Integrating Web Dynpro and SAP NetWeaver Portal: How to use extended features of the SAP Application Integrator

Applies to:

Integration of Web Dynpro for Java applications into the SAP NetWeaver Portal for the SAP NetWeaver 04s release.

Summary

The basic task of the SAP Application Integrator is to launch a Web Dynpro application running as a Web Dynpro iView within the SAP NetWeaver portal. Besides that there are several extended features like customer-specific parameter providers for the URL computing available, which helps also when running Web Dynpro iViews. The document gives a comprehensive overview about the SAP Application Integrator and describes how to use the extended features when running Web Dynpro iViews.

Author: Jochen Guertler

Company: SAP AG

Created on: 27th February 2007

Author Bio

Jochen Guertler works as a development architect within the Web Dynpro for Java team. His main responsibilities are the integration of Web Dynpro for Java with other components of SAP NetWeaver, especially the integration with the SAP NetWeaver portal.

Jochen is co-author of the “Maximizing Web Dynpro for Java” book.

Table of Contents

Integrating Web Dynpro and SAP NetWeaver Portal: How to use extended features of the SAP Application Integrator 1
Applies to: 1
Summary 1
Author Bio 1
Table of Contents 2
Introduction 3
Classification of Web Dynpro iViews 3
Prerequisites 3
Using the examples 3
Deploying the example application 3
Deploying the portal content 3
The SAP Application Integrator 4
Launching a Web Dynpro iView 4
Computing of the Web Dynpro application URL 4
Handling of NW04 Web Dynpro iViews 7
Handling of NW04s Web Dynpro iViews 7
Extending the URL template 7
<IView> context 8
<Profile> context 8
<Request> context 9
<System> context 10
<User> context 10
Using customer-specific parameter providers 11
Prerequisites 11
Limitations 12
First example 12
Creating a portal development component 12
Creating a portal service 14
Implementing the customer-specific parameter provider 17
Extending the service implementation 18
To get it compiled 19
Web Dynpro related scenarios 20
Dynamic computation of system or Web Dynpro application 21
Forwarding portal session content 21
Forwarding the iView profile 23
Mapping of JCo destinations 25
Copyright 27

Introduction

The basic task of the SAP Application Integrator is to launch a Web Dynpro application running as a Web Dynpro iView within the SAP NetWeaver Portal. Besides that there are several extended features like *customer-specific parameter providers* for the URL computing available, which helps also when running Web Dynpro iViews.

The document gives a comprehensive overview about the SAP Application Integrator and describes how to use the extended features when running Web Dynpro iViews including several running example applications using the discussed capabilities.

Classification of Web Dynpro iViews

Before we continue with the SAP Application Integrator we have to clarify the naming used in this document regarding the different types of Web Dynpro iViews.

We differentiate between two types of Web Dynpro iViews. First there are Web Dynpro iViews based on the *Web Dynpro iView template*. This type of Web Dynpro iView could only be run as isolated iViews. As this iView type is available since the SAP NetWeaver 04 release we name this *NW04 Web Dynpro iView*.

Second there are Web Dynpro iViews based on the *Web Dynpro page builder*. This type of Web Dynpro iView could be run embedded in Web Dynpro pages. As this iView type is available since the SAP NetWeaver 04s release we call this *NW04s Web Dynpro iView*.

For more details about the differences and capabilities of the Web Dynpro iView types please check the related document of this series.

Prerequisites

This document and the provided examples are developed and tested using the SAP NetWeaver 04s SP11 release.

Using the examples

To demonstrate the different capabilities of the SAP Application Integrator we provide a running example application: `ParametersApp`. This application shows the parameters passed to the Web Dynpro application. It furthermore allows filtering the list of parameters to show or hide for example the SAP internal parameters.

Deploying the example application

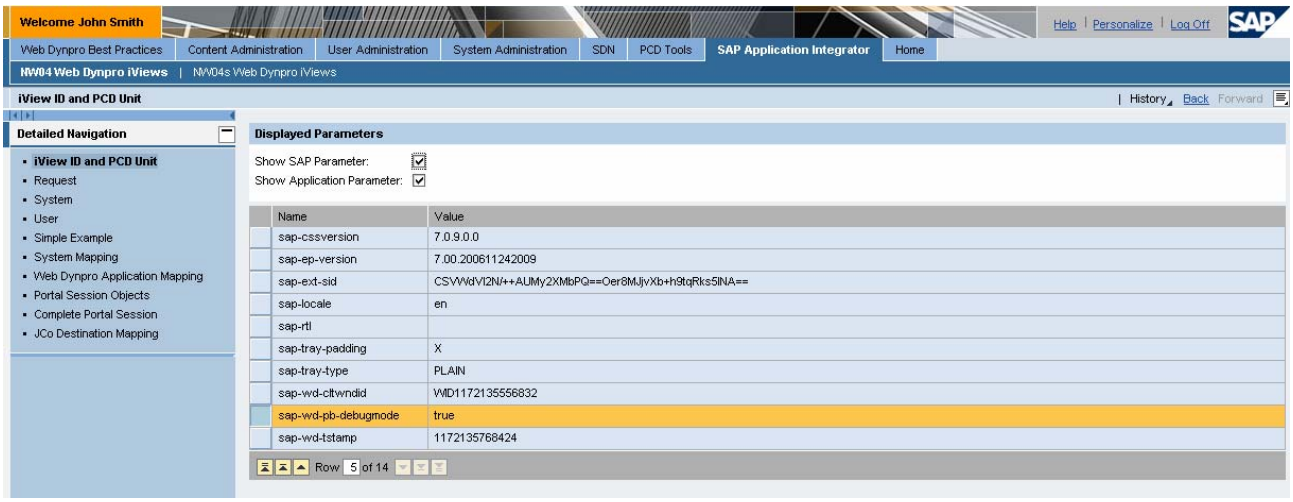
The `ParametersApp` example application is part of the `params` Web Dynpro development component. To deploy this development component you have to import it as a local development component into your NetWeaver Developers Studio. You can find the zipped development component here.

Deploying the portal content

After deploying the `params` development component you have to import the associated portal content under **System Administration -> Transport -> Transport Packages –> Import**. You can find the related portal content archive here.

After deploying the portal content you have to assign the provided SAP `Application Integrator` role under **User Administration -> Identity Management** to your user.

The following screenshot shows the provided example application running in the SAP NetWeaver Portal. We use the `ParameterApp` example application in several iView showing different usages of the SAP Application Integrator capabilities.



The SAP Application Integrator

One main benefit using the SAP NetWeaver Portal is the possibility to structure and offer different types of applications in one consistent way. Besides non-SAP applications all kinds of applications based on one of the available SAP technologies are supported.

The *SAP Application Integrator (AI)* is responsible to launch all types of SAP applications like applications based on HTMLB, ITS, BSP, Transactions or last but nor least Web Dynpro (both for Java and ABAP). The SAP Application Integrator is implemented as a portal component executed by the *Portal Runtime (PRT).*

Launching a Web Dynpro iView

Independently of the used Web Dynpro (for Java) iView type the main steps to launch a Web Dynpro application using the SAP Application Integrator are the following:

1. The iView URL is sent from the browser and it is handled by the Portal Runtime (PRT).
2. The PRT forwards the request to the SAP Application Integrator.
3. The SAP Application Integrator computes the target Web Dynpro application URL based on the settings of the launched iView. This application URL contains several parameters defining the current portal environment.

   For NW04 iViews the defined system is used to determine the host running the Web Dynpro application.

   The computed URL is based on a *URL template* which could be also extended for certain iViews. As we will describe later there is also a possibility to define certain exits for dynamic parameter resolving.
4. The SAP Application Integrator sends a HTTP redirect to the browser pointing to the computed Web Dynpro application.
5. The Web Dynpro runtimes starts the Web Dynpro application. The parameters defining the current portal environment are used automatically by the WD runtime to ensure that the Web Dynpro application is executed correctly. The most obvious example for this is the automatic usage of the theme defined by the SAP NetWeaver Portal.

Computing of the Web Dynpro application URL

As the SAP Application Integrator is responsible to run different types of applications depending on different URLs there are *URL templates* defining the structure and different parts of a computed URL. The URL template for a specific type of application is defined inside the SAP Application Integrator and there is no

need to change this. We will discuss later in this document how to extend the used URL template to add more parameters to the computed URL.

