How to realize suspendable Web Dynpro Java applications within the SAP NetWeaver Portal

Applies to:

This document and the presented code examples rely on the SAP NetWeaver 04s release.

Summary

To provide a flexible mechanism to switch between Web Dynpro applications and non Web Dynpro applications there is the Suspend-Resume mechanism as part of the Web Dynpro programming model. When running Web Dynpro applications as Web Dynpro iViews within the SAP NetWeaver Portal the standard Suspend-Resume mechanism does not work. The goal of this document is to describe the various options to provide a working Suspend-Resume mechanism also within the SAP NetWeaver Portal.

Author(s): Jochen Guertler

Company: SAP AG

Created on: 1st February 2007

Author Bio

Jochen Guertler works as a development architect within the NetWeaver UI 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

How to realize suspendable Web Dynpro applications within the SAP NetWeaver Portal 1
Applies to: 1
Summary.. 1
Author Bio 1
Table of Contents 2
Introduction 2
Prerequisites... 2
Using the examples 2
  Deploying the examples.. 3
  Deploying the portal content 3
Navigation inside the SAP NetWeaver Portal... 3
Suspendable Web Dynpro applications.. 4
  Suspending a Web Dynpro application.. 4
  Resuming a suspended Web Dynpro application.. 4
  Behavior of the Web Dynpro application during the Suspend state 5
  Expose more than one iView from one Web Dynpro application 6
Back navigation 10
  Generic usage of the portal navigation history.. 10
    Benefits and restrictions 11
  Explicit definition of the iView path... 12
    Getting the iView path 12
Summary... 13
Copyright... 14

Introduction

To provide a flexible mechanism to switch between Web Dynpro applications and non Web Dynpro applications there is the Suspend-Resume mechanism as part of the Web Dynpro programming model. When running Web Dynpro applications as Web Dynpro iViews within the SAP NetWeaver Portal the standard Suspend-Resume mechanism does not work.

The goal of this document is to describe the various options to provide a working Suspend-Resume mechanism also within the SAP NetWeaver Portal.

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 aspects of the Suspend-Resume mechanism we provide two running example applications: `SuspendApp` and `TargetApp`. The `SuspendApp` application is a suspendable application, which also allows navigating to another iView. The `TargetApp` application is used as navigation target and shows for example how to navigate back to the originator of a navigation step.

Besides these two example applications there is the `TwoIViewsApp` example application, demonstrating the combination of Suspend-Resume and the possibility to expose more than one iView from one Web Dynpro application.

Deploying the examples

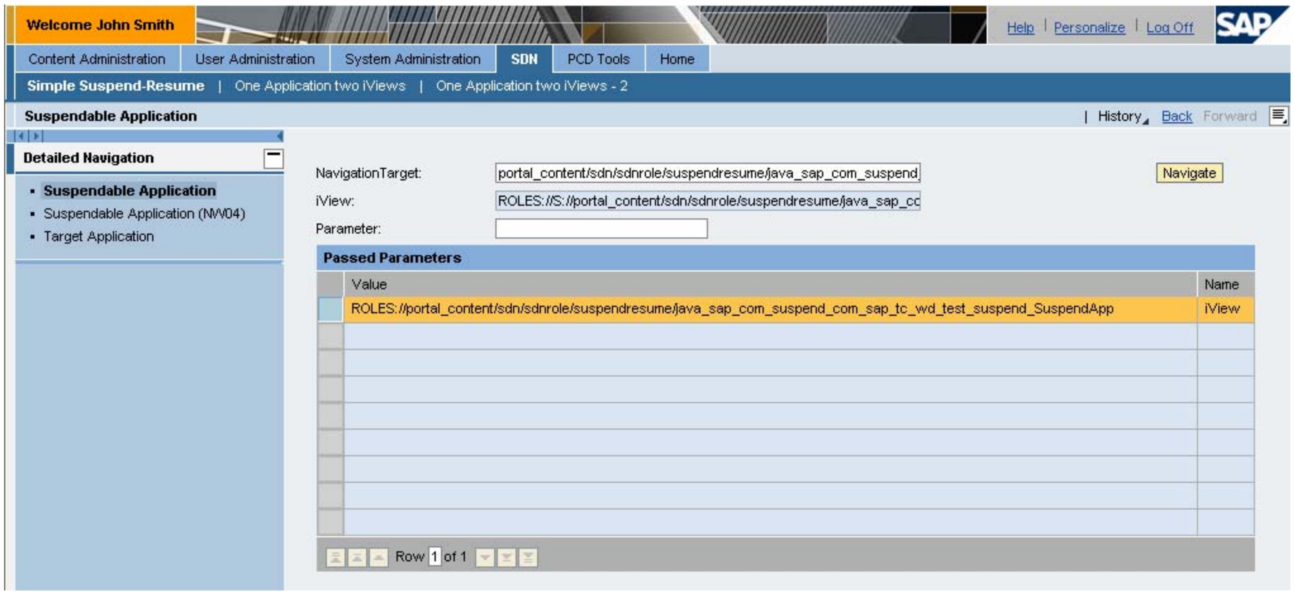
Both example applications are part of the `suspend` Web Dynpro development component. To deploy this development component you have to import it as a local development component into your NetWeaver Developers Studio. You can find the zipped development component here.

Deploying the portal content

After deploying the `suspend` 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 `SDN` role under **User Administration -> Identity Management** to your user.

The following screenshot shows the provided example applications running in the SAP NetWeaver Portal.



