Development Guidelines for Portal Applications Running in FPN Environment

Applies to:

SAP NetWeaver 7.0 and higher, SAP NetWeaver Portal, Federated Portal Network (FPN) For more information, visit the [Portal and Collaboration homepage](#).

Summary

A portal application, which is intended to run in a federated environment, should follow certain guidelines in order to ensure the same behavior in a remote scenario as in a local one.

The purpose of this document is to discuss these guidelines and the technical constraints to be considered when implementing such an application.

Author: Nisim Hafzadi

Company: SAP Labs Israel

Created on: June 2009

Author Bio

Nisim Hafzadi is a software engineer in the SAP NetWeaver Portal group, where his main responsibilities are integrating Web Dynpro for Java applications with the portal. Since joining the SAP Labs Israel in 2002, Nisim has acquired wide knowledge in Java portal development. You can contact him at nisim.hafzadi@sap.com.

Target Audience: developers of portal applications that can run in FPN

Level: intermediate

Knowledge Required: portal application development, Web Dynpro for Java

Table of Contents

Introduction

A federated portal network (FPN) allows organizations with multiple portals, SAP and non-SAP, to share content between the portals. By implementing a federated portal network and sharing content between portals, organizations can provide users at each location with a single portal access point. From each portal configured as an access point, the users are able to access information, services and applications distributed on portals throughout the entire organizational network.

Terminology

Producer: A portal that exposes its content for consumption by other portals in the federated network.

Consumer: A portal that accesses and uses remote content provided by other producer portals in the federated network.

General Guidelines

These guidelines are relevant for both Web Dynpro for Java and PRT applications.

Producer Resources

Bear in mind that all resources, such as services, mime types, handlers, are the producer resources.

Request Object

The Request object in a remote scenario returns the Request object of the producer machine.

Logs and Traces

In a remote scenario, the logs and traces are written in the producer log files.

Portal Services

If your application is using a portal service, be aware that the service is instantiated on the producer.

Potential problem: An application resolves the *navurl* hashed key into a PCD address using the *INavigationService* service. The *INavigationService* service is instantiated on the producer portal, and returns one of the following results:

- If the PCD address of the target application in the consumer is the same as in the producer, the correct PCD address is returned.

- Otherwise, a NULL result is returned, because the producer cannot resolve this hash.

Navigation Structure

When writing an application that uses navigation structure, be aware that the service call to get the navigation structure will return the producer's navigation structure whereas all navigations are performed on the client side of the consumer portal.

Potential problem: In an application that performs navigation to a page whose PCD address is taken from the navigation structure, the navigation can result in a "Page not found" message since this page is not located on the consumer.

Object Based Navigation

Object Based Navigation (OBN) provides developers with client-side API for launching OBN (EPCM.doObjBasedNavigate or EPCM.doNavigate with target=OBN://…). In addition, OBN provides developers with server-side API for querying OBN and building OBN context menus. Both API's result in a search for navigation targets in some of the roles that are assigned to the user.

The OBN framework has implemented a special logic to seamlessly support federated scenarios. The OBN framework routes OBN queries from applications to their correct destination (producer or consumer) based on the FPN use case. In an FPN environment, a role can be used in two ways:

- Remote Role Assignment (RRA) – the user is assigned to a role that is located in the producer. In this use case, OBN searches for navigation targets in the producer role.

- Remote Delta Link (RDL) – the user is assigned to a role located in the consumer that is a delta link of the role in the producer. In this use case, OBN searches for navigation targets in the consumer role.

For example, assume the user is assigned to two roles: R1 and R2, while R2 is a federated role. When the user clicks an OBN link in the application, the OBN framework searches for navigation targets in R1 and R2. If R2 is used via RRA, then the OBN framework searches for navigation targets in the producer portal. If R2 is used via RDL, then the OBN framework searches for navigation targets in the consumer portal.

In the RDL mode, the resolution of OBN targets takes place on the consumer. To resolve such links, use the getService(INavigationService.KEY) method of NetWeaver 7.0 rather than getService(IOBNService.KEY) of NetWeaver 7.1, even if the producer is running on NetWeaver 7.1. The getService(IOBNService.KEY) does not support RDL mode.

Client-Side Functionality

Be aware that all client-side functionality, such as EPCM, occurs on the consumer side, since the browser window points to the consumer and the application runs on the producer.

Potential problems:

- The producer is running a newer version of EPCM than the consumer. The application might call a new method that does not exist on the consumer.

- If a portal component writes JavaScript code to the response in the server-side *doContent()* method, invoked on the producer, the JavaScript is executed on the consumer client-side.

Running Successfully on the Producer

Check that your application runs as expected directly on the producer portal.

Web Dynpro Java Guidelines

This section extends the general guidelines for Web Dynpro applications.

In the Web Dynpro-Portal integration there is a tight coupling between the Web Dynpro applications to the portal Web Dynpro page.

The guidelines below are relevant for consuming portal pages.

Portal Service Factory

Web Dynpro iViews/applications are tightly coupled to the portal page when running in the portal.

In order for the contextual services, such as *IWDTrayService* or *IWDPageService*, to work properly, all the Web Dynpro content that runs in the content area should originate in the same producer.

Thus, a local Web Dynpro page cannot communicate with a nested remote Web Dynpro page on the server side. For example, if a local Web Dynpro page contains nested Web Dynpro iViews and a remote Web Dynpro page, the *IWDPageService.getVisibleIViews()* method call in the remote page returns the list of iViews on the remote page.

Related Content

[Implementing a Federated Portal Network](#)

[Overview: Implementing a Federated Portal Network](#)

[Using Object-Based Navigation in a Federated Portal Network](#)

For more information, visit the [Portal and Collaboration homepage](#).

Copyright