Calculate the Number of Portal Hits

Applies to:

SAP Netweaver Enterprise Portal.

Summary

This article describes one of the methods to count the number of portal hits. For more information, visit the Portal and Collaboration homepage.

Authors: Deepak Jaju

Company: Accenture, Bangalore

Created on: 10 September 2008

Author Bio

Deepak Jaju is an Enterprise Portal Consultant at the SAP Netweaver Center of Excellence at Accenture, Bangalore-India.

Table of Contents

Problem Description

Let's assume that XYZ organization would like to know about the total portal hits at any point of time and use this metric for better development of the portal or for any other purposes like sales strategies.

SAP ships the standard "Activity Reports" for portal monitoring purposes. The features provided by these reports are as follows:

- Most popular iViews / pages.
- Number of active users.

To implement the XYZ organization's requirement:

- One of the ways to accomplish is to enable the "Monitor Hits" attribute of an iView / page and keep track of the count using the "Activity Reports". The disadvantage of this method is that; if we revisit a page multiple times, the count is incremented again and again even in the same session.

Hence, we have to implement a solution where-in the count is tracked only once in a session at the time of successful log in and not thereafter.

Solution Approach

This article describes a solution to accomplish the above portal count activity. The count obtained can be displayed anywhere in the portal through an iView. The count logic is incorporated in the logon page of the portal.

The solution is a two step process:

1 Implement the count logic
   - The count is stored in a properties file in KM. The same is accessed and incremented whenever the portal logon page is accessed for login.

2 Access the portal hit count
   - The count stored in a properties file in KM is accessed and displayed through an iView.

Note: The current scenario is executed using SAP Enterprise Portal 7.0 SP 15 version.

Prerequisites

- SAP NWDS (The scenario in this article uses the NWDS 7.0.15 version).
- Familiarity with PAR files creation and deployment.
- Experience with the logon par modification.
- Good understanding of the Knowledge Management Framework and its API's.
- Create a properties file in KM before deploying the code.

Solution

Property file creation in KM

1. Create a new text file called PortalCount.properties on the local PC. (If there is a warning message regarding the change of the file extension, choose “Yes”.)
2. Open the file using notepad and write the following:

   PortalCount=0
3. Save the file and close.
4. Go to the Content Administration -> KM Content
5. Go to the /documents repository.
6. Upload this PortalCount.properties file in this repository.

Implement the count logic

The steps are as follows:

1. Get the logon par file of the portal from the System Administration.
2. Import the file using NWDS and open the umLogonPage.jsp file.
3. The logic here is to increment the count if there is no login error. Hence, navigate to the following code where error message is displayed in the umLogonPage.jsp.

   ```
   <% if ( error.getMessage() != null ) { %>
   ```

   Now to this “If” condition, add an else part where-in the logic of the count is implemented.
4. Here is the complete code for the same:

```
       <% if ( error.getMessage() != null ) { %>
<div class="urMsgBarErr" style="margin-bottom:3;">
  <table border="0" cellpadding="0" cellspacing="0">
      <tbody>
           <tr>
                <td class="urTxtStd"><span class="urMsgBarImgError"><img height="12"
width="12" src="<%=webpath%>css/common/1x1.gif"></span></td>
                  <td><span class="urTxtStd"
tabindex="0"><%=EncodeHtmlTag.encode(logonMessage.print(error.getMessage()))%></span>
</td>
                </tr>
                </tbody>
              </table>
            </div>

            <% } else {

    //Adding the else part of “IF” and the Portal Hit Counter Logic

            try {
                //Initialize the User

com.sapportals.wcm.repository.IResourceContext c =
ResourceFactory.getInstance().getServiceContext("cmadmin_service");

                try{
```

```
                //Initialize the repository location

String rLocation = "/documents/PortalCount.properties";

   //Get the actual resource        com.sapportals.wcm.repository.IResource resource
=

   ResourceFactory.getInstance().getResource(

   RID.getRID(rLocation),c  );

//Access the content of the resource

com.sapportals.wcm.util.content.IContent content = resource.getContent();

//Create a new Properties object and store the contents of the file //into it.
Properties prop = new Properties();

prop.load(content.getInputStream());

                content.close();


//Access the value of the key and increment it.
                String value = prop.getProperty("PortalCount");

String newValue="";
Integer bigInt;
int i1=0;

   //Parsing the String value to int.
                      try {
                            i1 = Integer.parseInt(value);
                      } catch (NumberFormatException e2) {
                            // TODO Auto-generated catch block
                            e2.printStackTrace();
                      }

                //After converting to int, increment it.
                i1++;

                //Convert int to Integer and then to String.
                bigInt = new Integer(i1);

                newValue = bigInt.toString();

   //Store this new value into the properties object.

prop.put(“PortalCount”,newValue);

   resource.delete();

//Delete the existing PortalCount.properties file so that a new file //can be created
and uploaded back to KM. Instead of deleting the //resource, the contents can also be
updated using the required //APIs. In this case, the former option is used.
```

```
   //Creating a new resource again.

   com.sapportals.wcm.repository.IResource resourceCreate;

resourceCreate =
ResourceFactory.getInstance().getResource(RID.getRID("/documents"),c);


   //Creating a collection
   com.sapportals.wcm.repository.ICollection collection1 =
(ICollection)resourceCreate;

 //Create a FileOutputStream to the file which is to be created.

 FileOutputStream fileOutput = new FileOutputStream("PortalCount.properties");

 //Store the properties object in that file.

prop.store(fileOutput,null);

 fileOutput.close();

//Inputstream to the file
 FileInputStream fin = new FileInputStream("PortalCount.properties");

//Store the contents in Content object of KM
com.sapportals.wcm.repository.Content con = new Content(fin,"text/plain",-1L);


//Create the actual resource using the Collection which was just //created above.
com.sapportals.wcm.repository.IResource newResource =
collection1.createResource("PortalCount.properties",null,con);
    con.close();
    fin.close();

   //Close all the I/O streams to avoid memory problems.

}//Close of inner try

   catch (com.sapportals.wcm.repository.ResourceException e1) {
          e1.printStackTrace();
   }
   catch (com.sapportals.wcm.util.content.ContentException e1) {
          e1.printStackTrace();
   }

 }//Close of the outer try

catch (Exception e){
   e.printStackTrace();
}

}//End of else part

    %>
```

```
<!--End of Portal Hits Counter Logic -->
```

5. Refer to the Note section before build and deploy
6. Build the project and deploy.
7. Call the logon page of the portal and the count is increased

Access the portal hit count

In the previous section, we implemented the portal count logic. In this section, we will access the portal count and display it using an iView. This iView can be placed anywhere in the portal.

The steps to implement this are as follows:

1. Create a Portal Project.

   After this, create a package called com.companyname.portalcount

2. Create a Bean to store the values.

   Create a java class with following name: ReadPortalHitBean.java

   The contents of the class are as follows:

```
package com.companyname.portalcount

import java.util.Properties;

/**
 * @author deepak.jaju
 *
 * To change the template for this generated type comment go to
 * Window&gt;Preferences&gt;Java&gt;Code Generation&gt;Code and Comments
 */
public class ReadPortalHitBean {

String errorMessage;
Properties portalHitCount;

	/**
	 * @return
	 */
	public String getErrorMessage() {
		return errorMessage;
	}

	/**
	 * @return
	 */
	public Properties getPortalHitCount() {
		return portalHitCount;
	}

	/**
	 * @param string
	 */
	public void setErrorMessage(String string) {
		errorMessage = string;
	}
```

```
        /**
         * @param properties
         */
        public void setPortalHitCount(Properties properties) {
               portalHitCount = properties;
        }

   }
```

3. Create an AbstractPortalComponent and write the logic in doContent method.

```
   ClassName: ReadPortalHitStatistics

package com.companyname.portalcount;

import java.io.IOException;
import java.util.Properties;

import com.sapportals.portal.prt.component.*;
import com.sapportals.wcm.repository.IResource;
import com.sapportals.wcm.repository.IResourceContext;
import com.sapportals.wcm.repository.ResourceException;
import com.sapportals.wcm.repository.ResourceFactory;
import com.sapportals.wcm.util.content.ContentException;
import com.sapportals.wcm.util.content.IContent;
import com.sapportals.wcm.util.uri.RID;

public class ReadPortalHitStatistics extends AbstractPortalComponent
{
   ReadPortalHitBean readBean = new ReadPortalHitBean();

public void doContent(IPortalComponentRequest request, IPortalComponentResponse
response)
    {
   try {
         //Initialize the User
         IResourceContext c =
ResourceFactory.getInstance().getServiceContext("cmadmin_service");

         try {
         //Get the Repository Location and access the file
         String rLocation = "/documents/PortalCount.properties";

IResource resource =
ResourceFactory.getInstance().getResource(RID.getRID(rLocation),c);

   //Access the content

   IContent content = resource.getContent();

   Properties portalCountProperties = new Properties();

               try {
```

```
                                                                    //Load the
properties object with the contents of the file.
   portalCountProperties.load(content.getInputStream());
                     } catch (IOException e) {
                           readBean.setErrorMessage("true");
                     }
   //Set the Bean using the properties object.

   readBean.setPortalHitCount(portalCountProperties);

   readBean.setErrorMessage("false");
//If there is no error, we set the message as false; else true.

   content.close();

   } // 2nd try block

         catch (ResourceException e1) {
               readBean.setErrorMessage("true");
         }
         catch (ContentException e1) {
               readBean.setErrorMessage("true");
         }
    } // 1st try block
   catch(Exception e){
         readBean.setErrorMessage("true");
   }

   request.getComponentContext().putValue("readBean",readBean);

   com.sapportals.portal.prt.resource.IResource jspFile =
   request.getResource("jsp", "jsp/Display.jsp");

   response.include(request, jspFile);

    } // End of doContent
} //End of the class
```

