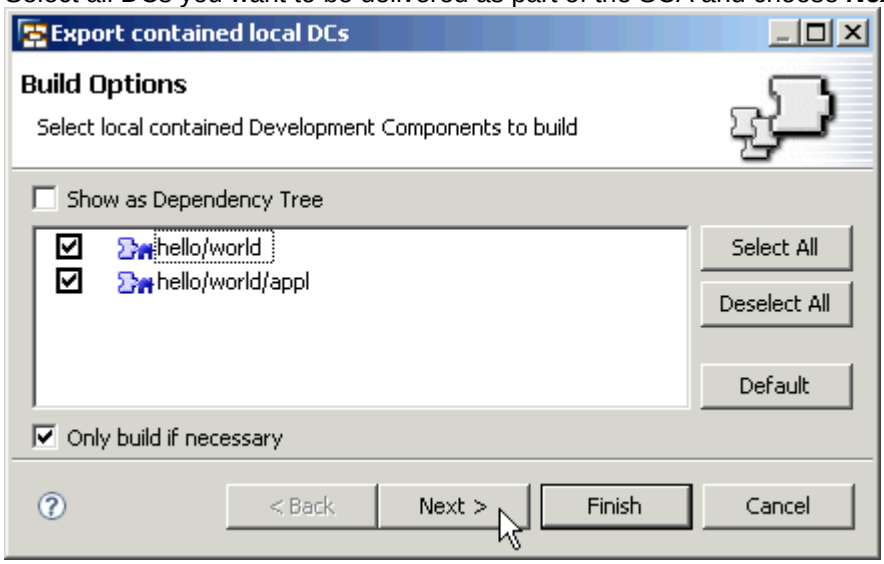
If you want to ship your DCs, you could deliver EAR files of your applications directly – the best way, however, is to put all the DCs that you have created in the context of the SC for development into the software component archive (SCA), which is the delivery form of any SC. This SCA file can be deployed into the SAP NetWeaver CE runtime system. Save your work and inform any colleagues about the availability if you are working in a team.

Note: Developer Studio version, required SCs (delivered with the Developer Studio), and the target runtime version need to be on the same release.

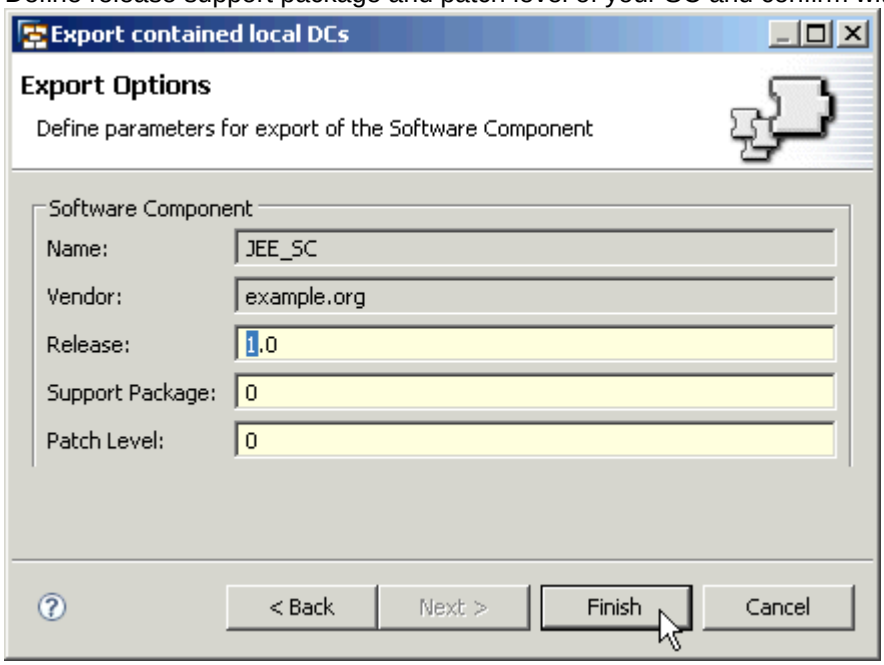
1. To create an SCA file of a specific state of your software component, navigate to the ***Component Browser*** of the *Development Infrastructure* perspective. In the context menu of your *Source SC* choose ***Export***:



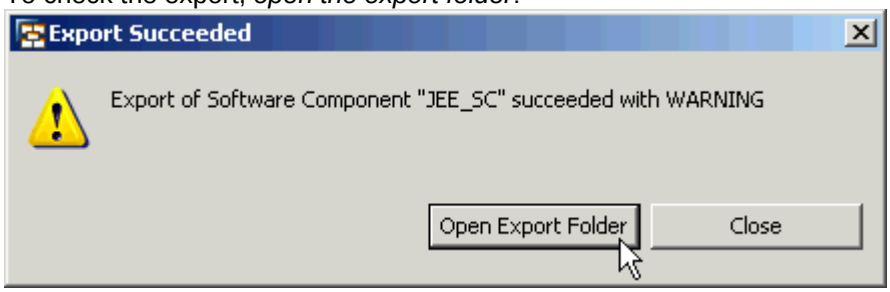
2. Select all DCs you want to be delivered as part of the SCA and choose ***Next***:



3. Define release support package and patch level of your SC and confirm with ***Finish***:

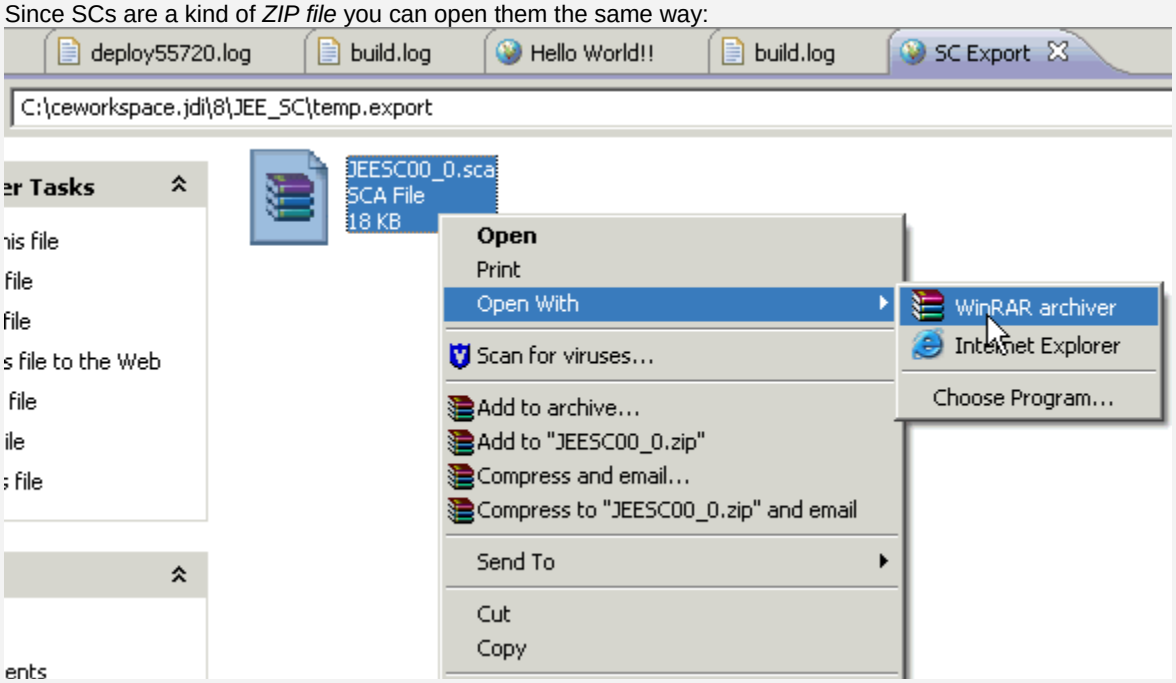


4. To check the export, *open the export folder*:

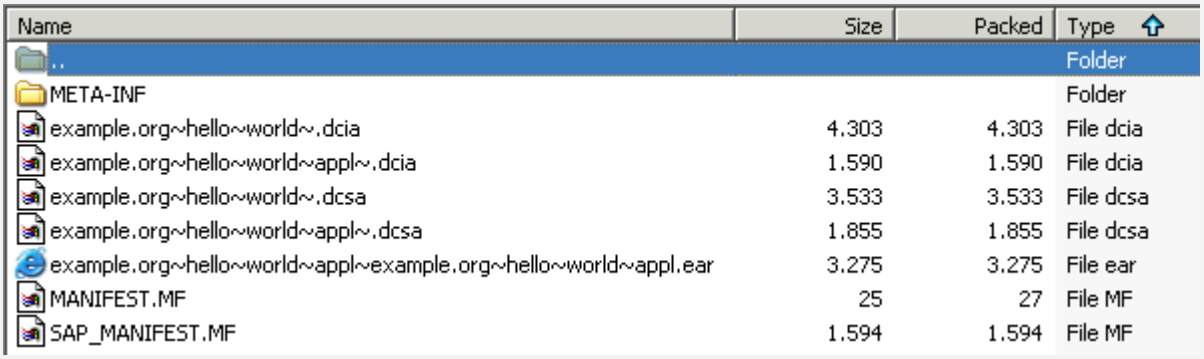


5. You will see the SCA in the SC Export tab.

Note: Since SCs are a kind of *ZIP file* you can open them the same way:



You will see the SCA file's content



Name	Size	Packed	Type
..			Folder
META-INF			Folder
example.org~hello~world~.dcia	4.303	4.303	File dcia
example.org~hello~world~appl~.dcia	1.590	1.590	File dcia
example.org~hello~world~.dcsa	3.533	3.533	File dcsa
example.org~hello~world~appl~.dcsa	1.855	1.855	File dcsa
example.org~hello~world~appl~example.org~hello~world~appl.ear	3.275	3.275	File ear
MANIFEST.MF	25	27	File MF
SAP_MANIFEST.MF	1.594	1.594	File MF

To deliver the application, simply deliver the *SCA file*. It can simply be deployed on the SAP NetWeaver release you chose when you selected the required *Archive SCs*.

How you can to work together with colleagues in a bigger project will be the topic for the third part of the article.

Removing a Development Configuration

In principle, you can remove any development configuration from its context menu. However, if you are only working locally without a source repository, you will lose your work. Therefore, in a local development scenario you should wait until the SCA file is created – you can then recreate this scenario any time by adding the SCA in question as SC for development, plus the set of archive SCs used before.

Conclusion

If you want to deliver JEE applications targeting SAP NetWeaver CE runtime systems, but do not need a complete infrastructure for your work – or are already using one – the scenario described here is what you were looking for. It provides you with a set of tools to create, build, and package applications according to SAP's component model. All these functions are delivered with the SAP NetWeaver Developer Studio so system requirements and installation effort are minimal. The steps of the development process in a single user installation can all be performed in the developer studio. A key function is the definition of a development environment (creating a local development configuration) *as you want*, which includes the definition of your own software component. The outcome of the process is a software component archive fully compatible with SAP NetWeaver and SAP's component model.

Related Content

You will find related SDN documents or web pages under the following links:

- The Fast Way to Component-Based Development Using SAP NetWeaver CE 7.1 – Part I: Concepts & Development Scenarios
- SDN Subscription Program – development licenses
- SAP NetWeaver Development Infrastructure, which brings in-depth discussions of the NWDI Services and SAP's component model

Copyright