NavigationTarget: portal_content/sdn/sdnrole/suspendresume/java_sap_com_suspend

iView: ROLES://S://portal_content/sdn/sdnrole/suspendresume/java_sap_cc

Parameter:

Passed Parameters

Value	Name
ROLES://portal_content/sdn/sdnrole/suspendresume/java_sap_com_suspend_com_sap_tc_wd_test_suspend_SuspendApp	iView

Navigation inside the SAP NetWeaver Portal

As soon as you run a Web Dynpro application as a Web Dynpro iView within the SAP NetWeaver Portal you have additional capabilities to navigate between different Web Dynpro applications.

First there are the *Top Level Navigation (TLN)* and *Detailed Navigation (DTN)* components provided by the portal UI. Using these two components a user could navigate to any (Web Dynpro or nor non Web Dynpro) iView, visible in the user-specific navigation structure.

The second option is the programmatically navigation using the `WDPortalNavigation` class, which allows both absolute / relative navigation to any iView or page and the object-based navigation (OBN) to trigger and navigation to a specific operation of a business object. The specified navigation target is not necessarily visible in the TLN or DTN components.

Independently of the used navigation mechanism the portal is responsible to handle the sessions of the called applications. The standard behavior is to destroy an application session as soon as the user navigates to another application (i.e. iView). Navigating back to the same application would mean in this scenario on the other side, that the iView (i.e. the associated application) is restarted.

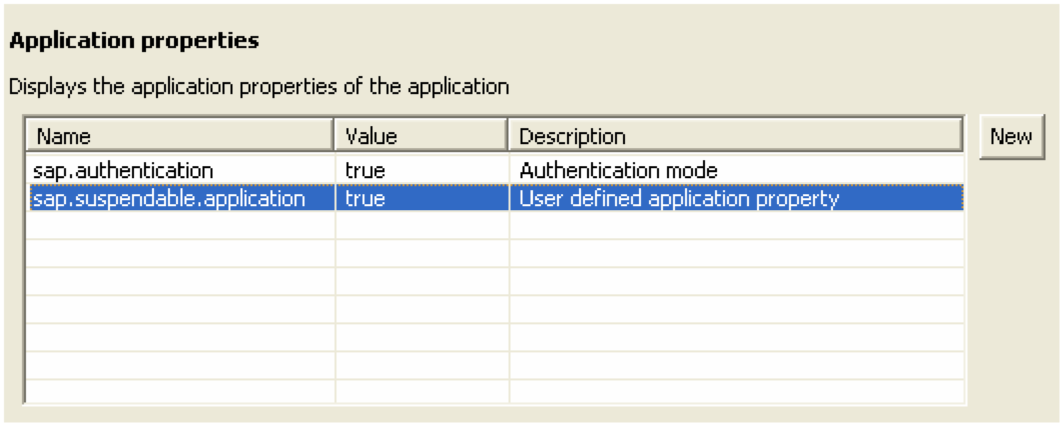
In general this behavior is correct for most of the scenarios, but what happens if the Web Dynpro application should not be destroyed but "kept alive" as long as the user navigates back? How does the portal know in this case whether the application should be destroyed or only suspended?

Suspendable Web Dynpro applications

In general there is no generic mechanism or heuristic which allows the SAP NetWeaver Portal to find out whether an application should be destroyed or suspended as soon as the user navigates away.

The only way to define this is to mark a Web Dynpro application explicitly as *suspendable*. To do this you have to define a specific application property named `sap.suspendable.application`. Possible values are `true` or `false`. Not defining this parameter means that the application is *not* suspendable.

The following picture shows the definition of this application property for the `SuspendApp` example application.



Application properties

Displays the application properties of the application

Name	Value	Description
sap.authentication	true	Authentication mode
sap.suspendable.application	true	User defined application property

Suspending a Web Dynpro application

As soon as a Web Dynpro application runs as an iView within the SAP NetWeaver Portal there is no need to fire the suspend plug to suspend an application (as this is needed for the standalone scenario).

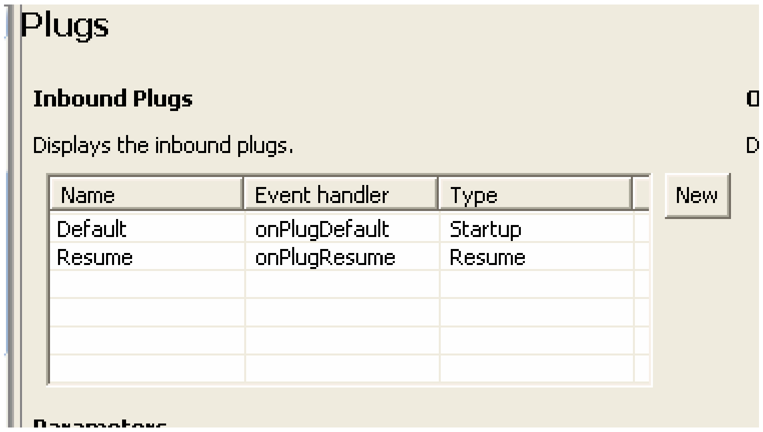
The application is suspended automatically as soon as the user navigates away to anther iView. The portal session management informs the Web Dynpro runtime in this case that the application should *not* be destroyed but suspended.

Using the `wdDoApplicationStateChange()` method, which we will describe later, is it possible to ensure that an suspended application does not allocate to much memory or lock any backend connection.

