Extensible Administration Console

This document contains the following sections:

Introduction

The Extensible Administration Console is a new Administration Console designed to mirror the flexibility of Geronimo's architecture. While the previous console was static, allowing configuration only for pre-defined components, the new Extensible Console changes dynamically with the components installed on your server.

This framework allows Geronimo plugin developers to package extensions to the Administration Console (called ACEs) with their components. On installation of a plugin, this new content will automatically be added to the Extensible Administration Console, so that the user can manage all of the configuration and tools from one place.

Objective

After reading this document, a Geronimo user will be able to

1. Install the Extensible Administration Console into a Geronimo minimal assembly (Tomcat or Jetty)
2. Download and install an Administrative Console Extension

Scope

This document covers installing Geronimo's Administration Console into a Geronimo minimal assembly and the process to install an ACE plugin. For more information about the underlying architecture and ACE development, see the Administration Console Extension Developer's Guide.

Who should read this document

These instructions are geared towards a basic user of Geronimo who wants to install new components on the administration console. Knowledge of the internal workings of the server is not necessary.

Assumptions

The reader should be familiar with the Geronimo application server and its applications. The reader should also be familiar with the basics of Geronimo's plugin framework. For more information about the architecture, this guide may also be a valuable resource: http://media.wiley.com/product_ancillary/31/04717854/DOWNLOAD/Pro_Apache_Geronimo_ch17.pdf.
The reader should have Apache Geronimo 2.1 Minimal ("Little-G") installed and running, with either Tomcat6 or Jetty6 as the web container. The Java EE assemblies already have the Extensible Aministration Console installed. For more information about getting started or getting updated to the right version, see this website: http://cwiki.apache.org/GMOxDOC21/installation.html.

Installation Planning

This section will prepare the user to install an ACE by providing the necessary terminology and introducing the installation scenarios that will be discussed.

Common Terminology

Plugin - An archive file (.car or .war) that can be installed into Geronimo to install a specific new service, such as ActiveMQ. For more about plugins, or to look at available plugins, check out geronimoplugins.com or geronimoplugincentral.org.
.CAR file (Configuration ARchive) - An archive file that stores Geronimo-specific configurations, as well as the classes, libraries, web pages, and other information associated with an application.
.WAR file (Web ARchive)- An archive file that contains a web application, including all of its classes, libraries, HTML and JSP pages, and other information. It can be deployed on any Java Enterprise compatible servlet container.
ACE (Administration Console Extension) - An archive file (either a .car or a .war), that includes Administration Console portlets. These portlets will be added to the Extensible Administration Console when the ACE is activated.
Extensible Administration console - A flexible version of Geronimo's original administration console. Once installed it is available at http://localhost:8080/console, and includes some portlets that correspond to the currently activated services in Geronimo.
Service - a component or set of functionality for Geronimo - it may be pre-installed, such as the Tomcat or Jetty web container, or it may be installed as a plugin
Minimal console - The administration console as it is first installed - with only the services necessary for basic functionality
Portlet - A web user interface component that can be assembled together with other similar components to create a web (portal) page. See the Portlet Specification JSR 168.

Installation Scenarios

1. Installing the extensible administration console into a Geronimo minimal assembly
2. Installing an ACE in the .war format from disk
3. Installing an ACE in the .car format from an online repository

Which kind of file format do I want ?

- **Need console extensions for a component you already have?** If you are planning to add Administration Console portlets for a component that is already installed on your server, a .war file is simplest. A .war-format ACE has only the needed portlets, and is dependent on the component already being installed.
- **Need the component and its extensions?** If you need to install both a new component and its associated Admin Console portlets, a .car file is the right choice. This will look up and download all the necessary components and dependencies, and will also install the new ACE file. A .car file can also be used if the component is already installed.
- **Not sure?** If you aren't sure that you have all the pre-requisites installed, a .car is the safest option. This will work regardless of whether the component is pre-installed or not.

What about installing a .car file from disk?

Unfortunately, it is not possible to install a .car file from disk with the current Geronimo Administration Console. However you can still use the "deploy install-plugin" tool from the command line. For more information, read up on the deployer tool at http://cwiki.apache.org/GMOxDOC11/deployer-tool.html

Installing the Extensible Administration Console

This section covers all the necessary steps to install the new Extensible Administration Console on your Geronimo 2.1 Minimal Server. The Geronimo 2.1 Java EE servers already have the console installed.

Prerequisites

1. Geronimo 2.1 - Minimal is installed ("Little-G")
2. No other Geronimo administration console is currently installed

Install as a plugin

1. Start Geronimo
2. Use the deploy/list-plugins command to download and install the admin console from the online plugin catalog. Depending on the web container support in your minimal server, choose:

```
./gsh deploy/list-plugins -r http://geronimo.apache.org/plugins/geronimo-2.1/ org.apache.geronimo.plugins/console-tomcat/2.1/car
```

or

```
./gsh deploy/list-plugins -r http://geronimo.apache.org/plugins/geronimo-2.1/ org.apache.geronimo.plugins
/console-jetty/2.1/car
```

Plugin Directory not updated with released plugins

The plugins directory (http://geronimo.apache.org/plugins/geronimo-2.1/) has not yet been updated with the released versions of the 2.1 plugins.

Once updated, we need to verify this (and subsequent) steps. There may be a dependency issue with Pluto...

Now you can point your web browser to http://localhost:8080/console and see the newly installed console.

Installing an ACE in .war format from an archive file

An ACE in .war format will install only the console extension. Its pre-requisite must already be installed on your system. For example, if you were installing a console extension to monitor a Derby database, Derby would need to be pre-installed on your server.
Prerequisites

1. You have a .war-format ACE to be installed.
2. The Extensible Administration Console is already deployed, as described above.
3. Any pre-requisites for the ACE are already installed.
   Instructions
4. Open the administrative console in an internet browser at address http://localhost:8080/console.
5. Select Deploy New on the left navigation bar.
6. Click browse and navigate to the ACE file.
7. Click Deploy.
8. Refresh your browser. Your new component will show up on the left side navigation menu.

Installing an Administration Console Extension in .car format from a repository

An ACE in the .car format installs a component and adds its configuration portlets to the Extensible Administration Console.
Prerequisites

1. Your plugin provider already has specified a repository URL. For a larger selection of available plugins, checking http://geronimo.apache.org/plugins/geronimo-2.1/repository/ is a good place to start.
2. The Extensible Administration Console already installed, as described in the previous section.

Instructions

1. Access the Extensible Administration Console by pointing a web browser to the following address http://localhost:8080/console.
2. