

Extending the FuegoBPM Work Portal V5

Fuego, Inc.

Extending the FuegoBPM Work Portal V5

by Fuego, Inc.

Published January, 2005 - Version 5.5. Revision 5 - April, 2005.

Copyright © 2001-2005 Fuego, Inc.

Extending the FuegoBPM Work Portal

Copyright 2001-2005 Fuego, Inc. All rights reserved.

This documentation is subject to change without notice. This documentation and the software described in this document contains proprietary trade secrets and confidential information of Fuego, Inc. and is also protected by U.S. and other copyright laws and applicable international treaties. Use of this documentation and the software is subject to the license agreement between you and Fuego, Inc. If no such license agreement exists, you may not use this documentation and software in any manner whatsoever. Unauthorized use of the documentation or software, or any portion of it, will result in civil liability and/or criminal penalties. U.S. Patent Pending.

Fuego, Fuego 4, Component Manager, Process Designer, Work Portal, Orchestration Engine, Execution Console, Process Analyzer, Organization Administrator are trademarks or registered trademarks of Fuego, Inc.

FuegoBPM 5, FuegoBPM Studio, FuegoBPM Designer, FuegoBPM Enterprise Administration Center, FuegoBPM Work Portal, FuegoBPM Portal Console, FuegoBPM Archive Viewer, FuegoBPM Logviewer, FuegoBPM Express Server, FuegoBPM Enterprise Server, FuegoBPM Application Server Edition, FuegoBPM Web Console, FuegoBPM Process Analyzer, FuegoBPM Data Store, FuegoBPM Dashboard, FuegoBPM BAM, FuegoBPM Portlets, FuegoBPM Suite, FuegoBPM Deployer, FuegoBPM Failover, FuegoBPM VCS, FuegoBPM Ant Tasks, FuegoBPM FDI, FuegoBPM Help Viewer, FuegoBPM Server are trademarks or registered trademarks of Fuego, Inc.

InstallAnywhere is a registered trademark of Zero G Software, Inc. Solaris and Java are trademarks of Sun Microsystems, Inc. Windows is a registered trademark of Microsoft Corporation.

All other trademarks, trade names, and service marks are owned by their respective companies.

Table of Contents

1. Introduction	6
Why not just use PAPI instead	6
2. Understanding the Work Portal	8
A Web Application	8
Directory Structure	9
How it works	10
3. Customizing the Work Portal	12
The portal.properties file	13
Application Title Logo	13
CSS	14
Styles used by Work Portal	16
Custom Images	17
Image Bundle	17
Creating a new ImageBundle	19
Modifying the JSPs	20
JSP Bundle	21
Creating a new JSPBundle	22
JSP Inside the JSPs	23
An example	25
WAPI	29
Example	30
Context	31
4. Appendix A: Main JSPs' layouts	33
instancesView.jsp	33
instanceDetail.jsp	33
search.jsp	34
5. JSP Descriptions	36
activityDoc.jsp	37
applicationsView.jsp	37
attachmentsView.jsp	38
auditTrail.jsp	39
changePassword.jsp	40
checkinAttachment.jsp	40
checkinWorkingAttachment.jsp	41

editAttachment.jsp	42
editOptions.jsp	42
executionLock.jsp	44
fileTypeAssociations.jsp	45
folder.jsp	45
instanceDetail.jsp	46
instancesView.jsp	48
login.jsp	49
logout.jsp	50
newAttachment.jsp	50
newNote.jsp	51
participantList.jsp	52
processDoc.jsp	53
search.jsp	54
send.jsp	57
sendTo.jsp	58
servererror.jsp	59
sessionExpired.jsp	60
viewAttachment.jsp	61
viewNote.jsp	62
viewWorkingAttachment.jsp	62
welcome.jsp	63

Chapter 1. Introduction

Introduction

The *Web-based Work Portal* is a Java Web Application. It was designed using the JSP Model 2 Architecture in order to achieve a clean separation of presentation from content. This means that the Work Portal follows the MVC (Model-View-Controller) design pattern, keeping the presentation (view) in JSP pages, and encapsulating the Model and Control in a servlet library.

This design gives place to a clean way to change the presentation layer of the Work Portal. Basically, no more than the JSP pages need modifications.

The Work Portal architecture helps the developer change the views in a controlled way, and WAPI (Work Portal Application Programming Interface) provides calls to interact with the servlets that govern the Work Portal logic.

Why not just use PAPI instead

PAPI is the Java interface used by client applications that need access to FuegoBPM processes. The applications in the FuegoBPM suite (like FuegoBPM Studio, FuegoBPM Console and FuegoBPM Work Portal) use PAPI to communicate with the process servers.

For most applications in which the user interface to the FuegoBPM system needs to be customized, personalizing the Work Portal and using WAPI is a wiser approach than using plain PAPI.

WAPI provides a higher level of abstraction on top of PAPI, and provides functionality that would otherwise be tedious to implement directly with PAPI, like:

- Authentication and PAM (Pluggable Authentication Modules)

- User session control
- Instance search and filters
- Instance Attachments management
- User preferences

Chapter 2. Understanding the Work Portal

Understanding the Work Portal

This chapter describes how the Work Portal is structured and explains how all the pieces work together. It provides the background knowledge needed to understand the Work Portal for later customization.

A Web Application

The Work Portal is a standard Servlet Web Application, thus, it runs within the context of a Servlet/JSP engine. This engine provides the environment where the web application runs, and plays the role of a mediator between the HTTP server (commonly called webserver) and the web application.

FuegoBPM Enterprise provides a web application servlet embedded and configured to ease Work Portal installation. Using FuegoBPM Enterprise Administration Center you only need to configure the directory server and change some Work Portal preferences in order to give them the values that better fit your needs. See chapter **FuegoBPM Enterprise Administration Center** in the System Administration Guide documentation.

The webserver and the Servlet/JSP engine must be setup to work together. Then, the later should be configured to run the Work Portal as a web application.

As an example, take the Resin (<http://www.caucho.com>) engine. It's a Servlet/JSP engine that can be configured to run with many webserver, like Apache (<http://httpd.apache.org>) for example. It also includes an embedded webserver for easy deployment and development.


To deploy new web apps in Resin, a new entry must be added in it's

XMLbased configuration file. For instance, in order to deploy the Web Work Portal, an element like this one could be added into it's configuration file (within a **host** element):

```
<host>
...
<!-- Work Portal 5.1 Application Entry -->
<web-app
id='/workPortal'
app-dir='/usr/local/fuego5.1/enterprise/webapps/portal'>
</web-app>
...
</host>
```

The previous web-app tells Resin about the new web application that is being added. In a few words, it specifies that the **/workPortal** URL prefix will be mapped to the web application located at the directory specified by the **app-dir** attribute.

Note

 The directory in the previous example (line 6) is a common location for the Work Portal after its installation on a Unix system. On Windows, it would be something like the following:
app-dir=C:\fuego5.1\enterprise\webapps\portal

Consult the [Fuego System Administrator's Guide] for more information on how to configure and deploy the Work Portal.

Directory Structure

On a standard FuegoBPM Enterprise software installation, the Work Portal files are located under the **fuego/enterprise/webapps/portal/** directory (where fuego is the Fuego installation directory). This is the Root directory of the Work Portal web application. Within this directory, the following sub-directories will be found:

- **css/** - CSS (Cascading Style Sheets) for the Work Portal. Modifying the CSS files is the easiest way to customize the look of the Portal. Within a CSS file, things like the font sizes, colors, margins and decorations can be changed. This directory includes example files with pre-defined styles for using with the Work Portal.
- **jsp/** - JSP pages which conform the views of the Work Portal.
- **img/** - Contains the set of images used in the views. The current set can be customized and changed at will, and new sets can be created.
- **WEB-INF/** - Following the Java Servlet Specification, this is the standard directory where the Work Portal web application meta-information is stored (the deployment descriptor file: WEB-INF/web.xml) together with the servlets' code (under the WEB-INF/classes/ subdirectory).
- **lib/** - External library files used by the Work Portal.
- **customjsp/** - JSP pages to be used with the fuego JSP component (from CIL).
- **wapi/** - Javadoc documentation for the WAPI interface.

For the purpose of modifying and extending the Work Portal's functionality or it's look and feel, the most relevant directories are: **css/**, **jsp/** and **img/**.

In the directory **fuego/enterprise/webapps/portal/WEB-INF** there is a file named *portal.properties*. This is the properties-file used to configure some aspects of the Work Portal. It's a plain-text configuration file that can be modified with a text editor for customization.

How it works

Once the Work Portal is configured and ready to run, a client can start using it by pointing the web-browser to the URL that is mapped to the Work Portal application (as configured in the webserver and servlet engine). For example:

`http://hostname/workPortal.`

When the client connects to the Work Portal URL, the index.html file in the Work Portal Root directory will be served to the client, who will be redirected to the Controller servlet, which takes control of the screen-flows from there on.

The Controller servlet, without more parameters, will start with the login page. From this point the whole Work Portal execution is, basically, a continuous flow of JSP pages that call a servlet which, in turn, displays another JSP.

Each JSP page has a defined purpose, and it gives the user links, buttons and other graphical means to perform different operations. This operations will actually be translated into calls to a servlet. This servlet will execute the operation requested (provided that all the required parameters for that operation were passed) and will call the next JSP page as needed.

Which JSP is to be called next is decided by the Controller servlet, taking into account things like the operation requested, parameters, context, possible errors and user preferences.

Chapter 3. Customizing the Work Portal

Customizing the Work Portal

There are a few distinct ways to customize the Work Portal. Each one offers a different level of customization. Which ones to use depends on what needs to be accomplished.

These are the main ways the Work Portal can be customized, in increased order of flexibility (and complexity):

- **portal.properties** file - Some of the Work Portal behavior and appearance can be customized by only changing the properties in this file.
- **CSS (Cascading Style Sheets)** - Modifying the CSS files is the easiest way to customize the look of the Work Portal. Within a CSS file, things like the font sizes, colors, margins and decorations can be changed. The big advantage is that only one file (the styles.css) needs to be modified.
- **Images** - The set of images (mainly .gif files) that the Work Portal uses can be changed as desired. This is also a very simple way of customizing the Work Portal, since it does not require programming knowledge either.
- **Modify JSP pages** - This approach is obviously more advanced than the previous two. It is both more complicated and more flexible. It involves changing the code within one or more of the standard JSP pages of the Work Portal.
- **Using WAPI calls** - This gives even more power and flexibility to the programmer. When modifying the JSP pages, or even creating new ones, WAPI calls provide access to the servlets' functionality.

The portal.properties file

As stated before, the portal.properties configuration file is used to configure some aspects of the Work Portal. It's a self-documented plain-text configuration file that can be modified with a text editor for customization. It is located in the webapps/portal/WEB-INF/ directory.

The Work Portal reads this file to figure out, among other things, which .css file to use, the desired set of images and the set of JSPs to use. Things like temporary directories, log files, logo, session time-out and more are also specified in this configuration file. Consult [Fuego System Administrator's Guide] for more information on how to configure and deploy the Work Portal.

Warning



The portal.properties file is self documented and easy to understand.

However, its information is required for the Work Portal to work properly. Thus, as with any configuration file, it is good practice to keep a copy of the original before doing modifications to it.

Note



This file is read when the Work Portal starts up (when the servlet engine loads it). Thus, for any modifications to take effect the servlet engine must be told to reload the Work Portal web application.

Application Title Logo

The default Fuego Logo located at the top left corner of the Work Portal pages can be easily changed by modifying the **fuego.portal.-apptitle.type** and **fuego.portal.apptitle.value** properties of the **portal.properties** file:

```
...
# LOGO
#####
# The apptitle.type is the type of logo.
```

```
# It can be "image" or "text"
# In case of image the value is the key in the ImageBundle
# In case of text the value is a title to be printed.
#####
fuego.portal.apptitle.type=<replaceable>image</replaceable>
fuego.portal.apptitle.value=<replaceable>LOGO</replaceable>
...
```

The logo image size is specified in the CSS file used by the FuegoBPM Work Portal. If your image has a different size, change it in the css file line `".logoimage {width: 160px; height: 23px; padding : 12px 12px 12px 12px;}"`

As the commentary explains, the value of `fuego.portal.apptitle.type` must be either text or image. The former is used to display plain text instead of an image logo.