The URL template used to compute a Web Dynpro application URL for NW04 Web Dynpro iView looks like the following (expressions denoted in angle braces <…> are resolved during runtime):

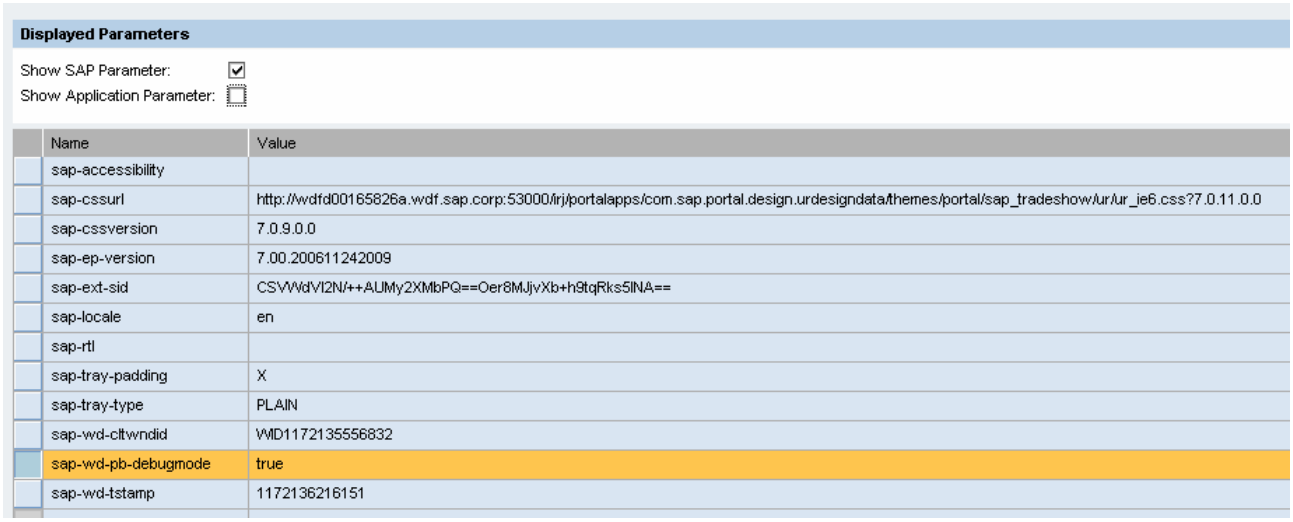
```
URL = <System.Access.WAS.protocol>://
  <System.Access.WAS.hostname>\
  /webdynpro/dispatcher<Request.DistributionZone>/
    <WebDynproNamespace>/<WebDynproApplication>\
  ;jsessionid=<Request.JSessionID>?\
  sap-ext-sid=<ESID[URL_ENCODE]>&\
  sap-wd-cltwndid=<ClientWindowID>&\
  sap-locale=<Request.Locale>&\
  sap-accessibility=<User.Accessibility[SAP_BOOL]>&\
  sap-rtl=<LAF.RightToLeft[SAP_BOOL]>&\
  sap-ep-version=<Request.Version[URL_ENCODE]>&\
  sap-cssurl=<LAF.StylesheetUrl[URL_ENCODE]>&\
  sap-cssversion=<LAF.Version[URL_ENCODE]>&\
  <Authentication>&\
  <DynamicParameter[PROCESS_RECURSIVE]>&\
    <ApplicationParameter[PROCESS_RECURSIVE]>
```

Once the URL template has been processed by the SAP Application Integrator, the following URL could, for example, be calculated which could be used to launch the `ParameterApp` example application as NW04 Web Dynpro iView within the SAP NetWeaver Portal:

```
http://wdhost.wdf.sap.corp:50000/webdynpro/dispatcher/
  sap.com/params/ParameterApp;
  jsesionid=(WDFD00146855A_P37_00)
  ID1615564250DB00783426715781195164End?
  sap-extsid=w5pCY0L2j19l%2FB28gt1HhA%3D%3Dv
  Vf%2B0Ky5v%2BiZ268a%2BYvFjA%3D%3D&
  sap-wd-cltwndid=WID1138899091639&
  sap-locale=en_US&
  sap-accessibility=&
  sap-rtl=&
  sap-ep-version=6.4.200509182320&
  sap-cssurl=http%3A%2F%2Fwdfd00146855a%3A50000 %2
  Firj%2Fportalapps%2Fcom.sap.portal.design.urdesigndata
  %2Fthemes%2Fportal%2Fsap_tradeshow%2
  Fur%2Fur_ie6.css%3F6.0.15.0.0&
  sap-cssversion=6.0.14.0.0
```

Based on the transferred parameters, different portal settings are sent to the Web Dynpro runtime environment. In addition to the preferred portal language (`sap-locale`) and the selected theme (`sap-cssurl`), these settings also include information on the preferred display variant (left-to-right or right-to-left defined by the `sap-rtl` parameter) and on determining whether a barrier-free display is required (`sap-accessibility`).

The next picture shows the `ParametersApp` example application showing the passed parameters.



Displayed Parameters

Show SAP Parameter: ☑
Show Application Parameter: ☐

Name	Value
sap-accessibility	
sap-cssurl	http://wdfd00165826a.wdf.sap.corp:53000/irj/portalapps/com.sap.portal.design.urdesigndata/themes/portal/sap_tradeshow/ur/ur_ie6.css?7.0.11.0.0
sap-cssversion	7.0.9.0.0
sap-ep-version	7.00.200611242009
sap-ext-sid	CSVWdVI2N/++AUMy2XMbPQ==Oer8MJjvXb+h9tqRks5lNA==
sap-locale	en
sap-rtl	
sap-tray-padding	X
sap-tray-type	PLAIN
sap-wd-cltwndid	WID1172135556832
sap-wd-pb-debugmode	true
sap-wd-tstamp	1172136216151

As all these parameters are passed as standard URL parameters you can access them easily inside a Web Dynpro application (although there is normally no need doing this, as the Web Dynpro runtime handles the settings defined by the different parameters automatically).

The following code example shows for example how to get the sent theme URL:

```
String themeURL =
      WDProtocolAdapter().
      getProtocolAdapter().
      getRequestObject().
      getParameter("cssurl");
```

Keep in mind that URL parameters are *not* forwarded to all follow-up requests. You have to access these parameters therefore in the initial request (typically in one of the `wdDoInit()`methods of a component or custom controller).

In the `ParametersApp` example application we copy the list of parameters to a certain Web Dynpro context node. To make sure that we get all values even in case that there are more than one value for one parameter we use the `WDProtocolAdapter.getProtocolAdapter().getRequestObject()` `.getParameterValues()` method as shown in the following code example.

```
// Store full list of parameters with the inital request
Enumeration parameterNames =
  WDProtocolAdapter.getProtocolAdapter().getRequestObject().getParameterNames();

while (parameterNames.hasMoreElements()) {
  String parameterName = (String) parameterNames.nextElement();
  if (parameterName.equals("eventQueue"))
    continue;

   String values[] = WDProtocolAdapter.getProtocolAdapter().
                       getRequestObject().getParameterValues(parameterName);

   for (int i = 0; i < values.length; i++) {

      IPublicParametersComp.ISourceElement parameter =
         wdContext.nodeSource().createSourceElement();
```



```
      parameter.setName(parameterName);
      parameter.setValue(values[i]);
      wdContext.nodeSource().addElement(parameter);
  }
}
```

Handling of NW04 Web Dynpro iViews

The computed application URL used for a NW04 Web Dynpro iView points directly to the Web Dynpro application as shown above. This URL is the only connection between the SAP NetWeaver Portal and the Web Dynpro runtime. The only way to share information between the SAP NetWeaver Portal and the Web Dynpro runtime is the usage of several parameters sends together with the application URL.

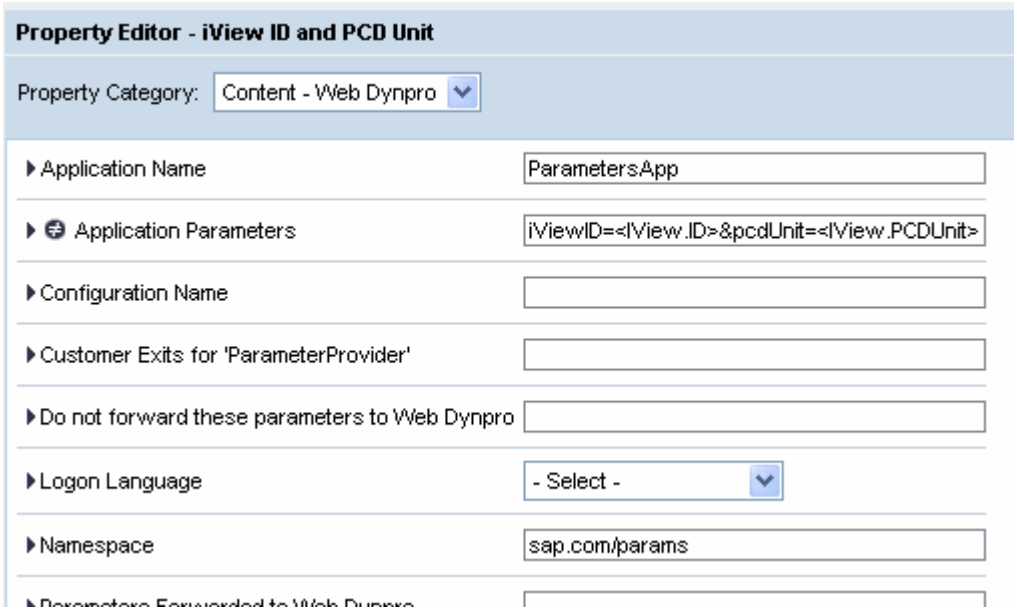
Handling of NW04s Web Dynpro iViews

The main difference between NW04 Web Dynpro iViews and NW04s Web Dynpro iViews is the dependency to the Web Dynpro page builder. NW04s Web Dynpro iViews are handled by the Web Dynpro page builder and the Web Dynpro page builder itself is responsible to launch the associated Web Dynpro application in the end. This is not done using additional HTTP requests but using internal server-side APIs between the Web Dynpro page builder and the Web Dynpro runtime.