Resuming a suspended Web Dynpro application

Although there is no need to implement a specific suspend plug when running as a Web Dynpro application as Web Dynpro iView, it could be useful to define a resume plug even in this scenario. The resume plug allows you to restore the session state of a suspended application.

A resume plug is defined as special inbound plug of the interface view controller of the root component of the suspended application. The following picture shows the definition of the `Resume` resume plug for the `SuspendApp` example application.



Displays the inbound plugs.

Name	Event handler	Type
Default	onPlugDefault	Startup
Resume	onPlugResume	Resume

Behavior of the Web Dynpro application during the Suspend state

As soon as a “suspendable” Web Dynpro application runs as an iView in the SAP NetWeaver Portal, the portal session management informs the Web Dynpro runtime that the application should *not* be destroyed but suspended in case the user navigates away from this iView. The length of the suspend state depends on the interval you set in the application property `ExpirationTime`.

If this duration is exceeded before the Web Dynpro application has been called again (by navigating back to the related iView), the suspended Web Dynpro application is terminated. If the user navigates back to a suspended iView after the associated application was destroyed due to the expiration time, the Web Dynpro application is restarted. If the application defines a resume plug this resume plug is used to restart the application instead of the standard startup plug.

When the resume plug is called in this case, it is possible to restore the state of the Web Dynpro application before it was terminated. To enable this you must obviously store the current state of the Web Dynpro application before it is terminated.

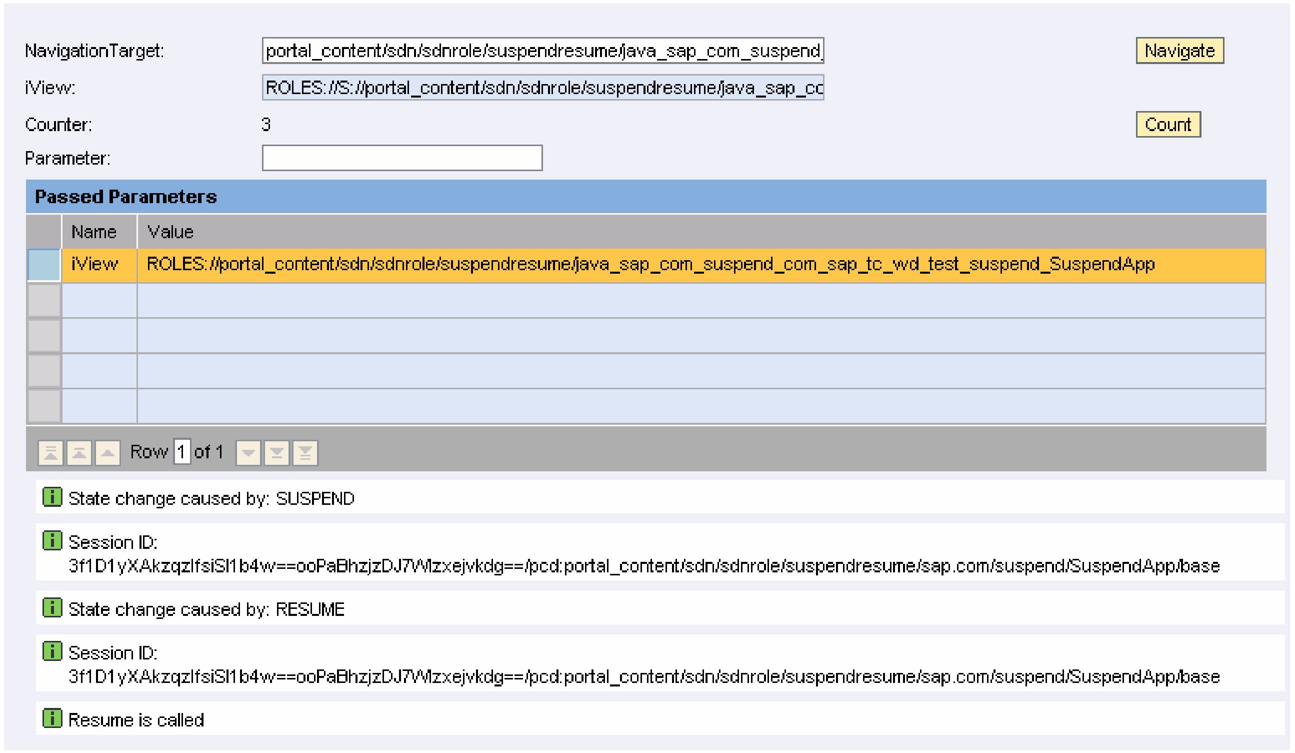
If the Web Dynpro application changes its state, the event `stateChangeEvent` is triggered, which can be queried in the `wdDoApplicationStateChange()` method of the component controller. Here, you can use the `IWDApplicationStateChangeInfo` class to query the status. Three reasons are possible for calling the method:

- `Suspend`: The Web Dynpro application is suspended. In this case you should minimize the storage requirement of the application and make sure that all backend connections are released. The `wdDoApplicationStateChange()` method is called with this reason as soon as the user navigates away to another iView.
- `Resume`: The Web Dynpro application is reactivated. If a resume plug is defined the resume plug is called to resume the application. The `wdDoApplicationStateChange()` method is called with this reason as soon as the user navigates back to an suspended application (iView).
- `Timeout`: The Web Dynpro application is terminated due to a timeout event. After that, the `wdDoExit()` methods of the component and view controllers are called.

The next picture shows the `SuspendApp` example application after navigating to another iView and navigating back to the suspended `SuspendApp` iView.

During the suspending of the application the `wdDoApplicationStateChange()` method was called with the `SUSPEND` reason. During navigating back (and resuming the application) the `wdDoApplicationStateChange()` method is called with the `RESUME` reason. Please check the session IDs – they are the same for both the suspending and resuming.

In the end the `Resume` resume plug is called.



NavigationTarget: portal_content/sdn/sdnrole/suspendresume/java_sap_com_suspend

iView: ROLES://S://portal_content/sdn/sdnrole/suspendresume/java_sap_cc

Counter: 3

Parameter:

Passed Parameters

Name	Value
iView	ROLES://portal_content/sdn/sdnrole/suspendresume/java_sap_com_suspend_com_sap_tc_wd_test_suspend_SuspendApp

Row 1 of 1

State change caused by: SUSPEND

Session ID:
3f1D1yXAkzqzlfsiSl1b4w==ooPaBhzjzDJ7WIzxejvkdg==/pcd:portal_content/sdn/sdnrole/suspendresume/sap.com/suspend/SuspendApp/base

State change caused by: RESUME

Session ID:
3f1D1yXAkzqzlfsiSl1b4w==ooPaBhzjzDJ7WIzxejvkdg==/pcd:portal_content/sdn/sdnrole/suspendresume/sap.com/suspend/SuspendApp/base

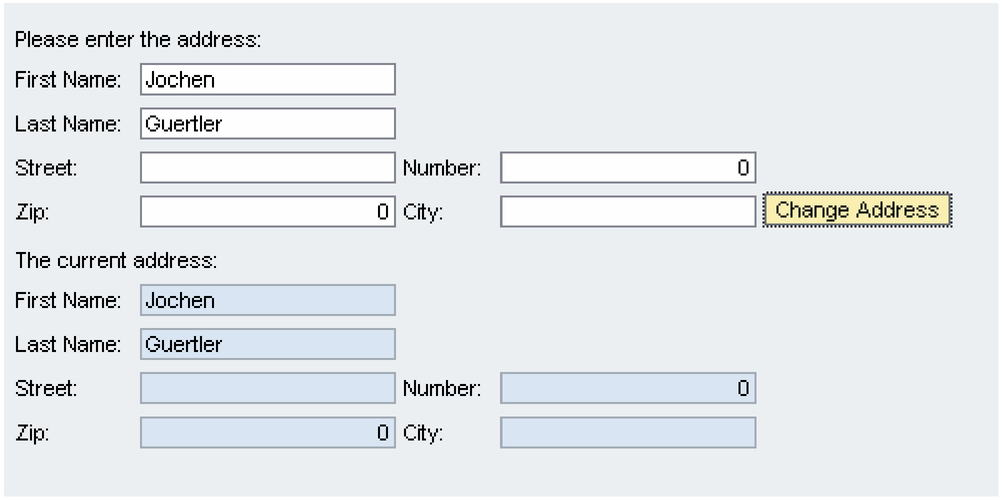
Resume is called

Expose more than one iView from one Web Dynpro application

Before we will discuss the capabilities how to navigate back to a suspended application we would like to mention another interesting usage of suspendable Web Dynpro applications when running as an NW04s Web Dynpro iView.

The first step is to define a Web Dynpro application exposing more than one iView. The `TwoIViewApp` example application shows the details.

It is a simple application allowing you to define an address. After pressing the **Change Address** button the address data is available also in the address display area (as both Web Dynpro views are using the same context mapping). The following picture shows the application running as standalone application.



Please enter the address:

First Name: Jochen

Last Name: Guertler

Street: Number: 0

Zip: 0 City:

The current address:

First Name: Jochen

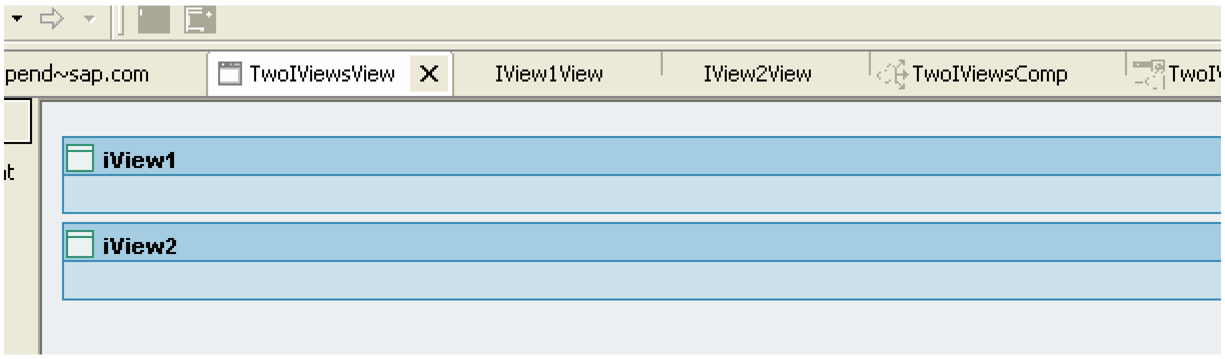
Last Name: Guertler

Street: Number: 0

Zip: 0 City:

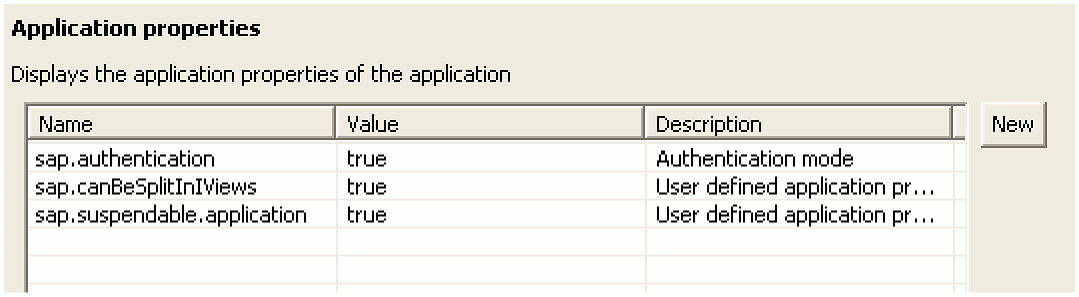
When running within the SAP NetWeaver Portal we would like to expose the two different aspects of the whole application (i.e. the definition of the address and the display of the address) as two different NW04s Web Dynpro iViews.

The list of exposed iViews is defined by certain `ViewContainerUIElement` UI element instance (in the example iView1 and iView2 as shown in the next picture).



Besides that you have to mark the Web Dynpro application as a "spittable" application (i.e. an application exposing more than one iView or in other words an application which could be spilt into different iViews), by defining the `sap.canBeSplitInIViews` application property. As we would like to demonstrate the combination with a suspendable application you have to define also the `sap.suspendable.application` application property.

The following picture shows the defined application properties of the `TwoIViewApp` example application.

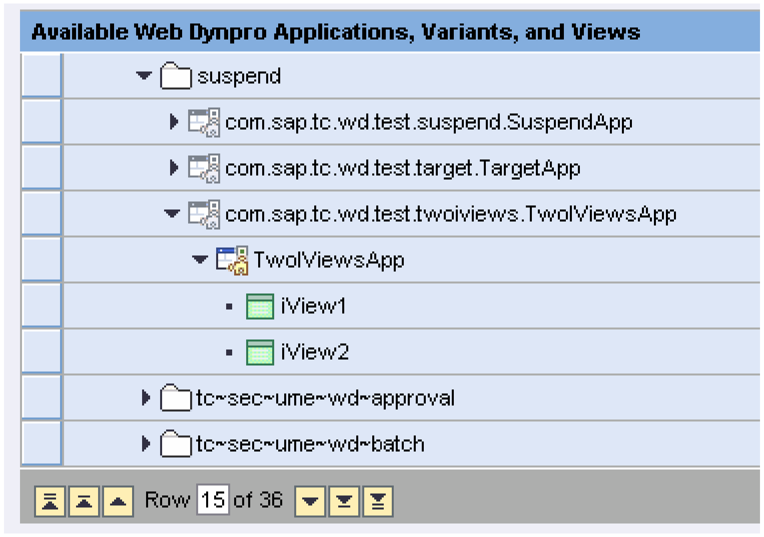


Application properties

Displays the application properties of the application

Name	Value	Description
sap.authentication	true	Authentication mode
sap.canBeSplitInIViews	true	User defined application pr...
sap.suspendable.application	true	User defined application pr...

During creation of the needed Web Dynpro iViews using the Web Dynpro iView wizard you could define for which theoretically exposed iViews you would like to create "real" iViews. The `TwoIViewApp` application exposes two iViews `iView1` and `iView2` as shown in the next picture.

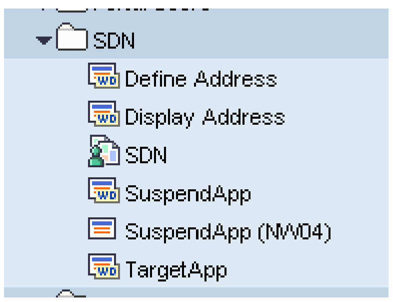


Available Web Dynpro Applications, Variants, and Views

- suspend
  - com.sap.tc.wd.test.suspend.SuspendApp
  - com.sap.tc.wd.test.target.TargetApp
  - com.sap.tc.wd.test.twoiviews.TwoIViewsApp
    - TwoIViewsApp
      - iView1
      - iView2
- tc~sec~ume~wd~approval
- tc~sec~ume~wd~batch

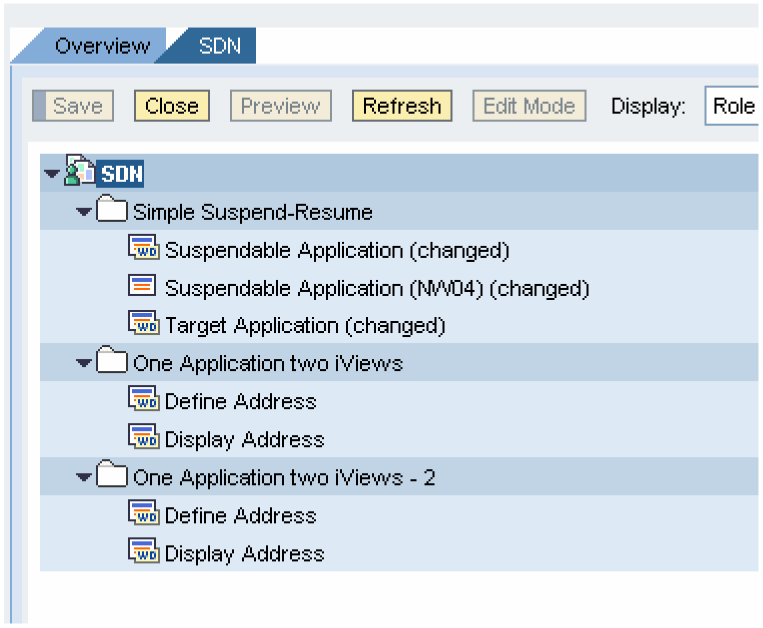
Row 15 of 36

In our example we use both iViews and name them `Define Address` and `Display Address.` The next picture shows the created portal content for the examples used in this document including the two iViews.