In the case of text, the `fuego.portal.apptitle.value` property will contain the text to be displayed as the logo. Otherwise, when type image is used, it will specify the key of the image within the image bundle.

The following example will use the text *MyCompany Inc.* in place of the Fuego image logo:

```
...
fuego.portal.apptitle.type=text
fuego.portal.apptitle.value=MyCompany Inc.
...
```

CSS

The Web Work Portal uses CSS (Cascading Style Sheets) to control the appearance and positioning of elements on the web pages.

Modifying the CSS files is the easiest way to customize the look of the Work Portal. Adapting the CSS may be enough to reflect the

company standards or to match the look and feel of a particular legacy application.

The standard CSS file used by the Work Portal is **style5.css**, which is located under the **/webapps/portal/css/** directory. Within this directory, more example CSS files will be found.

The portal.properties configuration file contains a property that tells the Work Portal which CSS file to use. It can be modified to use another file. This is an excerpt of the standard portal.properties after installation:

```
...
#####
# Css
#The name of the cascade style sheet file
# This file must be in portal/css directory
# The default options are:
# style.css, style1.css, style2.css, style3.css, style4.css
#####
fuego.portal.stylesheet=style5.css
...
```

So, to make the Work Portal use a different style, it can be changed. Like:

```
...
#####
fuego.portal.stylesheet=ourStyle.css
...
```

This is the recommended procedure to create a personalized style-sheet file:

1. Make a copy of the style-sheet file that looks closer to what is needed. Remember that the style-sheet file should be located in

the css/ directory for the Work Portal to find it.

2. Edit portal.properties again so that it uses the newly created style-sheet file.
3. Finally, make modifications to the new style-sheet file at will, until the desired look is achieved. Note that when the .css file itself is modified, there is no need to re-start the Work Portal.

For example, copy the styles.css file to mycompany.css. Edit portal.properties:

```
...
#####
# Css
# The name of the cascade style sheet file
# This file must be in portal/css directory
# The default options are:
# style.css, style1.css, style2.css, style3.css, style4.css
#####
fuego.portal.stylesheet=mycompany.css
...
```

And then, modify mycompany.css until you get the desired look and feel.

Styles used by Work Portal

The following table can serve as a reference for modifying the styles in use by the Wprk Portal.

CATEGORY	STYLES	PURPOSE
HTML Tags	BODY, A, TD, LI, SELECT, TEXTAREA, INPUT	Redefine the look and feel of html tags
General	label, fixedlink, text-small, loginbg	Define multipurpose styles

CATEGORY	STYLES	PURPOSE
Headers & Footers	apptitle, title, header, footer, headline1, headline2, headline3, welcome	Define the look&feel of header, footer and titles
Process	Tree ptreebg, ptreelevel0, ptreelevel1, ptreeleveldeprecated, menulinks, menulinkshi, enulinksdeprecated	Define the look and feel of the process tree text and its background
Dialog	dialogtitle, dialogtitle2, dialoglabel, dialogtext	Define look and feel of titles, labels and text for dialog windows
Tables	tablettitle, tablettitle2, tablettext2, tablettitle3, tablettext3	Used for different Work Portal tables
Calendar	calendartitle1, calendartitle2, calendarOn, calendarOff	Manage the style of the calendar pop-up window

Custom Images

The set (bundle) of images the Work Portal uses can also be customized.

The images are a mainly **.gif** files that reside somewhere in the **webapps/portal/img/** subdirectory of the Work Portal.

Image Bundle

In order to facilitate modification, the images the Work Portal uses are not hardcoded in the JSP pages' code. Instead, a separate file (the ImageBundle file) keeps a mapping between a symbolic name for the image and the actual file name of the **.gif** file. So, making the Work Portal use different icons is just a matter of modifying that file.

The **ImageBundle** file should be located in the `/enterprise/webapps/portal/WEB-INF/classes` directory. The standard installation uses the **ImagesBundleSet1.properties** file by default.

The name of the file must end with the **.properties** suffix.

The ImageBundle file is a plain-text file in which each line follows a simple KEY=value structure.

Here is an excerpt of the default ImagesBundleSet1.properties:

```
...
ATTACH = ../img/set1/attach.gif
GRABOFF = ../img/set1/icons/graboff.gif
PROCESSOFF = ../img/set1/icons/processoff.gif
SELECTOFF = ../img/set1/icons/selectoff.gif
SUSPENDOFF = ../img/set1/icons/suspendoff.gif
RELOAD = ../img/set1/reload.gif
...
```

The KEYS are the symbolic names the Work Portal uses to identify each icon (left side of the "=" sign). The values are the actual path+file-name to the image file on disk (right side of the "=" sign).

As the previous excerpt suggests, the paths to the image files are relative to the **webapps/portal/jsp/** directory. Nothing forces the images to be in the **img/** directory, but it is a good practice to keep all Work Portal images under it to keep consistency.

Note



Even in Windows environments, the forward slash "/" should work as a path separator.

The Work Portal knows which ImageBundle file to use because it takes it from the **fuego.portal.imageBundleFile** property specified in the **portal.properties** configuration file:

```
...
# Bundle Files #####
fuego.portal.imageBundleFile=ImagesBundleSet1
...
```

Note



The file name specified should not include the .properties suffix. It will be automatically appended by the Work Portal before looking for the file.

Creating a new ImageBundle

Following is the recommended procedure to create a new ImageBundle:

- Copy the supplied ImageBundle files into a new file. Remember that the file must be in the `classes/` directory, and its name should end with the .properties suffix. Example: `classes/MyCompanyImages.properties`.
- Modify `portal.properties` so that the new ImageBundle file is used instead of the standard one. Example:

```
...
# Bundle Files #####
fuego.portal.imageBundleFile=MyCompanyImages
...
```

- Create a new directory where the new icons and images will be placed. Preferably, make it a subdirectory of **img/**. Example: **img/mycompany/**.
- Copy the new image files into the directory created in the

previous step.

- Modify the new ImageBundle file (**MyCompanyImages.properties** in the previous example) so that the desired icons are taken from the new image files (located in the new subdirectory). Example: to use new **.gif** files the **ATTACH** and **RELOAD** icons:

```
...  
ATTACH = ../img/mycompany/attach.gif  
GRABOFF = ../img/set1/icons/graboff.gif  
PROCESSOFF = ../img/set1/icons/processoff.gif  
SELECTOFF = ../img/set1/icons/selectoff.gif  
SUSPENDOFF = ../img/set1/icons/suspendoff.g  
RELOAD = ../img/mycompany/reload.gif  
...
```

Only the desired images need to be changed on a new **ImageBundle**. **Images** used on other bundles can be reused. Actually, this is one of the advantages of using a new ImageBundle file instead of just replacing the standard **.gif** files. As an example, the images in **img/misc/** are used in both standard bundles **ImagesBundleSet1.properties** and **ImagesBundleSet2.properties**.

Modifying the JSPs

As explained previously, the JSPs that make up the Work Portal can also be modified to suit particular needs.


Warning



You must be aware that changing the standard JSP pages that conform the Work Portal application, implies that when a patch of Work Portal is applied, all the customizations made to the standard pages should be applied to the new version of the page, otherwise your version might contain errors or problems already fixed by Fuego.

See **appendix A** for a description of how the main pages of the Work Portal are laid out visually. For a comprehensive description of each page refer to **appendix B**.

Note

 Changing the standard JSPs that conform the Work Portal requires programming knowledge on Java, JSP and some JavaScript.

JSP Bundle

The JSPs could be changed directly. But, for greater flexibility, the Work Portal takes the set of JSPs to use from a JSPBundle file, which is similar to the ImageBundle files.


The JSPBundle file should be located in the **enterprise/webapps/portal/WEB-INF/classes** directory. The standard JSPBundle file is JSPBundle.properties. The name of the file must end with the .properties suffix.

This is an excerpt of the default JSPBundle.properties file:

```
...  
INSTANCE_DETAIL = /jsp/instanceDetail.jsp  
INSTANCES_VIEW = /jsp/instancesView.jsp  
LOGIN = /jsp/login.jsp  
LOGOUT = /jsp/logout.jsp  
...
```

The KEYs are the symbolic names the Work Portal uses to identify each of the JSP pages (left side of the "=" sign). The values are the actual path+file-name to the JSP file on disk (right side of the "=" sign).


Note

 Even in Windows environments, the forward slash "/" should work as a path separator.

The JSPBundle the Work Portal will use must be specified in the portal.properties configuration file, with the **fuego.portal.jspBundleFile** property:

```
...
# Bundle Files #####
...
fuego.portal.jspBundleFile=JSPBundle
...
```

Note

 The file name specified should not include the .properties suffix. It will be automatically appended by the Work Portal before looking for the file.

Warning



It is worth note that not all of the JSPs used by the standard Work Portal have a corresponding key in the JSPBundle. That is because some *.jsp* files are included by other composite pages. This means that if any of the included pages is copied to a new filename, the reference in the parent page must be updated. For example, the dispatcherMenu.jsp is not a whole page in itself (and thus, has no key in the bundle), instead, it is included by other pages such as instancesView.jsp and attachmentsView.jsp . Similarly, footer.jsp has no key , but is included by many other pages.

Creating a new JSPBundle

Following is the recommended procedure to create a new JSPBundle:

- Copy the supplied JSPBundle.properties file to a new file. Remember that the file must be in the **clasess/** directory, and its name should end with the .properties suffix. Example: **clasess/MyCompanyJSPs.properties**.
- Modify portal.properties so that the new JSPBundle file is used instead of the standard one. Example:

```
...
# Bundle Files #####
...
fuego.portal.jspBundleFile=MyCompanyJSPs
...
```

- Create a new directory where the new JSP pages will be placed. Preferably, make it a subdirectory of **jsp/**. Example: **jsp/mycompany/**.
- Copy the JSPs that need modification into the directory created in the previous step.
- Modify the new JSPBundle file (MyCompanyJSPs.properties in the previous example) so that the desired JSPs are taken from the new JSP files (located in the new subdirectory). Example: If the LOGIN and LOGOUT pages were to be modified:

```
...
INSTANCE_DETAIL= /jsp/instanceDetail.jsp
INSTANCES_VIEW = /jsp/instancesView.jsp
LOGIN = /jsp/mycompany/login.jsp
LOGOUT = /jsp/mycompany/logout.jsp
...
```

Only the desired JSPs need to be changed on a new JSPBundle. JSPs used on other bundles can be reused. Actually, this is one of the advantages of using a new JSPBundle file instead of just replacing the standard JSP files.

JSP Inside the JSPs

The JSPs of the Work Portal use some Java objects to interact with the servlets and the FuegoBPM Server. Basically, the types of objects

used within the JSPs can be divided in:

- Standard Java objects (like `java.lang.String`)
- PAPI objects (such as `fuego.papi.Process`)
- Or, new object types introduced by the Work Portal:
 - **`fuego.portal.SessionEnvironment`**
 - **`fuego.portal.UserOptions`**

All of the top-level JSPs in the Work Portal include `waminit.jsp` (either directly or indirectly through `init.jsp`) before doing anything else. This is so because this page initializes some useful variables that are necessary on all the Work Portal pages. These include:

- **`locale`** - A standard `java.util.Locale` object, containing the current country and language. This reflects what the user has selected in his options screen. If there is currently no user logged in, or if the user has no Locale selected, the default values will be taken from the `portal.properties` file.
- **`WamResources`** - A `fuego.resources.MsgBundle` object, which contains all the text messages the Work Portal displays to the user. This makes the Work Portal a locale-sensitive application.
- **`imagesUrl`** - Also a `java.util.ResourceBundle` object, which is used to access the current ImageBundle mappings.
- **`stylesheet`** - A String with the name of the CSS file in use.
- **`appTitleType`** - A String value, as defined in `portal.properties`.
- **`appTitleValue`** - A String value, as defined in `portal.properties`. .