The SAP Application Integrator is nevertheless responsible to compute the URL needed to redirect the original iView request to the Web Dynpro runtime. In contradiction to NW04 Web Dynpro iViews this URL points *not* to the launched Web Dynpro application but *always* to the Web Dynpro page builder. Besides that the same portal settings are transferred to the Web Dynpro runtime.

Extending the URL template

For some scenarios it could make sense to extend the used URL template to transfer additional information from the SAP NetWeaver Portal to the Web Dynpro runtime. The SAP Application Integrator provides therefore several *contexts* which could be used to add certain portal settings to the computed application URL. For Web Dynpro iViews you have to define these contexts using the `Application Parameters` iView property of the iView. The following picture shows for example the usage of the `<IView>` context:



The following sub-chapters describe the most important ones.

<IView> context

The `<iView>` context enables you to determine additional information on the Web Dynpro iView and transfer that information to the Web Dynpro application:

- `<IView.ID>`

  As mentioned in Section 5.2.2, all objects that you create for the SAP NetWeaver Portal are stored in the Portal Content Directory (PCD). Each object can be described by a unique path that is also used to specify navigation targets in the portal navigation.

  The `<IView.ID>` variable contains exactly this path for the called Web Dynpro iView. Using this path, you can directly access the iView object in the PCD within your Web Dynpro application.

- `<IView.PCDUnit>`

  The `<IView.PCDUnit>` variable provides access to the roles assigned to the iView within your Web Dynpro application. If the iView that is called has not been assigned to any role, the `<IView.PCDUnit>` variable provides the iView by itself.

The following picture shows the transferred data using the iView located under **SAP Application Integrator -> NW04 Web Dynpro iViews -> iView ID and PCD Unit**.

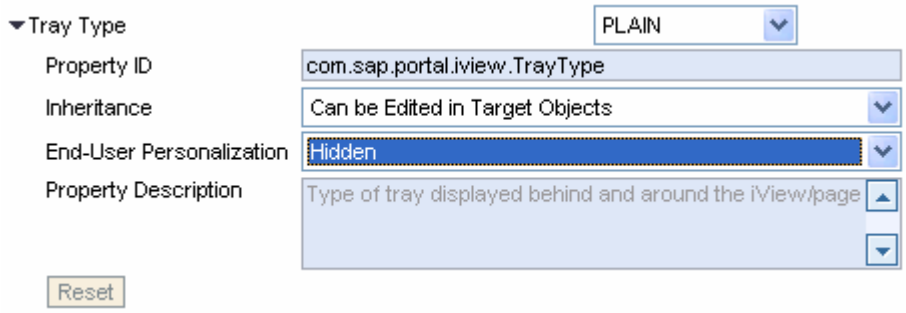
Name	Value
iViewID	pcd:portal_content/sdn/ai/nw04wd/appparamsnw04
pcdUnit	portal_content/sdn/ai/nw04wd/appparamsnw04

<Profile> context

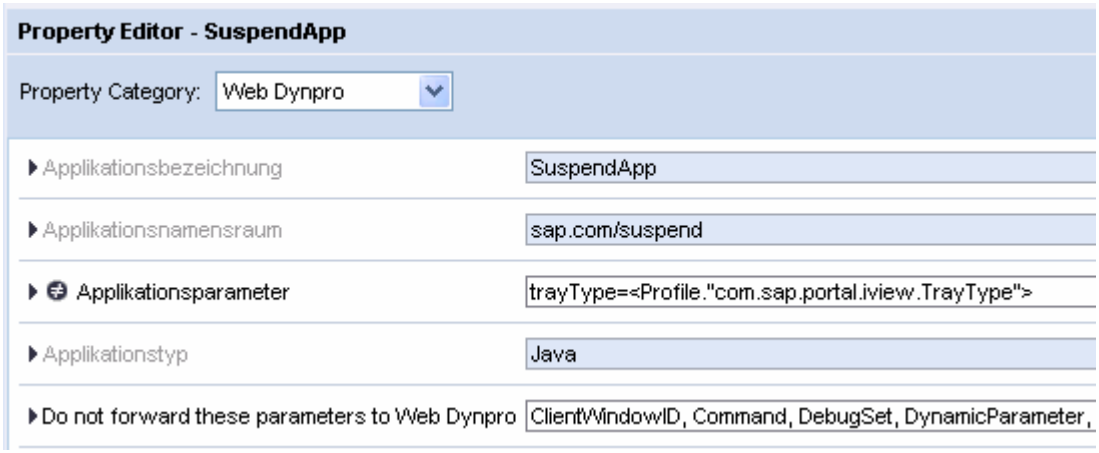
You can use the `<Profile>` context to transfer the values of any iView property to a Web Dynpro application. The `<Profile.xyz>` variable determines the value of the iView property `xyz`.

Please have a look at the following example: we would like to pass the `Tray Type` iView property to a Web Dynpro application. This could be used to adjust the background color of the Web Dynpro application based on the defined tray type of the iView running the application.

As we need the technical ID of the property we have to select this property in the property editor of the related iView. Using the small triangle icon on the left side we could display the *meta attributes* of the property as shown in the next picture.



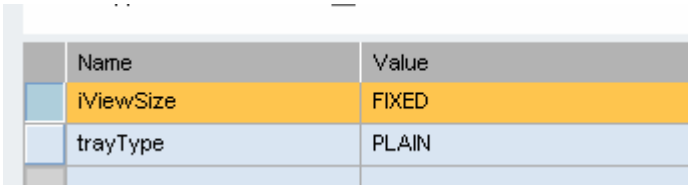
The meta attribute `Property ID` describes the technical name. Because `com.sap.portal.iview.TrayType` contains special characters (`.`), you must specify the URL template variable using the following syntax: `<Profile."com.sap.portal.iview.TrayType">`. By doing this, you can ensure that the URL template will be processed correctly. The following picture shows the definition of the variable using the `Application Parameters` iView property.



You can now access the forwarded `trayType` parameter using the `WDProtocolAdapeter` class as shown in the following code example:

```
String trayType =
      WDProtocolAdapter().
      getProtocolAdapter().
      getRequestObject().
      getParameter("trayType");
```

The following picture shows the transferred data using the iView located under **SAP Application Integrator -> NW04 Web Dynpro iViews -> Profile**. It shows the used tray type and the define iView size.



These parameters are passed using the following `Application Parameters` iView property value.

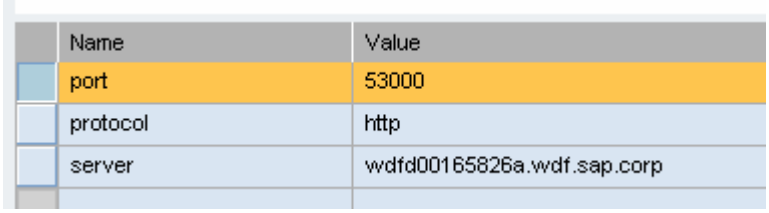
```
trayType=<Profile."com.sap.portal.iview.TrayType">&iViewSize=<Profile."com.sap.portal.iview.HeightType">
```

<Request> context

You can use the `<Request>` context to determine how and where the SAP NetWeaver Portal was launched. This can be particularly useful in scenarios when you don't run your Web Dynpro application on the same SAP NetWeaver installation as the SAP NetWeaver Portal.

`<Request.Protocol>` provides the protocol (HTTP or HTTPS), while `<Request.Server>` provides the fully qualified host name on which the SAP NetWeaver Portal was launched (for example, `portal.domain.com`). You can use the `<Request.Port>` variable to determine the port that is used. In addition to these technical parameters, you can also use the `<Request.xyz>` variable to obtain the value of any parameter that was transferred when the SAP NetWeaver Portal was launched; `<Request.xyz>` provides the value of the `xyz` transfer parameter. If no `xyz` transfer parameter has been defined, the SAP Application Integrator will ignore the variable when processing the URL template.