- SDN
  - Define Address
  - Display Address
  - SDN
  - SuspendApp
  - SuspendApp (NW04)
  - TargetApp

To use these iViews now we add them using the role editor to the `SDN` role. We create two sub folders `One Application two iViews` and `One Application Two iViews – 2` and add both iViews in both sub-folders. The next picture shows the structure of the `SDN` role.

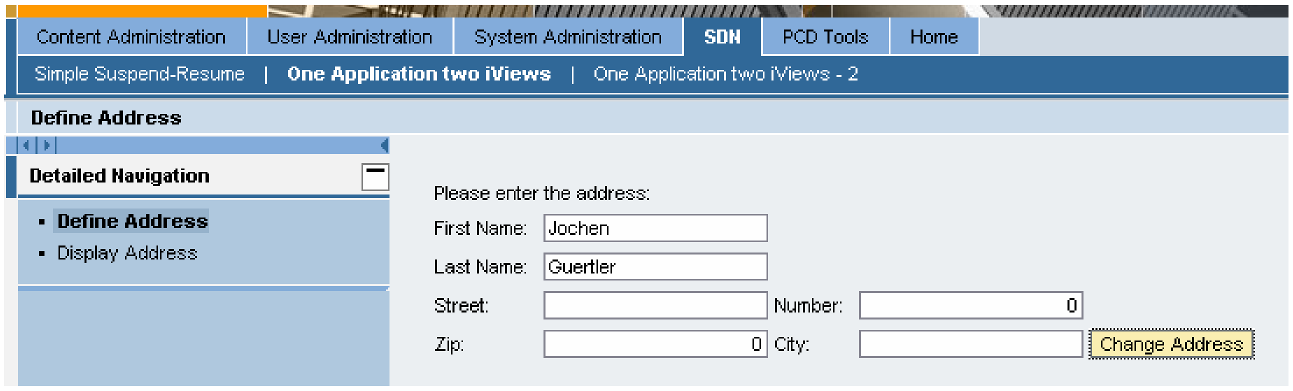


SDN
- Simple Suspend-Resume
  - Suspendable Application (changed)
  - Suspendable Application (NW04) (changed)
  - Target Application (changed)
- One Application two iViews
  - Define Address
  - Display Address
- One Application two iViews - 2
  - Define Address
  - Display Address

How is this now related to our discussions about the capabilities of a suspendable Web Dynpro application?

The answer is easy: as a suspendable application is not destroyed as soon as the user navigates to another iView the user could navigate between the `Define Address` and `Display Adddress` iViews using standard portal navigation capabilities. As on the other side both iViews are exposed by the same `TwoIViewApp` Web Dynpro application and the used application instance is shared as long as such iViews are located under the same PCD location both iViews are running with the same application instance and could share the state (although they are handled as different iViews).

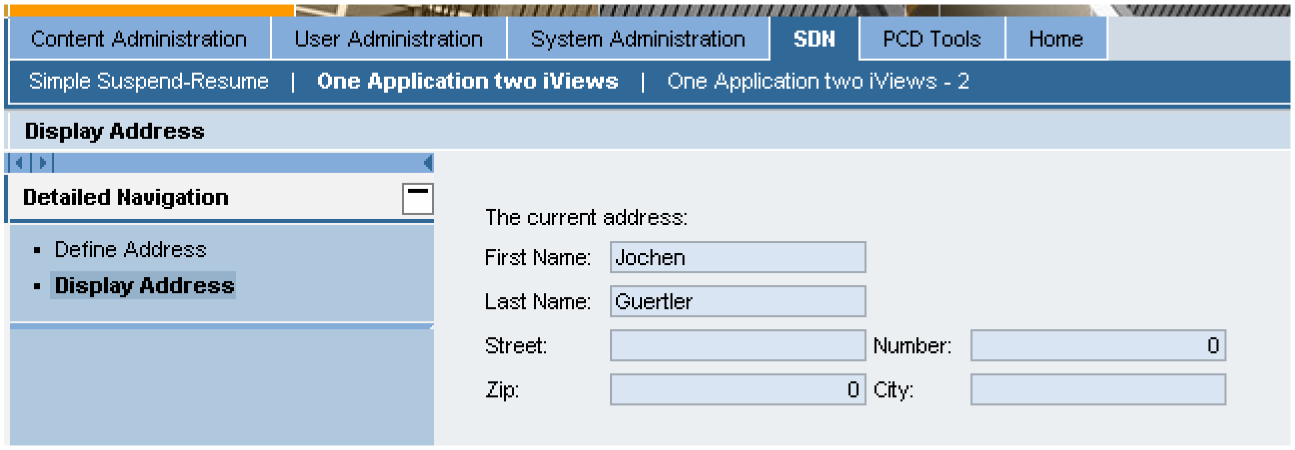
The following picture shows the `Define Address` iView. The user defines the first name and the last name and presses the `Change Address` button.