Secure/Non-secure Page

Some pages can not be displayed until the user is logged into the Work Portal system. But, pages like LOGIN, should be freely accessible.

As described earlier, some JSPs in the Work Portal include `init.jsp` (which in turn includes `waminit.jsp`) instead of including `waminit.jsp` directly. This is so because the user authentication is handled by the `init.jsp` page. Thus, pages should include `init.jsp` if they require the user to be logged in, otherwise, `waminit.jsp` is enough.

In addition to the variables initialized by `waminit.jsp` (described previously), `init.jsp` defines the following ones:

- **processServiceSession** - This is an object of type `fuego.papi.- ProcessServiceSession`, from PAPI. It represents a session that allows users access every process operation such as get instances, run tasks, attachment operations, participants operations, etc..
- **userOptions** - This Work Portal specific object, of type `fuego.portal.- UserOptions`, helps to deal with the current user's information and personal preferences.

The code in `init.jsp` also redefines some of the variables already created by `waminit.jsp`, taking into account that now a user is logged in. For example, the locale variable will have the user-specific Locale assigned.

An example

The Work Portal provides the user with a menu of all the available views (views tree) on the left side of the screen.

The default behavior of the Work Portal is that when the user clicks on a particular instance view, the list of instances in that view are shown on the right side of the screen. So are the attachments when

the clicked view is an attachments view and the applications when the clicked view is an applications view.

In the case of instances views, all the instances are shown as rows in a table where the columns are the instances' variables selected by the user for the presentation of that particular view. This simple example will show how to modify the way the instances are listed in the table, so that the instances with major priority are shown emphasized with a different background and font.

As the layouts in **appendix A** suggest, the instances information list is coded in the **detailContent.jsp** file. Since this is not a top-level page (instead, it is included by many other JSP pages), there's no key in the JSPBundle for this particular page. Therefore, the same filename will be used.

Hands on

First of all, a backup copy of the original `detailContent.jsp` should be created.

Inspecting the **detailContent.jsp** file shows that after getting the request attributes with the information needed, the columns of the table are drawn.

After including the **actionToolbar.jsp**, the titles of the variables included in the presentation of the view are shown as columns. After that, a row is inserted in the table for each instance the view has, and the information of every variable of the instance is a cell of that row in the table.

Depending on the kind of the variable, the cell is drawn in different ways, the participant cell, for example has an icon beside the participant name to allow users select/unselect the instance from there. But all the cells are displayed using a class defined in a styles file.

Adding a new class for the instances with high priority, the only change in the jsp will be assign this new class for those instances

whose priority is greater than NORMAL. So, the first thing we have to do is to add a new class in the style file we have set in the **portal.properties** file in order to emphasize the instances with high priorities. Remember that if you are using one of the style files provided by Fuego, you should customize the styles file as suggested in the section CSS on page.

```
...
/* Style of the web work portal's table cells */
.tablecell f font-family: Verdana, Arial,
    Helvetica, sans-serif;
font-size: 11px; color: #000000;
background-color: #DEE7F7g
/* Style of the web work portal's emphasized table cells */
.emphTableCell f font-family: Verdana, Arial,
    Helvetica, sans-serif;
font-size: 11px; color: #000000; background-color: #FFCCCCg
...
```

Once the new class has been added to the styles file, the next step is modify the page. In the **detailContent.jsp** file, the class used to draw the cells is "tablecell". Our modification is made up of a change in the class use for the table cells in those cases where the priority is higher than normal. So, in the for loop that traverses the instances list we have to add this portion of code :

```
<%
    for (int i = 0; i < processInstances.length; i++) f
        fuego.papi.InstanceInfo ii = processInstances[i];
        if (grabActivity != null) {
            instanceActions.setProcessInstance(ii,
                ii.getActivity(),
                processServiceSession,
                grabActivity);
        } else {
            instanceActions.setProcessInstance(ii,
                ii.getActivity(),
                processServiceSession);
        }
        String cellclass = "tablecell";
        if (ii.getPriority() > FilterAttribute.NORMAL_PRIORITY)
        {
```

```

        cellclass = "emphtablecell";
    }
%>

```

The previous lines assign the properly class to the variable `cellclass`. Then the variable is used to assign the class to the cell as follows :

```

...
</td>
<%
} else
    if (columns[j].getId().equals(VarDefinition.STATUS_ID)) {
%>
<td height="9" class="<%=cellclass%>">
    <table width="100%" border="0"
        cellpadding="0"
        cellspacing="0">
        <tr>
            <td height="9" width="100%"
                class="<%=cellclass%>">
                <%=formattedValue.length()==0?
                "&nbsp;" :
                formattedValue%>
            </td>
            <td height="9"
                valign="middle"
                align="right"
                class="<%=cellclass%>">
%>
                if(ii.isGrabbed()) f %>
                <a href=
                ...
                </a>
%>
            } else if(ii.isException()) {
%>
                <a href=
                ...
                </a>
%>
            } else if(ii.isSuspended()) {
%>
                <a href=
                ...
                </a>
%>
            } else if(instanceActions.isCompletable()) {

```

```
%>
    <a href=
        ...
    </a>
<%
    } %>
</td>
</tr>
</table>
</td>
<% } else { %>
    <td height="9" class="<%=cellclass%>">
        <%= (formattedValue == null ||
            formattedValue.length() == 0) ? "&nbsp;" :
            formattedValue %>
    </td>
<%
    } %>
<% } %>
...

```

This is just an example of how, with even simple modifications, the JSPs can be altered to suit different needs. The more the programmer analyses the JSPs, the more insight he will get to change them at will.

WAPI

WAPI is the Workportal API. It provides a clean interface for accessing the Work Portal servlets' functionality.

Note



In order to use WAPI, knowledge on Java, JSP, HTML and JavaScript is needed.

Currently, WAPI consists of one Java class: `URLForAction`. Each method of this class represents a service that will be provided by one of the Work Portal's servlets. Refer to the WAPI javadoc documentation for a detailed description of every method.

The way it works is simple: every WAPI method receives a `javax.-servlet.http.HttpServletRequest` which must contain the required parameters for that call, and returns a URL (in String form) which

will be a call to the correct servlet with the necessary parameters encoded.

That URL can be used for HTML form actions, and href links that give the user a graphical interface to fire that particular call.

Example

Here is how a Global activity is called from the Work Portal:

The **applicationView.jsp** is displayed when the user selects an Applications View from the menu. At this point, the list of Global activities (also called applications) are displayed to the user. When the user clicks on a particular Global activity link, a servlet call must be made in order to actually execute that activity.

The URL to fire that servlet call is constructed with *URLForAction.runGlobalApplication(request)*. Here is an excerpt of **applicationsView.jsp**:

```
...
<form method="post"
action="<%= URLForAction.runGlobalApplication(request)%>"
name="applicationsForm">
<input type="hidden" name="activityId">
</form>
...
```

So, there is an HTML Form that will be posted to the URL created by the call to *URLForAction.runGlobalApplication(request)*. Note that the required parameter for *runGlobalApplication()* (*activityId*) will be sent when the form is submitted. Also note that the value for the *activityId* field is not set yet.

Here is the *applicationsView.jsp* code that creates the link for a Global activity:

```
...  
<a class="tablecell"  
href="javascript:runglobal('<%=activity.getId()%>')">  
<%=activity.getLabel(locale)%></a>  
...
```

The link calls the javascript function `runglobal(aid)`, which submits the form after setting the value of the `activityId` field:

```
...  
function runglobal(aid) {  
  document.applicationsForm.activityId.value = aid;  
  document.applicationsForm.submit();  
}  
...
```

Context

After a call to a Work Portal servlet, a JSP page will be executed and the resulting screen shown to the user. Which of the Work Portal's JSP will that be? That decision is made by the Controller servlet, as described in section *How it works* in chapter 2.

Normally, a particular servlet action that is requested (a method of `URLForAction`) will be followed by the same JSP. For example, after a call to `URLForAction.enterNote()`, the `newNote.jsp` page will commonly be loaded (as could be expected), and after `URLForAction.viewActivityDocumentation()` the execution of `activityDoc.jsp` will follow.

However, in some cases, the next page to be displayed is not always the same. It depends on some context information. For example, the user can select an instance from two different screens: from the list of instances screen (`instancesView.jsp`) or from the screen that shows a particular instance's information (`instanceDetail.jsp`). So, when calling the action that (un)selects an instance, the next JSP to be executed will depend on the context: it should return to the

screen where the user was when (un)selecting the instance.

There might be some special cases in which the expected JSP page is not the one loaded right after a particular action. For example, if the user is working in the Work Portal and then remains idle for a while, the session would expire. Then, when the user tries to resume working, the login screen will be displayed (login.jsp). Note also that after the login, the Work Portal will take the user to the screen where he/she was when the session expired (and not to the default welcome.jsp that normally comes after a login).

When the Work Portal finds some error while processing a request, the `theservererror.jsp` page is executed. In the case of an unexpected Runtime Error (one that the Work Portal is not able to handle), the `error.jsp` page is executed instead. Actually, the `error.jsp` page is defined as the generic error-page in the Work Portal deployment descriptor file `web.xml`).

Chapter 4. Appendix A: Main JSPs' layouts

Main JSPs' layouts

As introduced previously, most of the top-level JSP pages in the Work Portal are actually composite pages, meaning that they include other pages to compose the final layout. Following is a visual description of how the main Work Portal pages are composed of other JSP files.

instancesView.jsp

The screenshot displays the FUEGO Work Portal interface. The layout is composed of several JSP files, as indicated by the red arrows and labels:

- header.jsp**: Points to the top navigation bar containing the FUEGO logo, "Work Portal", user greeting "Welcome, Mary Jones", and links for Search, Options, Help, and Logout.
- menu.jsp**: Points to the left sidebar menu with items like Inbox, Attachments, Bookmarks, Consultations, History, and "Orders > \$200".
- actiontoolbar.jsp**: Points to the toolbar above the table, containing various icons for actions like print, delete, and refresh.
- detailcontent.jsp**: Points to the table displaying order data.
- footer.jsp**: Points to the bottom bar showing "Fuego™ - Work Portal".

The table data is as follows:

✓	Description	State	Received	Activity	Participant	Payment type	Order Amount	Initiator
<input type="checkbox"/>	Scubapro Dive Shops OrderFill4	Running	5:56:11 PM	Check Credit		<input type="checkbox"/> credit	300.00	Jane Doe
<input type="checkbox"/>	Industrial Salvage OrderFill3	Running	5:56:49 PM	Check Credit		<input type="checkbox"/> credit	406.35	Jane Doe
<input type="checkbox"/>	Flipper Scuba OrderFill4	Running	5:57:04 PM	Check Credit		<input type="checkbox"/> credit	250.00	Jane Doe

instanceDetail.jsp

FUEGO Work Portal Welcome, Robert Adams Search - Options - Help - Logout

Inbox > Industrial Salvage OrderFill3

actiontoolbar.jsp

Details

Process:	Marine Supply Order Fill	Activity:	Check Freight
Priority:	Normal	Status:	Activity completed
Received:	Apr 20, 2004 1:15:18 PM	Deadline:	
Participant:		Copy:	0

Task	Participant	Status	Mandatory	Repeatable	Last execution
Check Freight		Completed	✓		Robert Adams (Apr 20, 2004 1:27:03 PM)

Notes

Description	Date	Participant	Activity
No notes available.			