The following picture shows the transferred data using the iView located under **SAP Application Integrator -> NW04 Web Dynpro iViews -> Request.** It shows the used host, port and protocol.



Name	Value
port	53000
protocol	http
server	wdfd00165826a.wdf.sap.corp

<System> context

As described earlier in this chapter, you use a system alias for each Web Dynpro iView in order to define the SAP NetWeaver installation on which you want to run the Web Dynpro application. The `<System.xyz>` variable of the `<System>` context enables you to determine any property of the system that is used in the iView, which has been launched. `<System.xyz>` determines the value of system property `xyz`.

<User> context

The `<User>` context helps you to transfer the various pieces of user information to the Web Dynpro application. This is especially useful on those occasions when the SAP NetWeaver installation that runs your Web Dynpro application is different from the SAP NetWeaver Portal and does not use the same user store. The `<User>` context defines the variables listed in the following. You can infer the meaning of these variables from their names:

- `<User.displayname>`
- `<User.salutation>`
- `<User.firstname>`
- `<User.lastname>`
- `<User.jobtitle>`
- `<User.department>`
- `<User.telephone>`
- `<User.fax>`
- `<User.streetaddress>`
- `<User.zip>`
- `<User.city>`
- `<User.country>`
- `<User.timezone>`
- `<User.mobile>`

The following picture shows some of the user data using the iView located under **SAP Application Integrator -> NW04 Web Dynpro iViews -> User.**

	Name	Value
	displayName	Smith, John
	email	
	firstName	John
	lastname	Smith
	salutation	

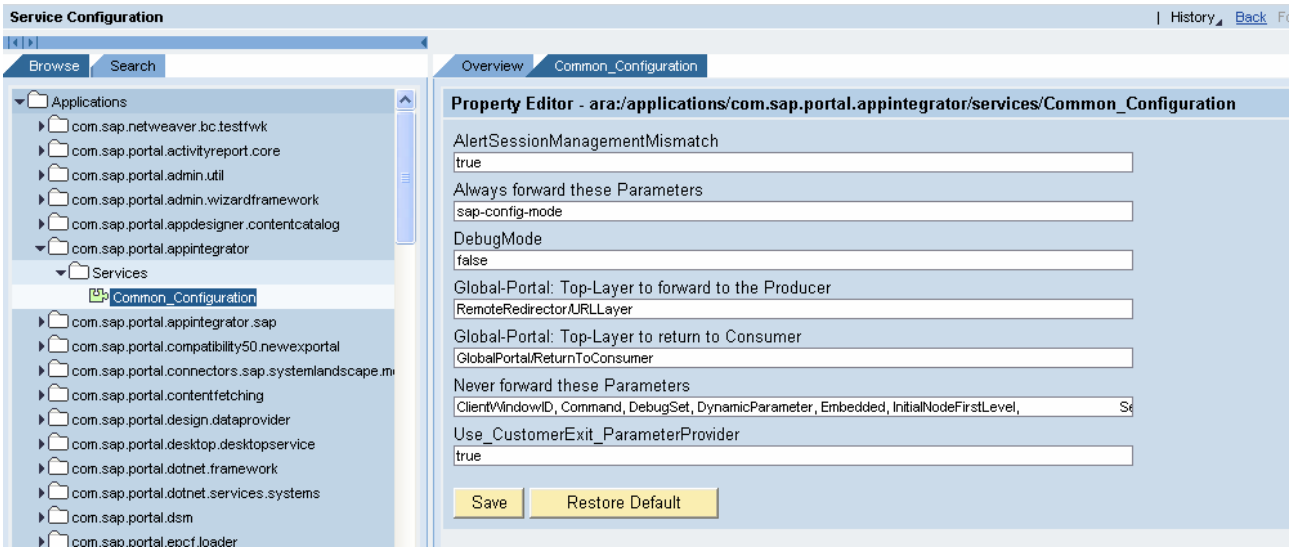
Using customer-specific parameter providers

In the last chapters we discussed how the SAP Application Integrator computes the needed Web Dynpro application URL. We explained the usage of URL templates and gave and overview how the used URL template could be extended to transfer certain data from the SAP NetWeaver Portal to the Web Dynpro runtime.

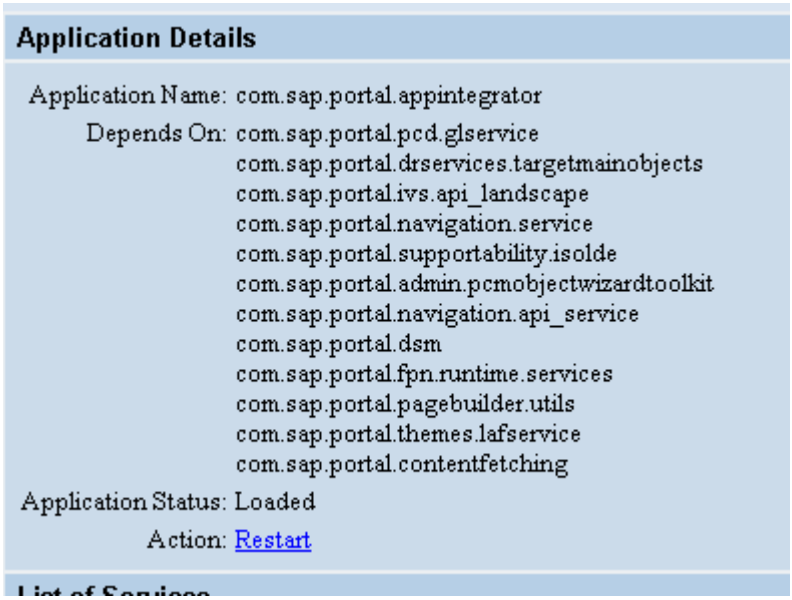
We now will continue with the customer-specific parameter providers, which offer a flexible approach to dynamically compute parameters (and their values).

Prerequisites

To make sure that customer-specific parameter providers are not used “by accident” you have to turn on the general support for this using the **Server Configuration** under **System Administration -> System Configuration**. You have to open the configuration for the `com.sap.portal.appintegrator` service as shown in the next picture using the **Configure** context menu entry of the `Common Configuration` node.



To turn on the support for customer-specific parameter providers you have to set the `Use_CustomerExit_ParameterProvider` parameter to `true`. After changing this value you have to restart the SAP Application Integrator service. To do so you have to open the administration view using the **Administrate** context menu entry of the `com.sap.portal.appintegrator` node. Using the **Restart** link you could restart the service.



After that you could use customer-specific customer-providers.

Limitations

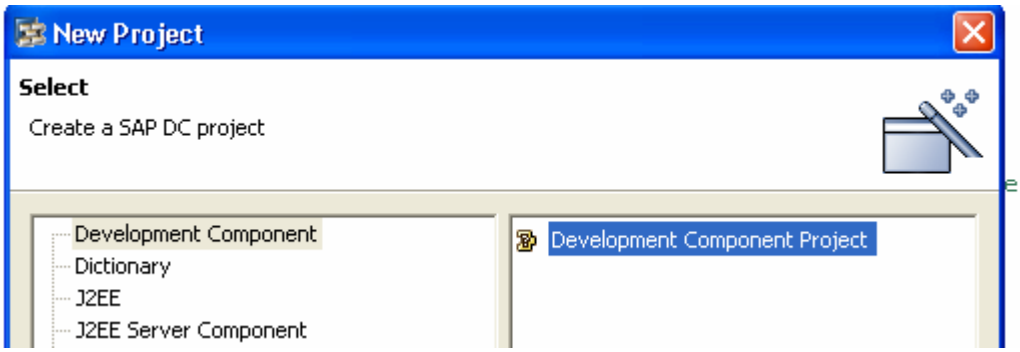
With the current SP of the SAP NetWeaver 04 release the usage of customer-specific parameter provider is only supported for NW04 Web Dynpro iViews. For NW04s Web Dynpro iView we will introduce this in one of the next SPs.