Please enter the address:

First Name:	Jochen			
Last Name:	Guertler			
Street:		Number:	0	
Zip:	0	City:		Change Address

Now the user navigates to the `Display Address` iView located in the same PCD folder. The next picture shows the same data as defined before. And this data is *not* forwarded using URL parameters as for standard portal navigation but is shared using standard Web Dynpro context sharing on the server side (do not forget that both iViews are sharing the same Web Dynpro application instance).



The current address:

First Name:	Jochen		
Last Name:	Guertler		
Street:		Number:	0
Zip:	0	City:	

It is important for us to mention here that the scope of iViews sharing the same application instance is always the PCD location of the iViews. The iViews located in the `One Application two iViews` folder and the iViews located in the `One Application two iViews – 2` folder do not share the same application instance as these are different folders (i.e. PCD locations).

Besides this restriction we could image a lot of interesting scenarios using suspendable applications exposing more than one iView. You could for example define the visible part of a Web Dynpro application in a role specific way (by adding only some of the exposed iViews to a specific role) without loosing the capability to easily even large amount of data between these iViews. Another example is the navigation between parts of a Web Dynpro application using the object-based navigation which allows you to combine the flexible and role-specific navigation with the tight coupling of Web Dynpro components or views.

We will now continue describing the capabilities to navigate back to a suspended application.

Back navigation

One important aspect in the standalone Suspend-Resume mechanism is the capability to generically navigate back to a suspended application. Or in other words: the application which was the navigation target when suspending the application has to know how to navigate back to the suspended application (and to resume it doing this). In the standalone scenario this is achieved by forwarding a specific application URL pointing to the suspended application. The application URL could be used later one to navigate back to the suspended application.

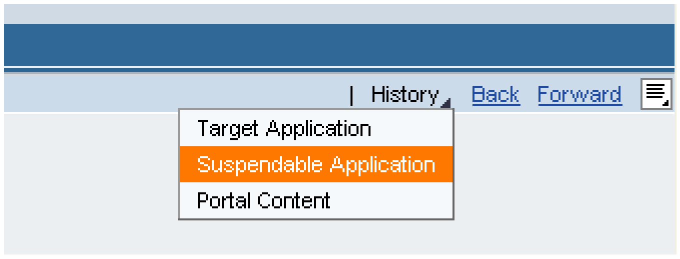
How does this fits to the scenario when running within the SAP NetWeaver Portal?

In general we could think of two possible solutions: first the generic usage of the portal navigation history and second the explicit definition of the iView path.

We will describe both approaches in the next two chapters.

Generic usage of the portal navigation history

The portal navigation history contains the last executed navigation steps. It is visualized as part of the portal page header as shown in the next picture.



The user could use the **Back** and **Forward** links to navigate through the navigation chain stored in the navigation history or could use the **History** drop down box to select explicitly one entry of the navigation history.

As soon as a user uses the TLN or DTN to trigger navigation to another iView, the executed navigation step is automatically added to the portal navigation history. Besides that, it is also possible to add a programmatically triggered navigation step (using the `WDPortalNavigation` class) to the navigation history.

You have to use the `WDPortalNavigationHistoryMode` parameter therefore. The available values are the following:

- `WDPortalNavigationHistoryMode.NO_HISTORY`

  The navigation step is *not* visible in the portal navigation history. You can use the **Back** / **Forward** links to navigate through the navigation chain nevertheless.

- `WDPortalNavigationHistoryMode.NO_DUPLICATIONS`

  The navigation step is visible in the portal navigation history. If there are several navigation steps to the same navigation target there are only visible once in the portal navigation history.

- `WDPortalNavigationHistoryMode.ALLOW_DUPLICATIONS`

  The navigation step is visible in the portal navigation history. If there are several navigation steps to the same navigation destination but with different parameters they are visible as different entries in the navigation history.

  This option makes sense if you navigate to an iView displaying details of a customer for example. The customer ID can be defined using a parameter and doing this several navigation steps to this details iView with different parameters (i.e., different customer IDs) results in different entries in the portal navigation history.

Using the `targetTitle` you can furthermore define the title of the entry in the portal navigation history (e.g. *Details for Customer 4711* and *Details for Customer 007*).

Besides the direct usage of the portal navigation history (using the **Back** / **Forward** links or the **History** drop down box) there is also a programmatically way to trigger a back navigation based on the portal navigation history. The following code example shows the details:

```
WDPortalEventing.fire(
  "urn:com.sapportals:navigation",
  "historyNavigate",
  "<steps>");
```

By firing the `historyNavigate` portal event you can trigger a navigation based on the current portal navigation history within your Web Dynpro application. `<steps>` defines the history entry. Negative entries define a back navigation, positive entries a forward navigation. To navigate back to the last navigation step you have to use `-1` here.

If you use an invalid value for `<steps>` nothing happens.

Benefits and restrictions

The big advantage of this approach is the possibility to generically navigate back without any knowledge about the concrete navigation target. This helps in scenarios, where the different applications / iViews do not know each other.

The main restriction on the other side is the fact, that it is not possible to pass some parameters together with the navigation back to the navigation target. Scenarios like “Overview iView navigates to Creation iView to create new entry” – “After creation of the new entry the Creation iView navigates back to the Overview iView passing the ID of the new entry” are not possible.

Another restriction of course is the limitation, that only navigation steps, which are executed inplace (i.e. without opening a new browser window), are part of the portal navigation history.

Explicit definition of the iView path

To overcome the limitation that it is not possible to pass a certain parameter together with the back navigation you could use another approach, which makes especially sense in case that all involved applications are provided by the same group or the same business case. In these scenarios the iViews knows each other and could explicitly trigger the navigation to a certain navigation target.