Attachments

Name	Version	Creator	Filename	Locked by	Action
No attachments available.					

notesinfo.jsp

attachelementsinfo.jsp

Fuego™ - Work Portal

search.jsp

Search OptionsHelp

Search Clear Close

Processes

☐ BAMDashboard
☐ CommercialInformation
☒ CustomerManagement
☐ OrderFulfillmentDashBoard
☐ Stock Administration

Filter Options

Get Instances Assigned To: All
Case Sensitive Matching: ☐

Include Instances

☒ In process
☐ Completed
☐ Aborted

Conditions

Match all of the following ☐

Add Condition: Activity +

filteratributes.jsp

2:09:42 PM Showing 1-2 of 2

Description	Activity	Priority	State	Received	Deadline	Participant
Pet & Pet	Assign Credit Limit To Customer	Normal	Running	Jul 12 05:21:12 PM		
Smith & Smith	Approve Customer	Normal	Running	2:09:27 PM		

searchdetail.jsp

Search Clear Close

Filter Description

Fuego™ - Work Portal

footer.jsp

35

Chapter 5. JSP Descriptions

JSP Descriptions

This section describes each of the main JSP pages that conform the Work Portal. For each page, the KEY in the JSPBundle, the default file name, the parameters used, and a screenshot are exposed.

For each parameter listed, its Java-type is described as well as the key needed to access this parameter.

The following convention is used to specify the parameter keys:

- If the key is all uppercase, it means it is an attribute of the ApplicationConstants class. Example: REQUESTED PROCESS is ApplicationConstants.REQUESTED PROCESS
- If the key is between double quotes, it is a java String. Example: **title**
- If the key includes the class name, it is a static attribute of that class. Example: **FileTypeAssociations.ATTRIBUTE MODEL**

As an example, given the following parameters:

- fuego.papi.Process (REQUESTED PROCESS)
- java.lang.Exception (*authexception*)
- java.util.Hashtable (FileTypeAssociations.ATTRIBUTE MODEL)

here is the Java code that could be used in the JSP to get them:

```
fuego.papi.Process process =  
    (fuego.papi.Process) request.getAttribute
```

```
(ApplicationConstants.REQUESTED_PROCESS);  
Exception exception =  
    (Exception) request.getAttribute("authexception");  
java.util.Hashtable assocHash = (java.util.Hashtable)  
    request.getAttribute(FileTypeAssociations.ATTRIBUTE_MODEL);
```

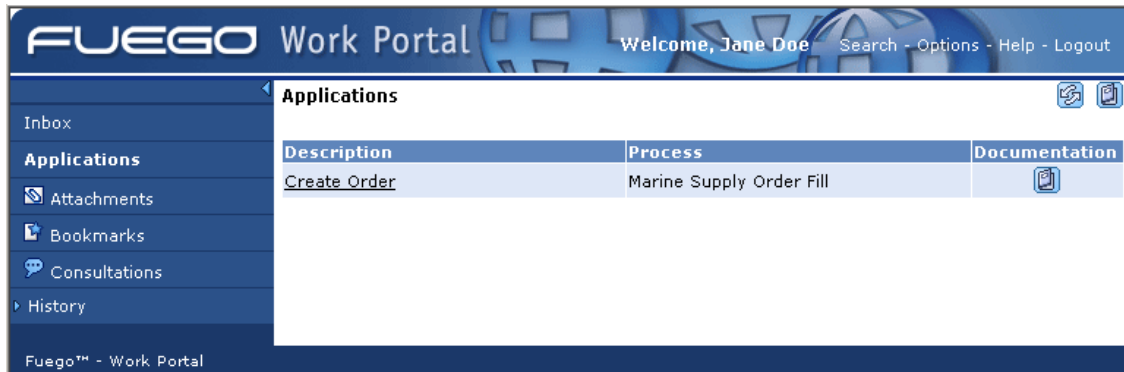
activityDoc.jsp

GrabTest > Check Credit > Documentation

This activity checks if the customer still has available credit for the current order. Takes into account all the pending orders the customer has still to be delivered and the invoices not paid yet.

- KEY: ACTIVITY_DOCUMENTATION
- Default filename:/jsp/activityDoc.jsp
- Parameters:
 - java.lang.String (PROCESS_NAME)
 - java.lang.String (ACTIVITY_NAME)
 - java.lang.String (ACTIVITY_DOC_PATH)

applicationsView.jsp








- KEY: APPLICATIONS_VIEW
- Default filename: /jsp/applicationsView.jsp
- Parameters:
 - fuego.papi.ApplicationsView (REQUESTED_VIEW)
 - fuego.papi.Activity[]
(REQUESTED_APPLICATIONS_FOR_CURRENT_VIEW)
 - java.lang.String[]
(REQUESTED_ACTIVE_PROCESSES_FOR_CURRENT_VIEW)

attachmentsView.jsp



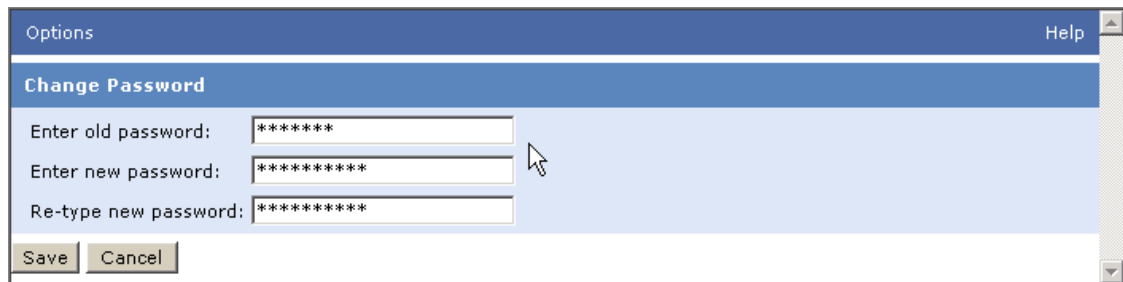
- KEY: ATTACHMENTS_VIEW
- Default filename: /jsp/attachmentsView.jsp
- Parameters:
 - fuego.papi.AttachmentsView (REQUESTED_VIEW)

auditTrail.jsp

Audit trail Help				
Marine Supply Order Fill > Check Credit > Industrial Salvage OrderFill4				
Activity	Event	Responsible	Date	Copy
<input type="checkbox"/> create	 Completed		Dec 22, 2003 5:43:33 PM	0
	Creation	Jane Doe	Dec 22, 2003 5:43:33 PM	0
<input type="checkbox"/> ReviewOrder	 Completed		Dec 22, 2003 5:44:02 PM	0
	Enter	Account Manager	Dec 22, 2003 5:44:02 PM	0
	Item Execution 	John Smith	Dec 22, 2003 5:44:20 PM	0
	Exit	John Smith	Dec 22, 2003 5:44:24 PM	0
<input type="checkbox"/> Check Credit	 Processing		Dec 22, 2003 5:44:24 PM	0
	Enter	John Smith	Dec 22, 2003 5:44:24 PM	0
	Item Execution 	Mary Jones	Dec 22, 2003 5:45:42 PM	0
<input type="button" value="Close"/>				
Fuego™ - Work Portal				

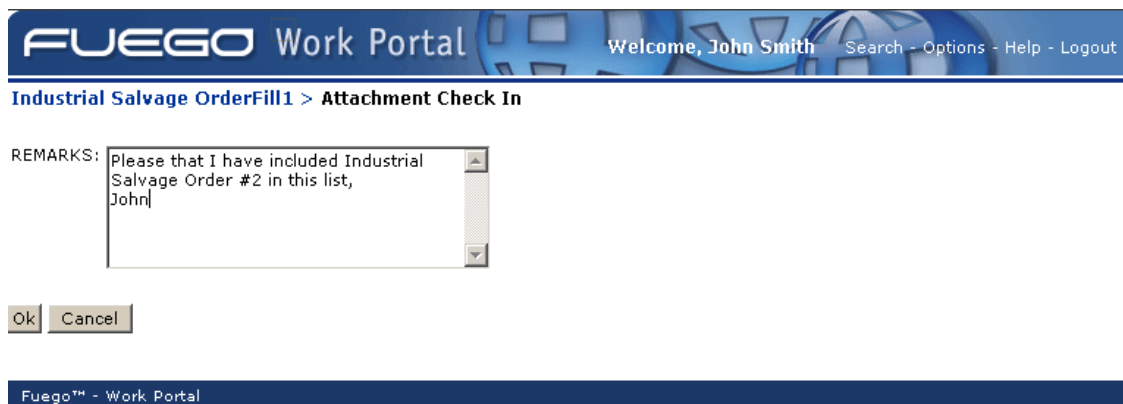
- KEY: AUDIT_TRAIL
- Default filename: /jsp/auditTrail.jsp
- Parameters:
 - fuego.papi.InstanceInfo (REQUESTED_INSTANCE)
 - fuego.papi.utils.AuditTrail (EVENTS)
 - java.lang.String (NODE_ID)

changePassword.jsp

A screenshot of a web form titled "Change Password". The form has a blue header bar with "Options" on the left and "Help" on the right. Below the header, the title "Change Password" is displayed. The form contains three input fields: "Enter old password:" with a masked value "*****", "Enter new password:" with a masked value "*****", and "Re-type new password:" with a masked value "*****". A mouse cursor is pointing at the "Enter new password:" field. At the bottom of the form are two buttons: "Save" and "Cancel".

- KEY: CHANGE_PASSWORD_OPTION
- Default filename: /jsp/changePassword.jsp
- Parameters:
 - java.lang.String (REQUESTED_ERROR)

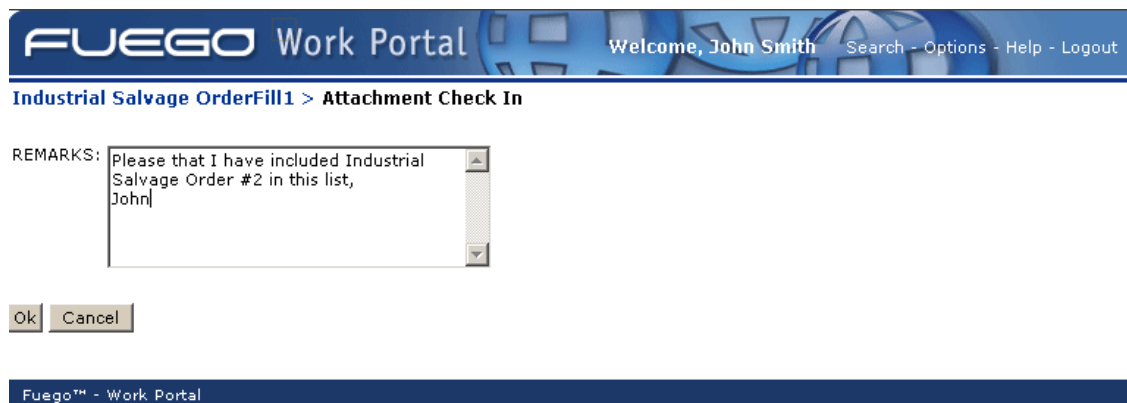
checkinAttachment.jsp

A screenshot of a web form titled "checkinAttachment.jsp". The form is part of the "FUEGO Work Portal" and shows a user "John Smith" logged in. The breadcrumb trail is "Industrial Salvage OrderFill1 > Attachment Check In". The form has a "REMARKS:" label and a text area containing the text "Please that I have included Industrial Salvage Order #2 in this list, John". Below the text area are "Ok" and "Cancel" buttons. The footer of the page says "Fuego™ - Work Portal".