First example

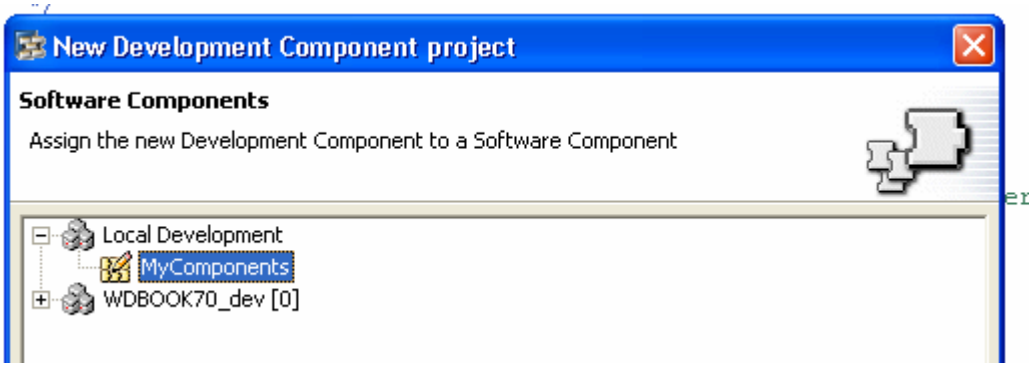
Creating a portal development component

We will create now a first simple example for a customer-specific parameter provider. We will do this using a local portal development component. To create such a development component you have to switch to the `Enterprise Portal Perspective` of the SAP NetWeaver DeveloperStudio as shown in the next picture.

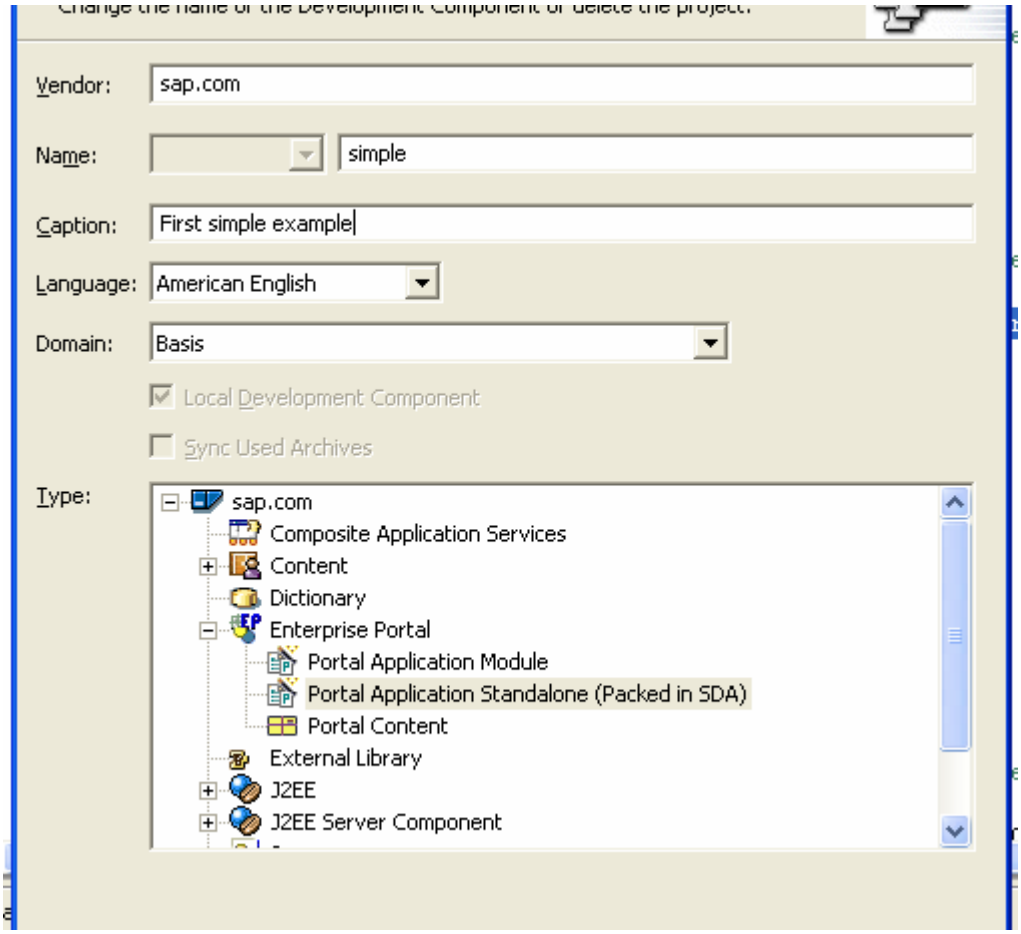
Use **File -> New -> Project** to open the `New Project` dialog. Choose a `Development Component Project` and press the **Next** button at the bottom of the dialog.



On the next screen you have to select whether you want to create a local development component or a development component of a specific development configuration. For our test examples we use local development components (as this does not require a full *NetWeaver Development Infrastructure (NWDI)* installation). Choose **Local Development -> MyComponents** and go to the next dialog step pressing the **Next** button.



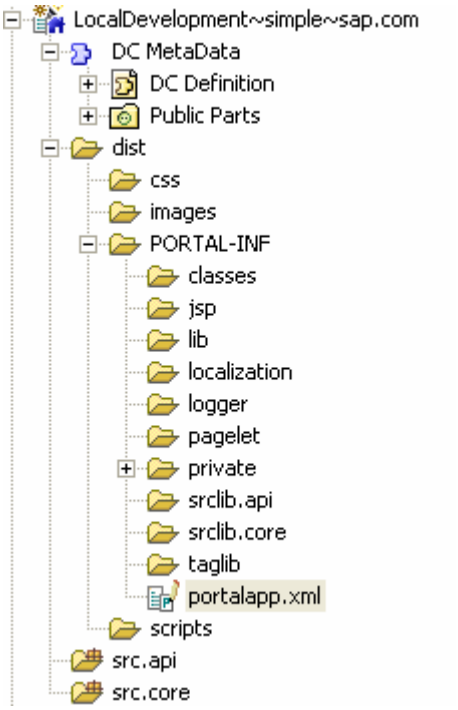
Now you have to define the protect name, an (optional) caption text and the type of the development component. We choose the `Portal Application Standalone (Packed as SDA)` type as shown in the next picture.



The last dialog step does not need any further input and after pressing the **Finish** button the `simple` development component (and the associated `LocalDevelopment~simple~sap.com` project) is created automatically. The next screen shows the created project structure. Most of the shown folders are empty at the beginning.

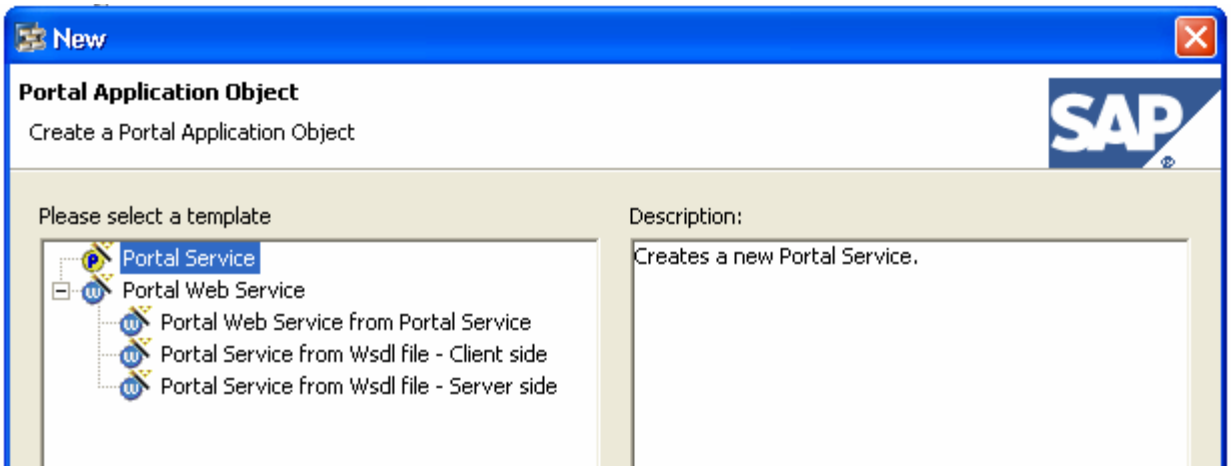
Creating a portal service

A customer-specific parameter provider is from a technical point of view implemented as a standard portal service. Therefore we have to create now such a portal service. The easiest way to do this is to open the `portalapp.xml` editor by double-clickling the `portalapp.xml` entry under **dist -> PORTAL-INF**.