The easiest way to achieve this is to implement the back navigation using a standard absolute or relative navigation step. Using one of the `WDPortalNavigation` methods it is possible to pass any parameter to the navigation target.

The iView navigating back to the (probably suspended) iView (i.e. application) defines the concrete iView path to it. To get more flexibility it does make sense *not* to hard code this iView path but to provide the possibility to configure this iView path somewhere.

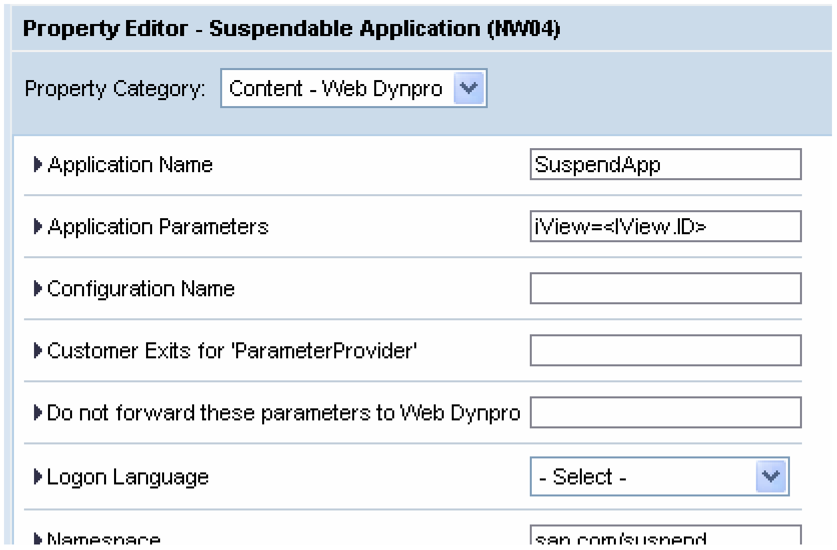
The example applications `SuspendApp` and `TargetApp` use this mechanism.

Getting the iView path

Depending on the type of the used Web Dynpro iView there is a helpful mechanism to get the iView path of the current iView.

In case you are using a Web Dynpro iView based on the Web Dynpro iView template (we name this *NW04 Web Dynpro iView* as this iView type is available since the SAP NetWeaver 04 release) you can define special variables using the `Application Parameters` iView property.

The following screenshot shows the definition of this property for the `SuspendApp` iView.



Property Editor - Suspendable Application (NW04)

Property Category: Content - Web Dynpro

Application Name	SuspendApp
Application Parameters	iView=<IView.ID>
Configuration Name	
Customer Exits for 'ParameterProvider'	
Do not forward these parameters to Web Dynpro	
Logon Language	- Select -
Namespace	sap.com/suspend

The *SAP Application Integrator*, which is in the end responsible to launch the associated Web Dynpro application resolves such variables and adds the result to the URL calling the Web Dynpro application. In the shown example we use the `<IView.ID>` variable to get the iView path of the called iView. The SAP Application Integrator resolves this and forwards it as value for the parameter *iView*.

The `SuspendApp` example applications accesses this parameter in the `wdDoInit()` method using

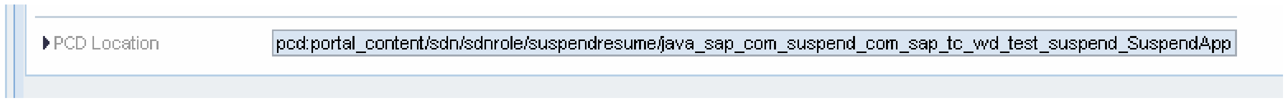
```
    WDProtocolAdapter
       .getProtocolAdapter()
```

```
      .getRequestObject()
      .getParameter("iView");.
```

To get more examples for the usage of the variables resolved by the SAP Application Integrator please has a look to the Maximizing Web Dynpro for Java book.

Using a Web Dynpro iView based on the Web Dynpro page builder (we name this *NW04s Web Dynpro iView* as this is available since the SAP NetWeaver 04s release) the usage of such variables is currently not supported.

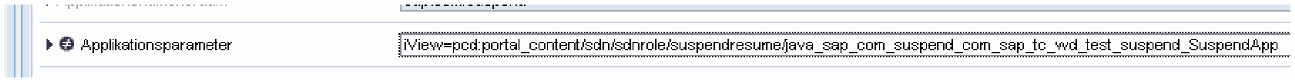
As a fallback you could get the iView path of an iView using the iView editor as shown in the next picture.



PCD Location
pcd:portal_content/sdn/sdnrole/suspendresume/java_sap_com_suspend_com_sap_tc_wd_test_suspend_SuspendApp

The `PCD Location` iView property defines the iView path of an iView.

To forward this to the associated Web Dynpro application you have to define this for NW04s Web Dynpro iViews explicitly using the `Application Parameters` iView property shown in the next picture.



Applikationsparameter
iView=pcd:portal_content/sdn/sdnrole/suspendresume/java_sap_com_suspend_com_sap_tc_wd_test_suspend_SuspendApp

The big disadvantage of this approach is obviously, that you have to maintain this value as soon as you move the iView to another role or another location inside the PCD.

Summary

We discussed the possibilities to run suspendable Web Dynpro applications within the SAP NetWeaver Portal. Besides that we showed the available mechanisms to provide a more or less generic navigation back to a suspended Web Dynpro application.

Copyright