- KEY: CHECK_IN_INSTANCE_ATTACHMENT
- Default filename: /jsp/checkinAttachment.jsp

- Parameters:
 - fuego.papi.InstanceInfo (REQUESTED_INSTANCE)
 - fuego.papi.Attachment (V_ATTACHMENT)
 - java.lang.String (REMARKS)
 - java.util.Hashtable (ATTACHMENT_SERVLET_ERRORS)

checkinWorkingAttachment.jsp



FUEGO Work Portal Welcome, John Smith Search - Options - Help - Logout

Industrial Salvage OrderFill1 > Attachment Check In

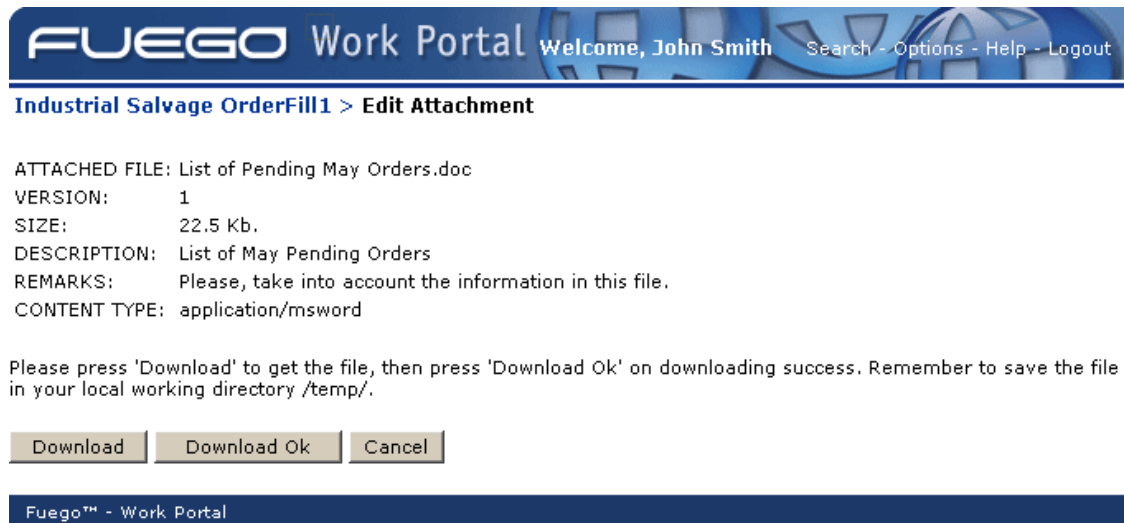
REMARKS: Please that I have included Industrial Salvage Order #2 in this list, John

Ok Cancel

Fuego™ - Work Portal

- KEY: CHECK_IN_WORKING_ATTACHMENT
- Default filename:/jsp/checkinWorkingAttachment.jsp
- Parameters:
 - fuego.papi.Attachment (V_ATTACHMENT)
 - java.lang.String (REMARKS)
 - java.util.Hashtable (ATTACHMENT_SERVLET_ERRORS)

editAttachment.jsp



- KEY: EDIT_INSTANCE_ATTACHMENT
- Default filename: /jsp/editAttachment.jsp
- Parameters:
 - fuego.papi.InstanceInfo (REQUESTED_INSTANCE)
 - fuego.papi.Attachment (V_ATTACHMENT)

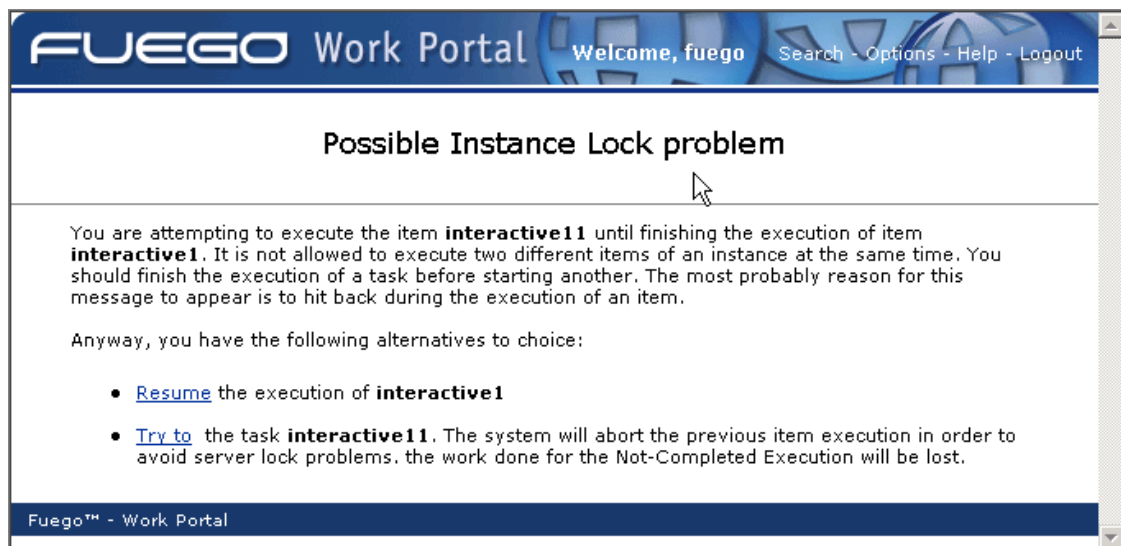
editOptions.jsp

Options		Help
User Information		
Full Name:	fuego Fuego	
Login Name:	fuego	
E-mail:		
PASSWORD:	Change Password	
Browser settings		
Enable Flash version menu:	<input type="checkbox"/>	
Enable DHTML support:	<input checked="" type="checkbox"/>	
Settings		
Sort instances by:	Received	
Instances order:	Ascending	
Show hidden views:	<input type="checkbox"/>	
Follow the Instance:	<input type="checkbox"/>	
Notify me by e-mail when new instances arrive:	<input type="checkbox"/>	
Keep instance view:	<input type="checkbox"/>	
Enable applet for attachment management:	<input type="checkbox"/>	
Enable remote scripting for FuegoObject presentations:	<input checked="" type="checkbox"/>	
Show applications:	In a folder	
User Working Directory:	<input type="text" value="/temp/"/>	
	(Including last path separator, ie.: 'c:\temp\').	
Maximum number of searches in history:	10	
Display options		
Number of instances:	10	
Language:	English	
Country:	United States	
TimeZone:	GMT-03:00	
<input type="button" value="Save"/> <input type="button" value="Close"/>		

- KEY: EDIT_OPTIONS
- Default filename: /jsp/editOptions.jsp
- Parameters:

- java.lang.String[] (FT_TIME_ZONES)
- boolean (ATTR_PART_AUTOLOGIN)
- boolean (IS_CHANGE_PASSWORD_SUPPORTED)
- fuego.papi.Presentation.Column[] (DEFAULT_COLUMNS)

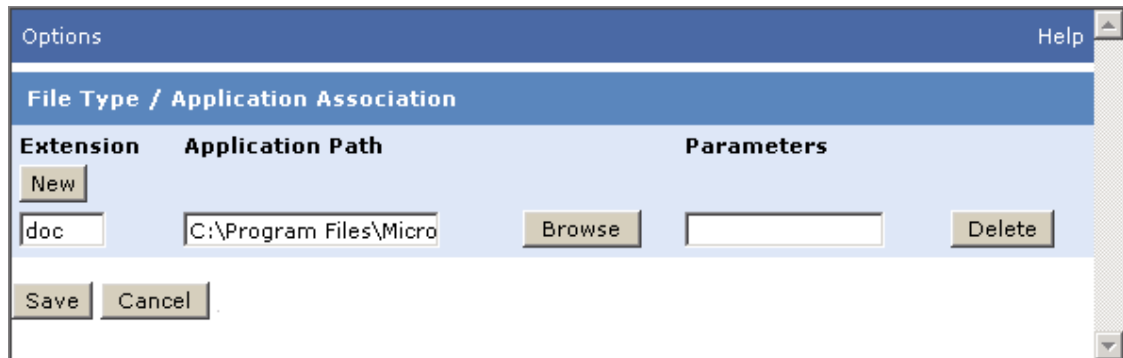
executionLock.jsp



- KEY: EXECUTION_LOCK
- Default filename: /jsp/executionLock.jsp
- Parameters:
 - java.lang.String (OLD_ITEM_KEY)
 - java.lang.String (OLD_ITEM_DESCR)
 - java.lang.String (NEW_ITEM_KEY)

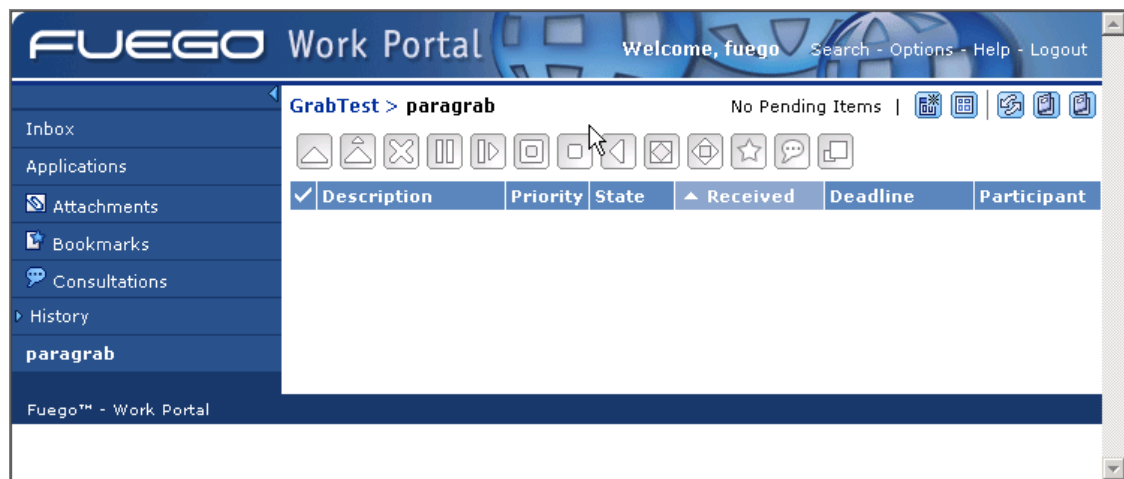
- `java.lang.String` (`NEW_ITEM_DESCR`)

fileTypeAssociations.jsp



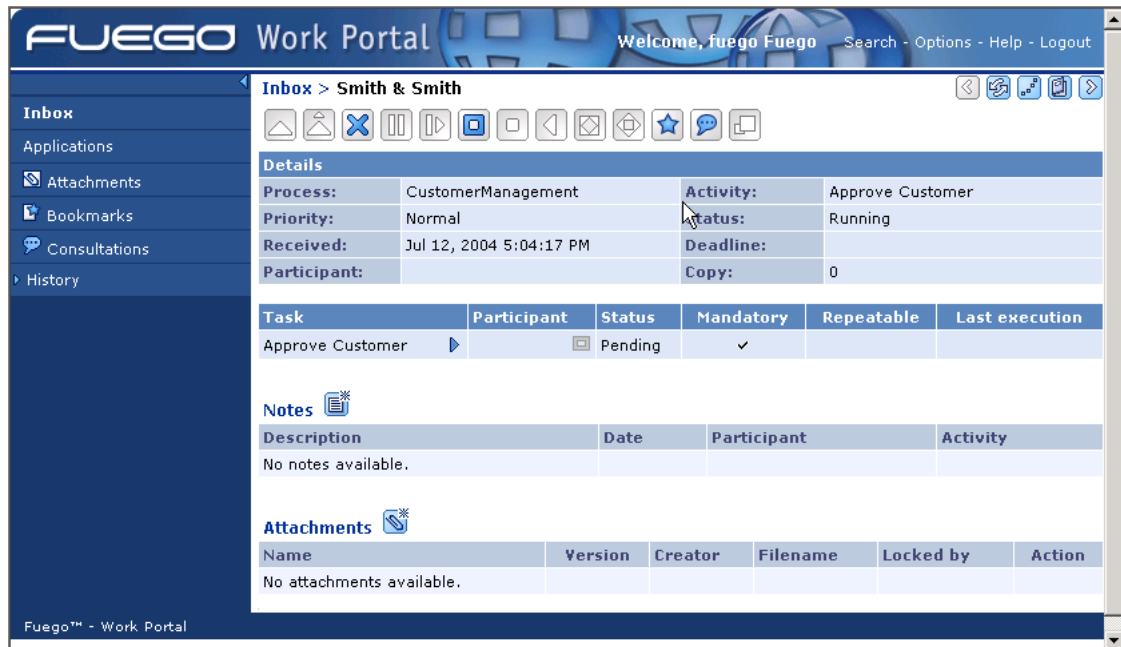
- KEY: `FILE_TYPE_ASSOCIATION_OPTION`
- Default filename: `/jsp/fileTypeAssociations.jsp`
- Parameters:
 - `java.util.Hashtable` (`FileTypeAssociations.ATTRIBUTE_MODEL`)