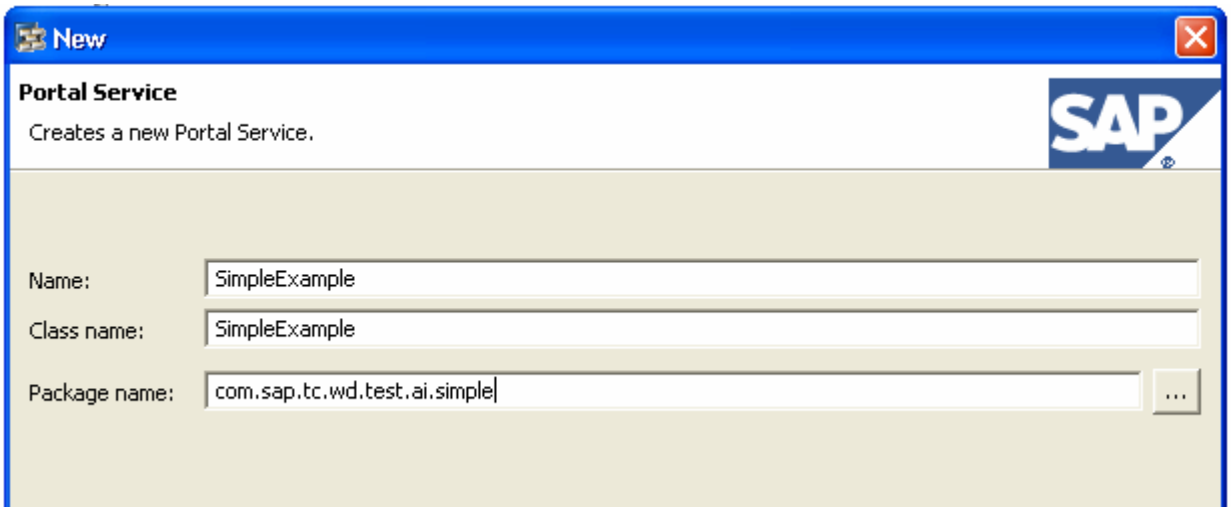


This editor offers several tabs. Please choose the **Services** tab and create a new portal service using the **Create …** button at the right top side of the editor.

The next picture shows the dialog, which allows you to create any kind of portal application. We would like to create portal service; therefore we select `Portal Service` and switch to the dialog step by using the **Next** button.



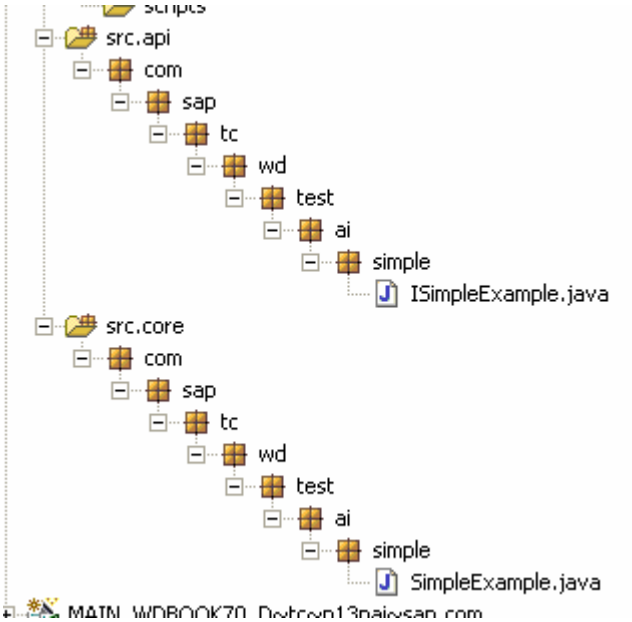
We have to define now the name, the class name and the package name of the portal service. The next picture shows the chosen values. We use the same entry for both the name and the class name.



After pressing the **Finish** button the portal service is created. There are three important parts are created. First the `portalapp.xml` is extended as shown in the following (you can see this by choosing the **Source** tab of the `portalapp.xml` editor.

```
<application>
  <application-config/>
  <components/>
  <services>
    <service name="SimpleExample">
      <service-config>
        <property name="className"
                  value="com.sap.tc.wd.test.ai.simple.SimpleExample"/>
      </service-config>
    </service>
  </services>
</application>
```

Besides that two Java files are created. First the service interface called `ISimpleExample.java` and second the service implementation called `SimpleExample.java` as shown in the next picture.



Implementing the customer-specific parameter provider

The only change needed in the `ISimpleExample.java` file is the definition of the service key as shown in the next code example.

```
import com.sapportals.portal.prt.service.IService;

public interface ISimpleExample extends IService
{
    public static final String KEY = "SimpleExample";
}
```

We use the service name also as key. This is not mandatory but it makes things easier as we do not have to keep in mind different values. Before we describe the needed extension in the `ISimpleExample.java` file we would like to discuss the needed extension of the `portalapp.xml`. The following example shows the extended `portalapp.xml` (the bold entries are added).

```
<application>
  <registry>
    <entry  path="/com.sap.portal.appintegrator/customer_exits/
                  parameter_provider/SimpleExample"
     name="SimpleExample" type="service"/>
  </registry>
  <application-config>
    <property name="SharingReference" value="com.sap.portal.appintegrator"/>
  </application-config>
  <components/>
```



```
  <services>
    <service name="SimpleExample">
      <service-config>
        <property name="className"
                  value="com.sap.tc.wd.test.ai.simple.SimpleExample"/>
        <property name="SafetyLevel" value="no_safety"/>
        <property name="startup" value="true"/>
      </service-config>
    </service>
  </services>
</application>
```

The most important extensions are the definition of the registry of the portal service (doing this the SAP Application Integrator could access later on this portal service) and the definition of a sharing reference to the `com.sap.portal.appintegrator` portal application (needed to run in the end the portal service correctly).

Extending the service implementation

Although a customer-specific parameter provider is from a technical point of view a standard portal service there are some extensions needed comparing to a standard portal service. The most important on is the fact, that the portal service implementation class must implement besides the `ISimpleExample` interface the `com.sapportals.portal.appintegrator.parameter.ICustomerParameterProvider` interface.

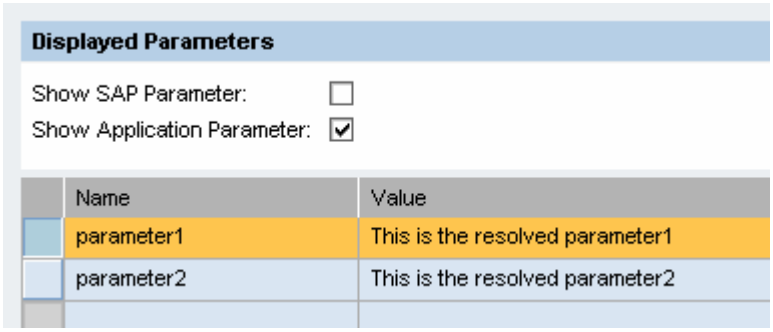
The most important method if this interface is the `getParameter()` method. The following code example shows the implementation for our first simple example:

```
if ("parameter1".equals(key)) {
      return "parameter1=This is the resolved parameter1";
}
if ("parameter2".equals(key)) {
      return "parameter2=This is the resolved parameter2";
}
return null;
```

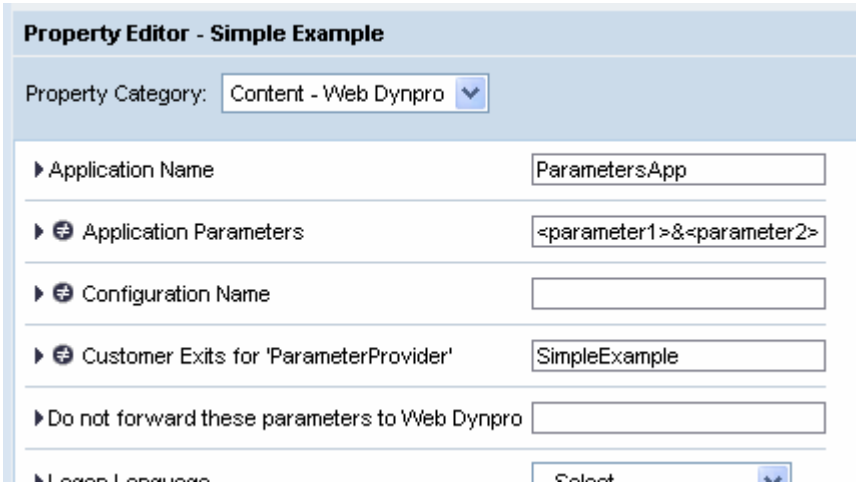
The implementation is quite simple but shows nevertheless the basic idea. The `key` parameter defines the requested parameter. Depending on the current value we return a string value defining at least one name/value pair.

The SAP Application Integrator adds this string to the computed Web Dynpro application URL. In the example above there are in the end two parameters `parameter1` and `parameter2` added and accessible within the called Web Dynpro application.

The next picture shows the `ParameterApp` example application displaying the mentioned parameters.



The usage of a customer-specific parameter provider is really simple. The only thing to do is to define for the needed iViews instance the used customer-specific parameter provider. This is done using the `Customer Exits for 'ParameterProvder'` iView property. The parameter provider is called during the URL computation as soon as you add the needed variables to the `Application Parameters` iView property. The next picture shows the usage of the `SimpleExample` parameter provider within a NW04 Web Dynpro iView.