4. Create jsp page to display the output.

   Create a jsp file in the PORTAL-INF -> jsp folder

   Name of the jsp file: Display.jsp

   Contents of Display.jsp:

```
<%@ page import = "java.util.Properties" %>
<%@ page import = "java.util.Enumeration" %>

<jsp:useBean id="readBean" scope="application"
class="com.companyname.portalcount.ReadPortalHitBean" />

<% Properties portalCountProperties = new Properties();
portalCountProperties = readBean.getPortalHitCount();
               String value = portalCountProperties.getProperty("PortalCount");
String errorMessage=readBean.getErrorMessage();
```

```
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Portal Hit Count</title>
</head>

<% if (readBean.getErrorMessage().equalsIgnoreCase("true")) { %>


<body bgcolor="#F8F6E0">

    <table border="0" width="100%" height="100%" id="table1" cellspacing="0"
cellpadding="0" bgcolor="#F8F6E0">
   <tr>
          <td width="102" style="font-family: sans-serif; font-size: 11px">Portal Hit
Count:</td>
          <td style="font-family: sans-serif; font-size: 11px">Information not
available</td>
   </tr>
</table>

<% }

else

{ %>

<body leftmargin="40" bgcolor="#F8F6E0">

<table border="0" width="100%" height="100%" id="table1" cellspacing="0"
cellpadding="0" bgcolor="#F8F6E0">
   <tr>
          <td width="54" style="font-family: sans-serif; font-size: 11px">Portal
Hits:</td>
          <td style="font-family: sans-serif; font-size: 11px"><%=value%></td>
   </tr>
</table>

<% } %>

</body>
</html>
```

5. Refer to the Note section before build and deploy.
6. Build the project and deploy.
7. Create an iView out of this par file which can be used to display the portal count.

Note

Implement the count logic

i. Add the following page imports to the umLogonPage.jsp:

```
<%@ page import = "java.util.Enumeration" %>
<%@ page import = "java.util.Properties" %>
<%@ page import = "java.io.FileInputStream" %>
<%@ page import = "java.io.FileOutputStream" %>

<%@ page import = "com.sapportals.wcm.util.uri.RID" %>
<%@ page import = "com.sapportals.wcm.repository.ICollection" %>
<%@ page import = "com.sapportals.wcm.repository.ResourceFactory" %>
<%@ page import = "com.sapportals.wcm.repository.Content" %>
```

ii. Add the relevant KM jar files using the following path from NWDS:

- Choose “Java Build Path” from the Project properties.
- Click on “Add variable”
- Select the “ECLIPSE_HOME” variable and click on “Extend”.
- Choose the plugins folder.
- Select the “com.sap.km.rfwizard_7.1.15” folder -> lib folder. (The folder name can vary depending on the NWDS version being used.)
- Select all the available jars and click OK.

iii. Modify the portalapp.xml

- Open the portalapp.xml
- Choose the “Application” tab and add the following:
  - SharingReference=knowledgemanagement
- Choose the “Components” tab and add the following component-config property to default profile.
  - SafetyLevel=no_safety

Access the portal hit count

i. Add the relevant KM jar files using the following path from NWDS:

- Choose “Java Build Path” from the Project properties.
- Click on “Add variable”
- Select the “ECLIPSE_HOME” variable and click on “Extend”.
- Choose the plugins folder.
- Select the “com.sap.km.rfwizard_7.1.15” folder -> lib folder. (The folder name can vary depending on the NWDS version being used.)
- Select all the available jars and click OK.

-

ii. Modify the portalapp.xml

- Open the portalapp.xml
- Choose the “Application” tab and add the following:
  - SharingReference=knowledgemanagement

Additional Information

The count is increased when the logon page is called. Hence, if log off action redirects to the logon page; the count is increased again. In this case, additional coding needs to be done during log off action.

- Logon page called – Count increased.
- Confirmed Log off action (i.e. choosing “Yes” while log off) – Capture the log off action and decrement the count.
- Since log off action redirects to the logon page, the count is increased again. Hence, the count is increased only once during a complete session.

If log off redirects to some other page, then no additional coding is required.

Related Content

SDN How to Guides

Enterprise Knowledge Management

SAP Help

For more information, visit the Portal and Collaboration homepage.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.