folder.jsp



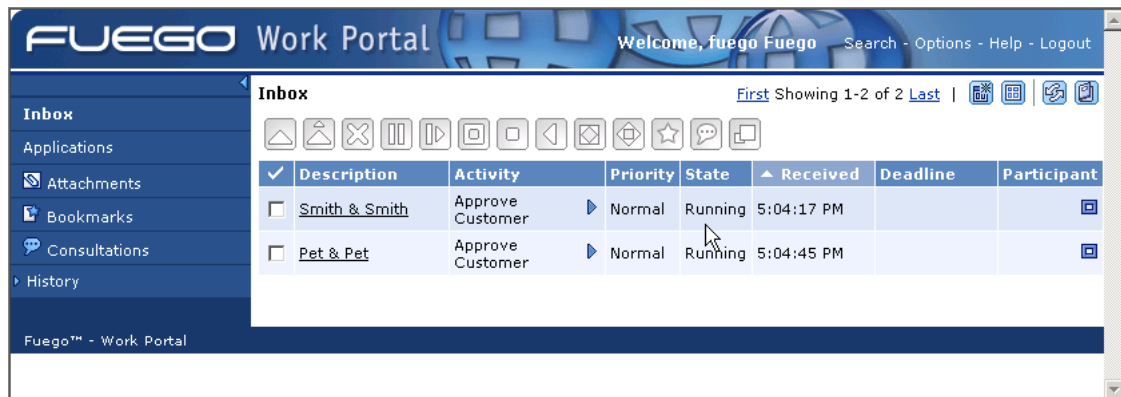
- KEY: FOLDER
- Default filename: /jsp/folder.jsp
- Parameters:
 - fuego.papi.View (REQUESTED_VIEW)

instanceDetail.jsp



- KEY: INSTANCE_DETAIL
- Default filename: /jsp/instanceDetail.jsp
- Parameters:
 - List (PROCESSES_WITH_SHOW_IMAGE_ACTIVITY)
 - fuego.papi.Process (REQUESTED_PROCESS)
 - fuego.papi.Activity (REQUESTED_ACTIVITY)
 - fuego.papi.InstanceInfo[] (REQUESTED_INSTANCES_COMPLETE_SET)
 - fuego.papi.InstanceInfo (REQUESTED_INSTANCE)
 - fuego.papi.View (REQUESTED_VIEW)

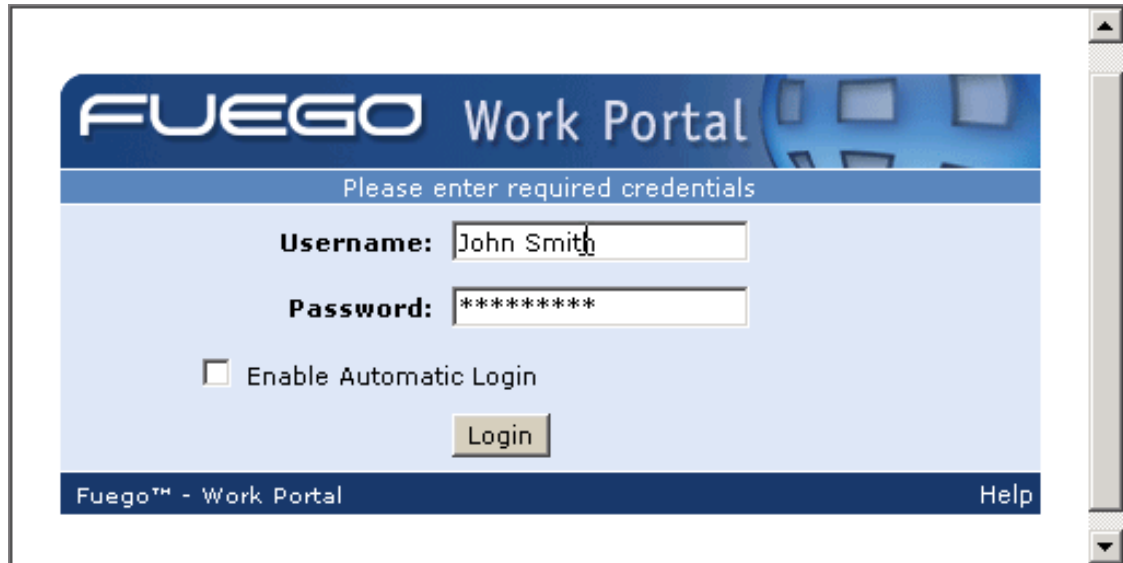
instancesView.jsp



- KEY: INSTANCES_VIEW
- Default filename: /jsp/instancesView.jsp
- Parameters:
 - List (PROCESSES_WITH_SHOW_IMAGE_ACTIVITY)
 - fuego.papi.Process (REQUESTED_PROCESS)
 - fuego.papi.Activity (REQUESTED_ACTIVITY)
 - fuego.papi.InstanceInfo[] (REQUESTED_INSTANCES)
 - fuego.papi.View (REQUESTED_VIEW)
 - boolean (SHOW_PARAMETERS)
 - fuego.papi.Presentation (VIEW_PRESENTATION)
 - java.lang.Integer (REQUESTED_INSTANCE_FROM)
 - java.lang.Integer (REQUESTED_INSTANCE_TO)
 - java.lang.Integer (REQUESTED_INSTANCES_COUNT)
 - java.lang.String (SORT_BY)

- java.lang.String (SORT_ORDER)

login.jsp

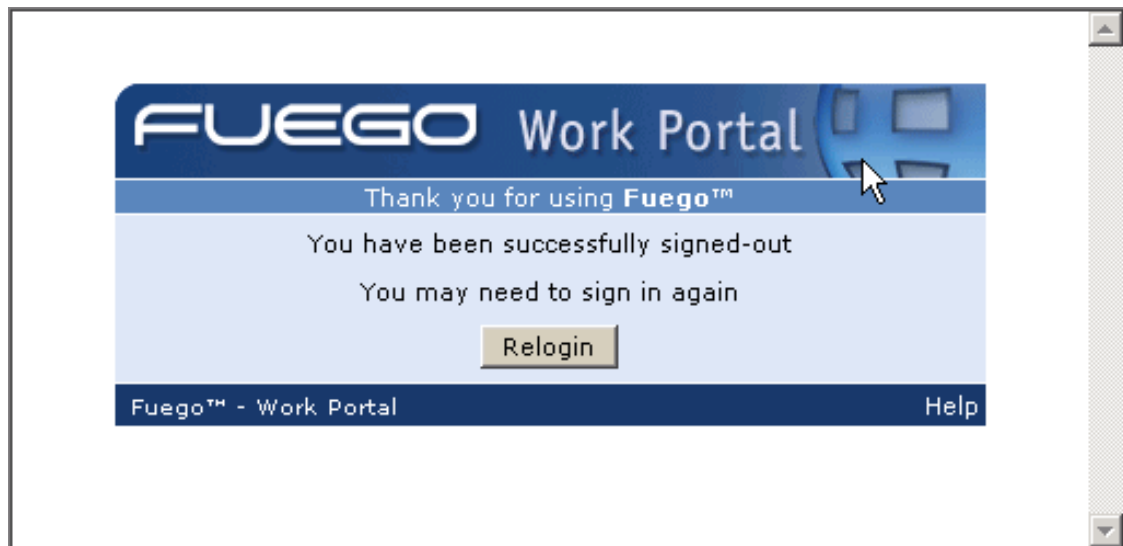


The screenshot shows a web browser window displaying the 'FUEGO Work Portal' login page. The page has a blue header with the 'FUEGO' logo and 'Work Portal' text. Below the header, a blue bar contains the text 'Please enter required credentials'. The main content area is light blue and contains two input fields: 'Username:' with the value 'John Smith' and 'Password:' with the value '*****'. Below these fields is a checkbox labeled 'Enable Automatic Login' which is unchecked. A 'Login' button is positioned below the checkbox. At the bottom of the page, a dark blue footer bar contains the text 'Fuego™ - Work Portal' on the left and 'Help' on the right. The browser window has a standard scrollbar on the right side.

- KEY: LOGIN
- Default filename: /jsp/login.jsp
- Parameters:
 - java.lang.String (USERNAME)
 - java.lang.String (ORIGINAL_ABSOLUTE_URL)
 - java.lang.String (POST_STATE)
 - java.lang.String (ATTR_LOGIN_ERROR_MSG)
 - java.lang.String (ATTR_LOGIN_ERROR_DETAIL)

- java.lang.String (ATTR_LOGIN_ERROR_TECH_DETAIL)
- java.lang.String (ATTR_LOGIN_ERROR_HAS_DETAIL)
- java.lang.String (ATTR_LOGIN_ERROR_HAS_TECH_DETAIL)
- boolean (DEVELOPMENT_ENVIROMENT)

logout.jsp



- KEY: LOGOUT
- Default filename: /jsp/logout.jsp
- Parameters: NONE

newAttachment.jsp

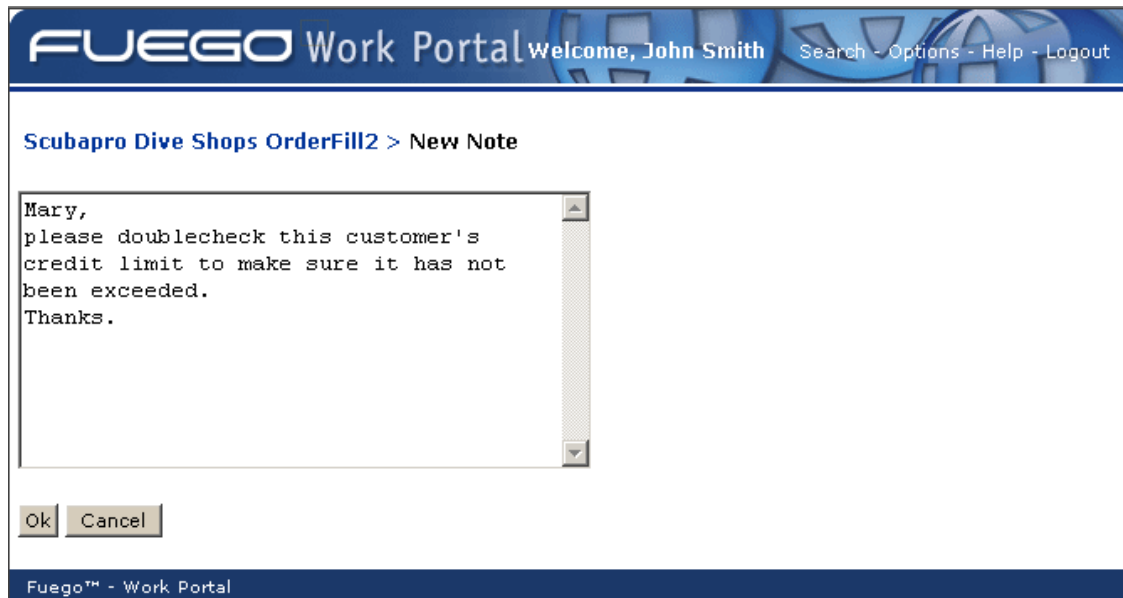
The screenshot shows a web browser window with the title 'FUEGO Work Portal'. The navigation bar includes 'Welcome, John Smith', 'Search', 'Options', 'Help', and 'Logout'. The main heading is 'Industrial Salvage OrderFill1 > New Attachment'. The form contains the following fields and controls:

- ATTACH FILE:** A text box with the path 'C:\Documents And Settings\My Documents\' and a 'Browse...' button. A checked checkbox labeled 'Advanced options' is to the right.
- CONTENT TYPE:** Two radio buttons: 'auto-detect' (unselected) and 'select from list:' (selected). The 'select from list:' option is followed by a dropdown menu showing 'plain text (text/plain)' and a text box containing '/application/msword'.
- YOU CAN BROWSE TO SELECT THE FILE.** A text label.
- DESCRIPTION:** A text box containing 'List of Pending May Orders'.
- REMARKS:** A text area containing 'Please, take into account the information in this file.' with a vertical scrollbar.
- Buttons:** 'Ok' and 'Cancel' buttons at the bottom left.