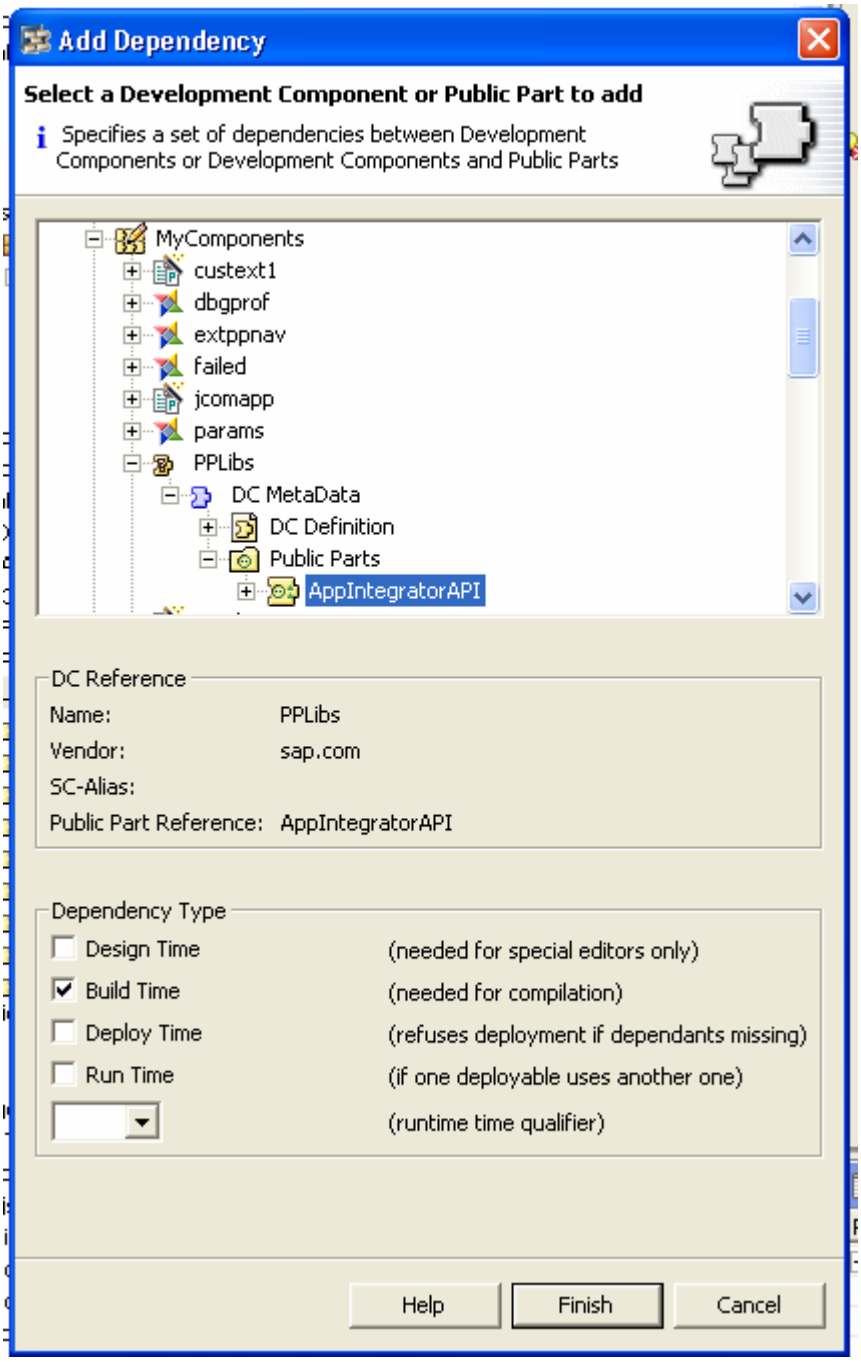
In general it is also possible to use more than one customer-specific parameter provider. You have to define all needed parameter providers using the `Customer Exits for 'ParameterProvder'` iView property. Separate the different parameter provider names using commas.

The `getParameter()` method is called for all parameters added to the computed Web Dynpro application URL. You could therefore also use this method to change one of the technical parameters like the application name or the namespace of the Web Dynpro iView. We will describe this in more detail later one.

Besides the `getParameter()` method you should implement at least the `getProviderName()` method. You should return the same value as defined as service key here. The `getAllParameterNames()` and `getParameterDefault()` methods are optional and are especially helpful during debugging.

To get it compiled

As mentioned above your portal service has to implement the `com.sapportals.portal.appintegrator.parameter.ICustomerParameterProvider` interface. We would like to mention that the needed JAR archives are not automatically available for a local portal development component. One way to access them is to define a certain external library development component containing the needed JARs. Doing this you could access these JARs using a certain public part as shown in the next picture.



To get more information about the definition and usage of library development components please have a look to help.sap.com or sdn.sap.com.

Web Dynpro related scenarios

After the general description of a simple parameter provider we will now discuss several usages of a parameter provider which are especially useful when running Web Dynpro iViews.

Each of the following parameter provider are implemented as the discussed simple example as local portal development components. You will find a link to the archives containing the local development components in each of the following chapters.

Dynamic computation of system or Web Dynpro application

The `getParameter()` method of the parameter provider interface is called for each parameter, which is resolved during the application URL computation. Therefore you could also change predefined iView properties like the `System`, `WebDynproNamespace` or `WebDynproApplication` properties used for a NW04 Web Dynpro iView to define the host running the Web Dynpro application and the Web Dynpro application itself.

The `system` parameter provider allows you to define a mapping to another system. The mapping could be based on any logic. Possible scenarios are for example a system mapping depending on the current user (user with last name starting with “A-P” is running on system A and users for “Q-Z” are running on system B for example). Another example could be the mapping based on the current location. The system parameter provider uses only a very simple `getParameter()` method.

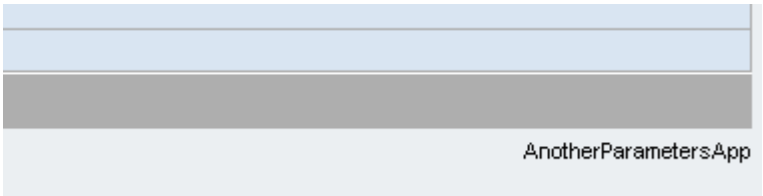
```
// Feel free to add here any logic you want
if ("System".equals(parameter)) {
  return "WebDynproSystem1";
}
return null;
```

Important to mention here is the fact, that for all default iView properties the return value is only the value and not a complete name/value pair.

The `appmapp` parameter provider is an example how to map the application namespace and / or the application name of the executed Web Dynpro application. The `getParameter()` method is also a very simple implementation but you could use here any meaningful logic.

```
// Feel free to add here any logic you want !!!
if ("WebDynproApplication".equals(parameter)) {
  return "AnotherParametersApp";
}
return null;
```

To demonstrate the application mapping we provide a second example application named `AnotherParameterApp`. We following picture shows the running application, displaying the current application name. As soon as you start the iView located under **SAP Application Integrator -> NW04 Web Dynpro iViews -> Application Mapping** you get the mapped application name as shown in the next picture.



Forwarding portal session content

Especially in scenarios where Web Dynpro and non Web Dynpro iView run together within the SAP NetWeaver Portal there could be the need to access certain values stored in the portal session within a Web

Dynpro application. The `session` parameter provider shows how to achieve this. It supports two usages. You can specify *explicitly* which attributes stored in the portal session should be forwarded to the Web Dynpro application. To do this you have to define the session attribute using for example `<"Session.parameter1">` to add the `parameter1` attribute to the URL.

The second usage is to pass all attributes stored in the portal session to the Web Dynpro application. For obvious reasons this makes sense only for string attributes. To pass the complete portal session to the Web Dynpro application you have to add `<"Session.all">` to the `Application Parameters` iView property.

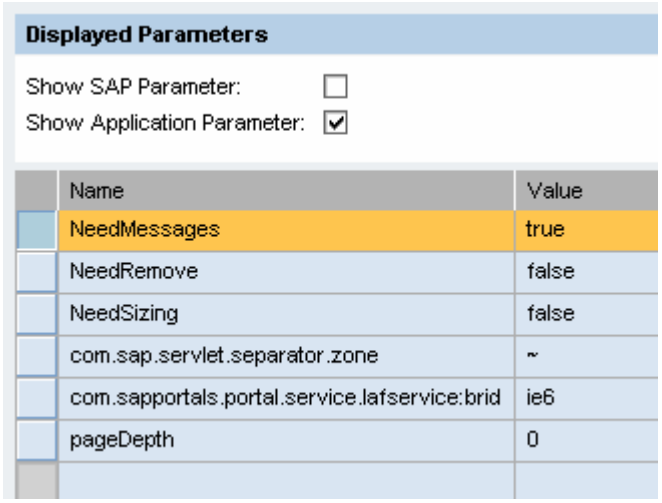
The `getParameter()` method of the `session` parameter provider support both usages:

```
if (parameter != null) {
  if (parameter.startsWith("Session")) {

  String key = parameter.substring(8);
   StringBuffer parameterString = new StringBuffer(1024);
   boolean firstParameter = true;

   if (key.equals("all")) {

     Enumeration attributes =
       request.getServletRequest().getAttributeNames();

    while (attributes.hasMoreElements()) {
      String parameterKey = (String) attributes.nextElement();

      if (!firstParameter) {
        parameterString.append("&");
      } else {
        firstParameter = false;
      }

     parameterString.append(parameterKey).append("=").append(

     request.getServletRequest().getAttribute(parameterKey).toString());

  }
  return parameterString.toString();
} else {
 object object =
   request.getServletRequest().getSession().getAttribute(parameter);

if (object == null) {
```



```
        return parameter + "=Object not found";
      } else {
        return parameter + "=" + object.toString();
      }
    }
  }
}
return null;
```

The following picture shows the iView located under **SAP Application Integrator -> NW04 Web Dynpro iViews -> Complete Portal Session**.



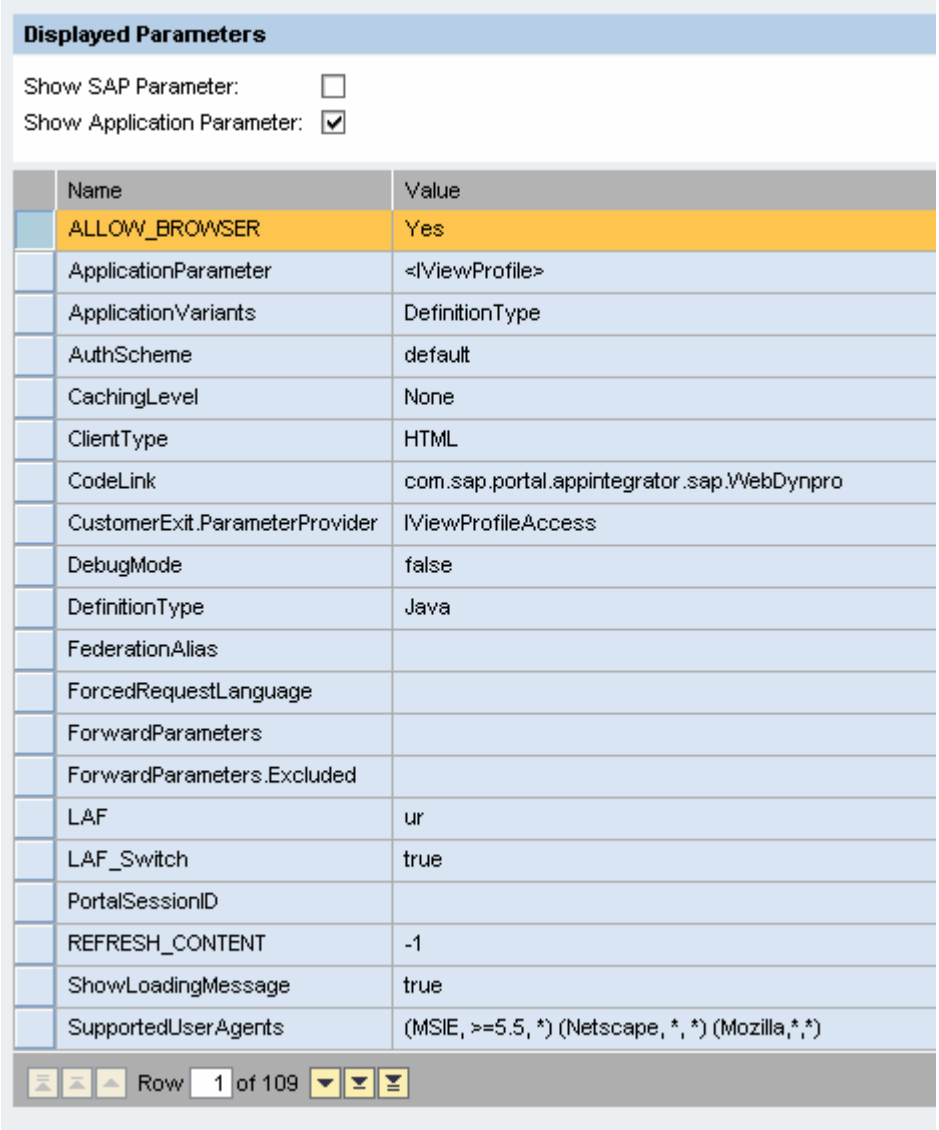
Forwarding the iView profile

Another usage of a customer-specific parameter provider is the access to the complete iView profile within a Web Dynpro application. Using the <Profile.xyz> context you could add specific profile properties to the computed application URL. Using the profile parameter provider you can add the complete profile to the application URL. The getParameter() method of the parameter provider accesses the profile, loops the list of properties and creates one string containing all properties as name / value pairs.

```
  if ("IViewProfile".equals(parameter)) {
    StringBuffer parameterString = new StringBuffer(1024);
    boolean firstParameter = true;

    Enumeration attributeNames =
       request.getComponentContext().getProfile().getProperties();

    while (attributeNames.hasMoreElements()) {
       String attribute = (String) attributeNames.nextElement();
```

```
        if (!firstParameter) {
          parameterString.append("&");
        } else {
          firstParameter = false;
        }

      parameterString.append(attribute).append("=")
        .append(request.getComponentContext().
                  getProfile().getProperty(attribute));
    }

    return parameterString.toString();
  }
  return null;
```

The following picture shows the iView located under **SAP Application Integrator -> NW04 Web Dynpro iViews -> iView Profile**. I contain all properties of the profile of the current iView.



Displayed Parameters

Show SAP Parameter: ☐
Show Application Parameter: ☑

Name	Value
ALLOW_BROWSER	Yes
ApplicationParameter	<IViewProfile>
ApplicationVariants	DefinitionType
AuthScheme	default
CachingLevel	None
ClientType	HTML
CodeLink	com.sap.portal.appintegrator.sap.WebDynpro
CustomerExit.ParameterProvider	IViewProfileAccess
DebugMode	false
DefinitionType	Java
FederationAlias	
ForcedRequestLanguage	
ForwardParameters	
ForwardParameters.Excluded	
LAF	ur
LAF_Switch	true
PortalSessionID	
REFRESH_CONTENT	-1
ShowLoadingMessage	true
SupportedUserAgents	(MSIE, >=5.5, *) (Netscape, *, *) (Mozilla,*,*)

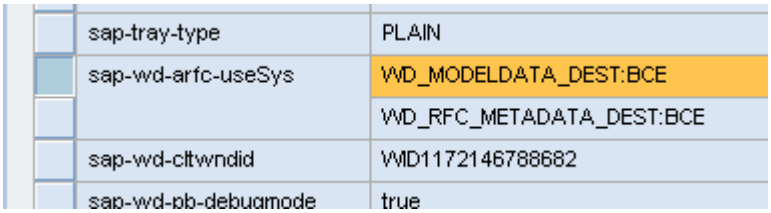
Row 1 of 109

Mapping of JCo destinations

The last example how to use a customer-specifc parameter provider is the definition of JCO destination mappings which is done using the `sap-wd-arfc-useSys` URL parameter. In the past we got a lot of questions regarding this. The `jcomapp` parameter provider provides an easier (and less error-prone) way to define the needed mappings.

To define a JCO mapping you have to add for example `<"WDJCODest:WD_MODELDATA_DEST :WD_RFC_METADATA_DEST:BCE">` to the `Application Parameters` iView property. This defines a mapping of the `WD_MODELDATA_DEST` and `WD_RFC_METADATA_DEST` JCo destinations.

The following picture shows the iView located under **SAP Application Integrator -> NW04 Web Dynpro iViews -> JCo Destination Mapping**.



sap-tray-type	PLAIN
sap-wd-arfc-useSys	WD_MODELDATA_DEST:BCE
	WD_RFC_METADATA_DEST:BCE
sap-wd-cltwndid	WD1172146788682
sap-wd-pb-debugmode	true

The `getParameter()` method creates the needed values of the `sap-wd-arfc-useSys` URL parameter automatically.

```
if (parameterID != null && parameterID.startsWith("WDJCODest:")) {
  StringTokenizer scanSysMap = new StringTokenizer(parameterID, ":");

  String dummy = scanSysMap.nextToken();
   String from1 = scanSysMap.nextToken();
   String from2 = scanSysMap.nextToken();
   String to = scanSysMap.nextToken();

   return "sap-wd-arfc-useSys="
     + from1 + ":" + to + "&sap-wd-arfc-useSys="
     + from2 + ":" + to;
   }
 return null;
```


Copyright