The footer of the page reads 'Fuego™ - Work Portal'.

- KEY: NEW_INSTANCE_ATTACHMENT
- Default filename: /jsp/newAttachment.jsp
- Parameters:
 - fuego.papi.InstanceInfo (REQUESTED_INSTANCE)
 - java.lang.String (TITLE)
 - java.lang.String (REMARKS)
 - java.util.Hashtable (ATTACHMENT_SERVLET_ERRORS)


newNote.jsp



The screenshot shows a web application interface for 'FUEGO Work Portal'. The header bar is blue with the text 'FUEGO Work Portal' and 'Welcome, John Smith'. To the right of the header are links: 'Search - Options - Help - Logout'. Below the header, the breadcrumb path 'Scubapro Dive Shops OrderFill2 > New Note' is displayed. The main content area contains a text input field with the text: 'Mary, please doublecheck this customer's credit limit to make sure it has not been exceeded. Thanks.'. Below the text field are two buttons: 'Ok' and 'Cancel'. At the bottom of the page, a blue footer bar contains the text 'Fuego™ - Work Portal'.

- KEY: NEW_NOTE
- Default filename: /jsp/newNote.jsp
- Parameters:
 - fuego.papi.IstanceInfo (REQUESTED_INSTANCE)
 - fuego.papi.Activity (REQUESTED_ACTIVITY)
 - java.lang.String (ACTIVITY_ID)

participantList.jsp

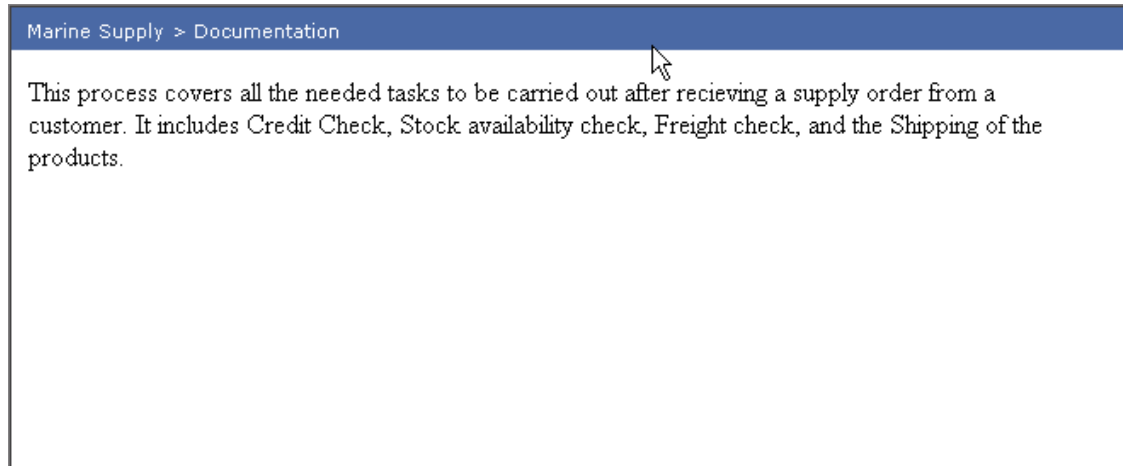


A screenshot of a web browser window showing a table with three columns: Name, UID, and E-mail. The table contains five rows of data. The first four rows have blue hyperlinks in the Name column. The E-mail column is empty for the first four rows and contains the text 'null' for the fifth row.

Name	UID	E-mail
Ann Parker	Ann Parker	
Fuego Fuego	fuego	
John Smith	John Smith	
Sam Baldwin	Sam Baldwin	
sa	sa	null

- KEY: PARTICIPANT_LIST
- Default filename: /jsp/participantList.jsp
- Parameters:
 - java.lang.String (CN)
 - java.lang.String (HIDDEN_CN)
 - java.lang.String (UID)
 - fuego.papi.Participant[]
(REQUESTED_PARTICIPANT_ITERATOR)
 - java.lang.String (PARTICIPANT_SORT_BY)
 - java.lang.String (PARTICIPANT_SORT_ORDER)
 - java.lang.String (MATCHING_NAME)

processDoc.jsp



- KEY: PROCESS_DOCUMENTATION
- Default filename:/jsp/processDoc.jsp
- Parameters:
 - java.lang.String (PROCESS_NAME)
 - java.lang.String (PROCESS_DOC_PATH)

search.jsp

Search Options
Help

Search Clear Close

Processes

☐ BAMDashBoard
☐ CommercialInformation

☒ CustomerManagement
☐ OrderFulfillmentDashBoard

☐ Stock Administration

Filter Options

Get Instances Assigned To: All

Case Sensitive Matching: ☐

Include Instances

☒ In process
☐ Completed
☐ Aborted

Conditions

Match all of the following ☐

Add Condition: Activity +

2:09:42 PM
Showing 1-2 of 2

Description	Activity	Priority	State	Received	Deadline	Participant
<u>Pet & Pet</u>	Assign Credit Limit To Customer	Normal	Running	Jul 12 05:21:12 PM		
<u>Smith & Smith</u>	Approve Customer	Normal	Running	2:09:27 PM		

Search Clear Close

Filter Description

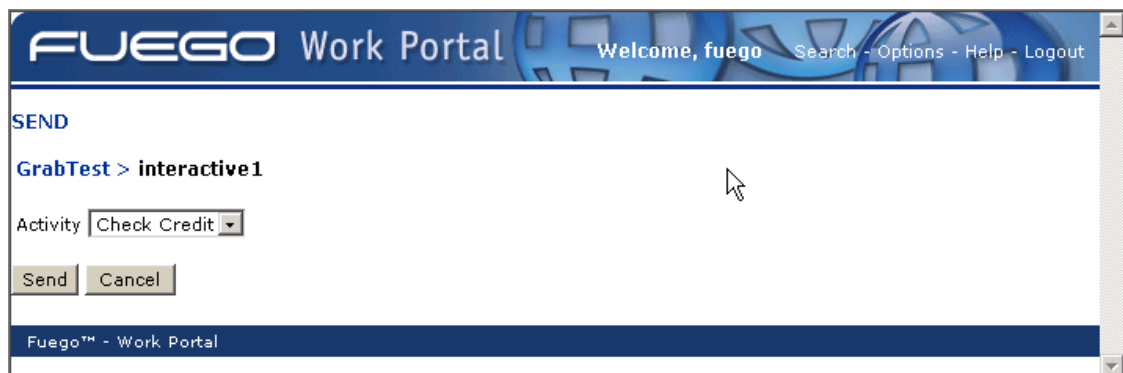
Fuego™ - Work Portal

- KEY: SEARCH
- Default filename:/jsp/search.jsp
- Parameters:
 - java.lang.Boolean (SEARCH_INCLUDE_ABORTED_INSTANCES)
 - java.lang.Boolean (SEARCH_INCLUDE_COMPLETED_INSTANCES)

- java.lang.Boolean
(SEARCH_INCLUDE_IN_PROCESS_INSTANCES)
- java.lang.String[] (REQUESTED_SORTED_VIEWID_ARRAY)
- java.lang.String (SHOW_RESULTS)
- java.lang.String (SHOW_PRESENTATION)
- java.lang.String (NEW_PRESENTATION_ID)
- java.lang.String (VIEW_ID_CHANGED)
- java.lang.String (OLD_VIEW_ID)
- java.lang.ArrayList (REQUESTED ACTIVITY LIST)
- java.lang.String (IS_HIDDEN)
- fuego.papi.View (REQUESTED VIEW)
- java.lang.String (SEARCH_VIEW_ID)
- java.lang.String (PARENT_VIEW_ID)
- fuego.papi.Filter (REQUESTED_FILTER)
- java.lang.String[] (SELECTED_PROCESSES)
- java.lang.Integer (REQUESTED_INSTANCE_FROM)
- java.lang.Integer (REQUESTED_INSTANCE_TO)
- java.lang.Integer (REQUESTED_INSTANCES_COUNT)
- java.lang.String (SORT_BY)
- java.lang.String (SORT_ORDER)
- fuego.papi.Presentation (REQUESTED_PRESENTATION)
- fuego.papi.Process [] (REQUESTED_PROCESS_ITERATOR)

- fuego.papi.Presentation [] (AVAILABLE_PRESENTATIONS)
- fuego.papi.Presentation.Column [] (AVAILABLE_COLUMNS)
- fuego.papi.Presentation.Column [] (SELECTED_COLUMNS)
- java.util.ArrayList (SELECTED_COLUMNS_IDS)
- java.lang.String (PROCESS_IS_FIXED)
- fuego.papi.Process (REQUESTED_PROCESS)
- fuego.papi.VarDefinition[] (PROCESS_VARS)

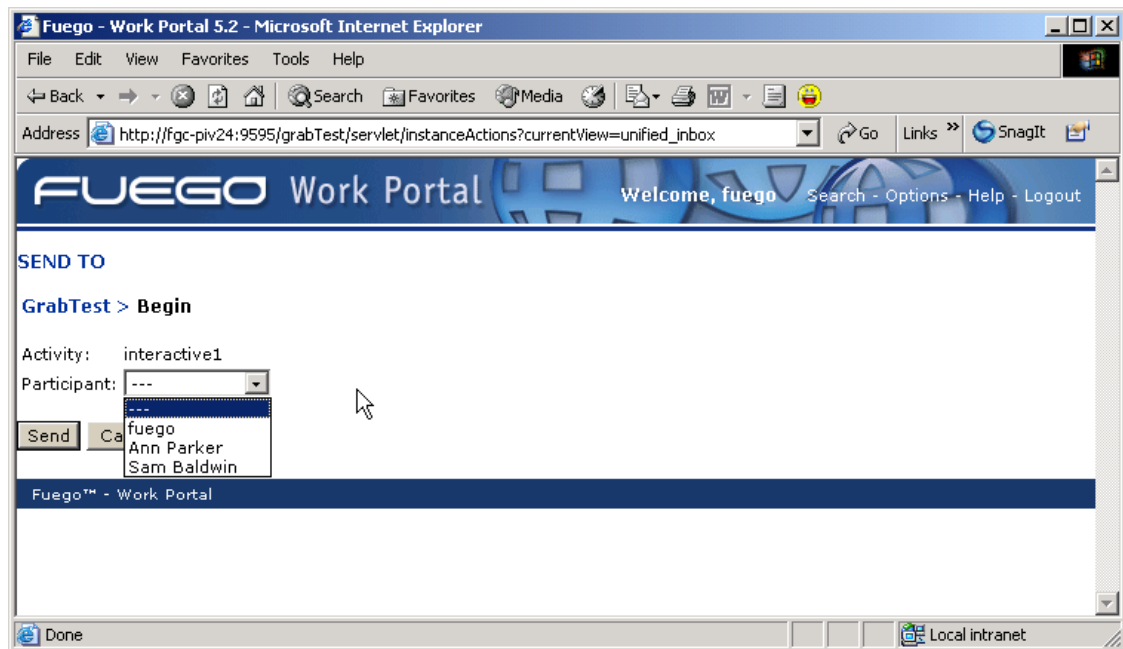
send.jsp



- KEY: SEND
- Default filename: /jsp/send.jsp
- Parameters:
 - fuego.papi.InstanceInfo (REQUESTED_INSTANCE)
 - fuego.papi.Activity (REQUESTED_ACTIVITY)

- java.util.List (TARGET_ACTIVITIES)
- fuego.papi.Process (REQUESTED_PROCESS)
- java.lang.String (PROCESS_NAME)
- java.lang.String (ACTIVITY_ID)
- java.lang.String (INSTANCE_STAMP_ID)
- java.lang.String (NEXT_INSTANCE_STAMP_ID)

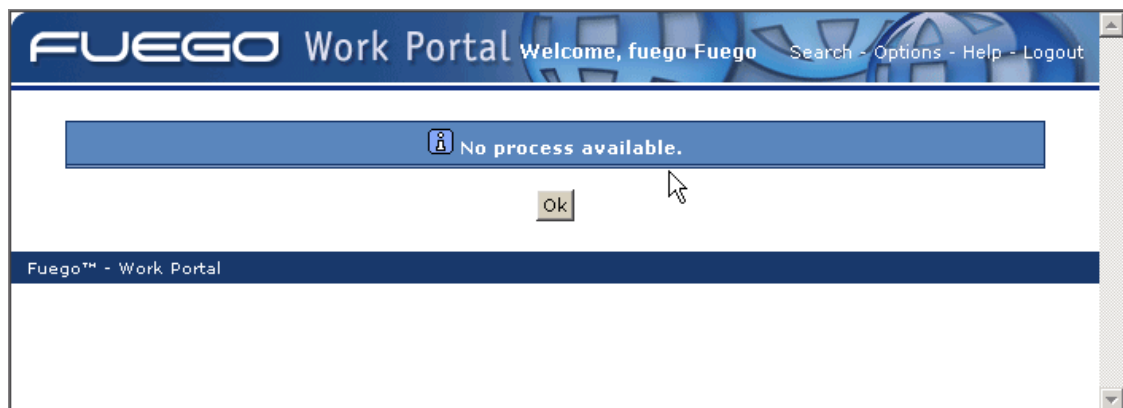
sendTo.jsp



- KEY: SEND_TO
- Default filename: /jsp/sendTo.jsp
- Parameters:

- fuego.papi.InstanceInfo (REQUESTED_INSTANCE)
- fuego.papi.Activity (REQUESTED_ACTIVITY)
- fuego.papi.ParticipantsForActivities (TARGET_PARTICIPANTS)
- fuego.papi.Process (CURRENT_PROCESS)
- java.lang.String (PROCESS_NAME)
- java.lang.String (ACTIVITY_ID)
- java.lang.String (INSTANCE_STAMP_ID)
- java.lang.String (NEXT_INSTANCE_STAMP_ID)

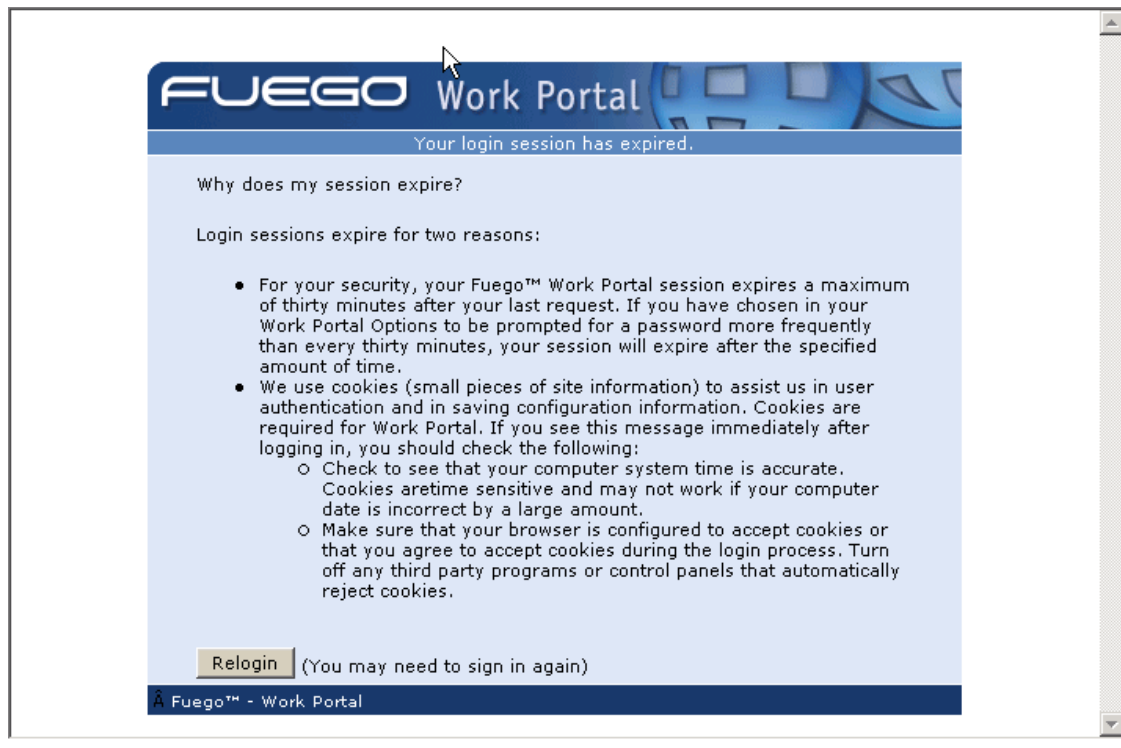
servererror.jsp



- KEY: SERVER_ERROR
- Default filename: /jsp/servererror.jsp
- Parameters:
 - java.lang.Integer (TYPE)

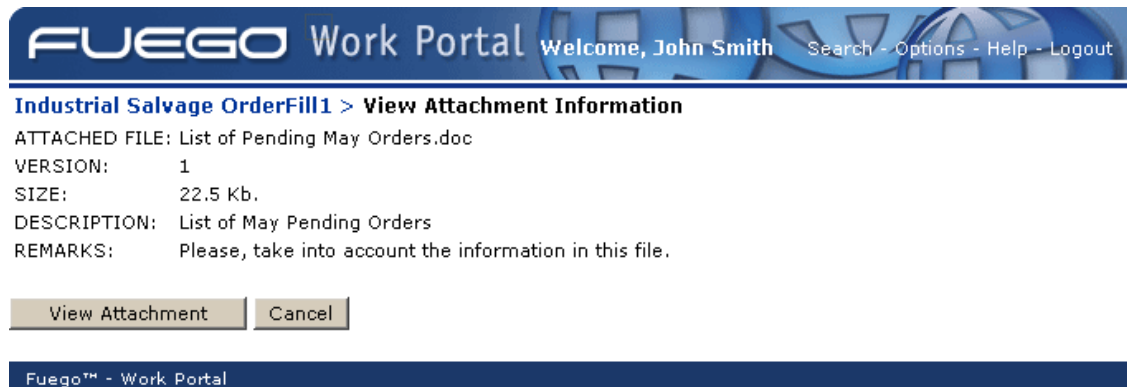
- java.lang.String (REF)
- java.lang.String (METHOD)
- java.lang.String (PRESERVE_STATE)
- java.lang.Boolean (ONLY_BACTH_ERROR_LIST)
- java.lang.String (DETAIL)
- java.lang.Boolean (HAS_DETAIL)
- java.lanf.String (HIDDEN_EXCEPTION_INFO)
- java.lang.Boolean (IS_BATCH_OPERATION_EXCEPTION)
- java.util.HashMap (REQ_HASH_ERROR)

sessionExpired.jsp



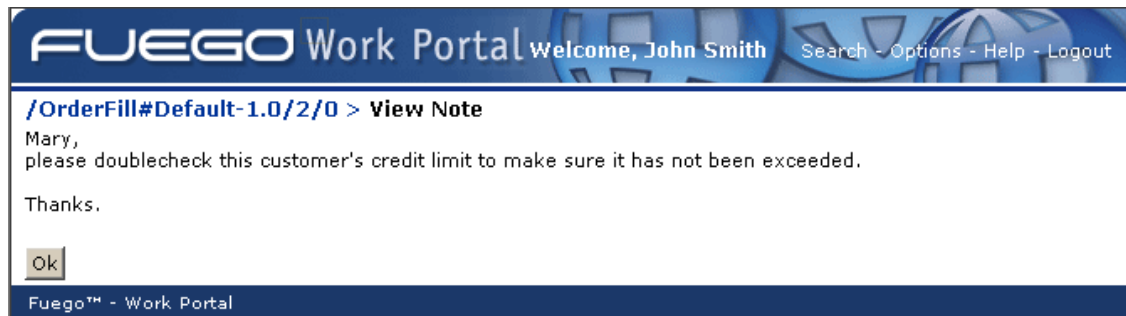
- KEY: SESSION_EXPIRED
- Default filename: /jsp/sessionExpired.jsp
- Parameters:
 - java.lang.String (ORIGINAL_ABSOLUTE_URL)
 - java.lang.String (REF)
 - java.lang.String (PRESERVE_STATE)

viewAttachment.jsp



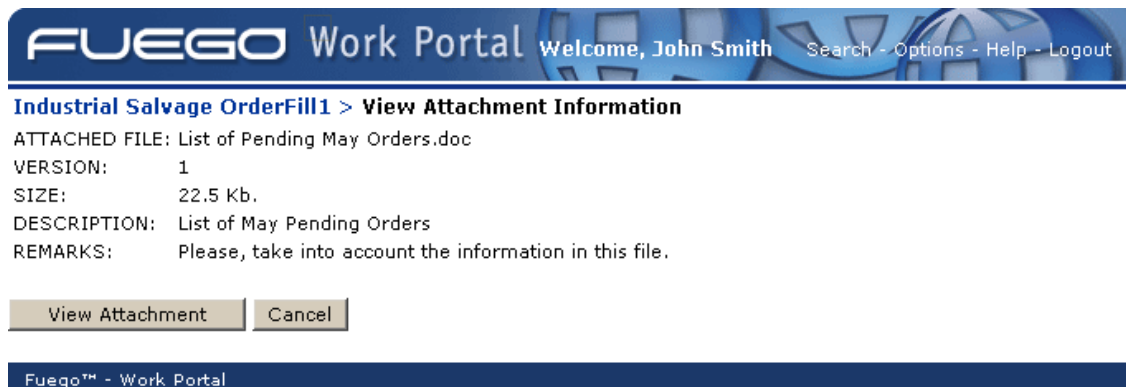
- KEY: VIEW_INSTANCE_ATTACHMENT
- Default filename: /jsp/viewAttachment.jsp
- Parameters:
 - fuego.papi.InstanceInfo (REQUESTED_INSTANCE)
 - fuego.papi.Attachment (V_ATTACHMENT)

viewNote.jsp



- KEY: VIEW_NOTE
- Default filename: /jsp/viewAttachment.jsp
- Parameters:
 - java.lang.String (REQUESTED_REMARK)
 - fuego.papi.InstanceInfo (REQUESTED_INSTANCE)

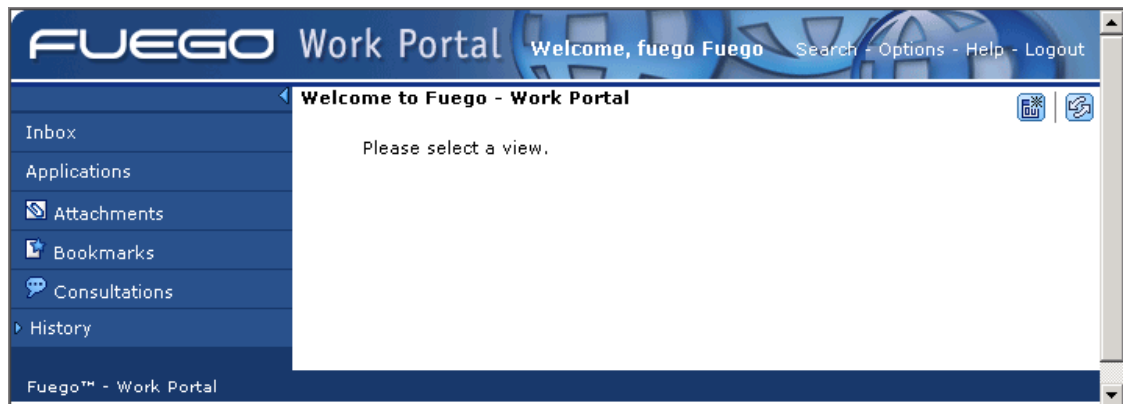
viewWorkingAttachment.jsp



- KEY: VIEW_WORKING_ATTACHMENT

- Default filename: /jsp/viewWorkingAttachment.jsp
- Parameters:
 - fuego.papi.Attachment (V_ATTACHMENT)

welcome.jsp



- KEY: WELCOME
- Default filename: /jsp/welcome.jsp
- Parameters: NONE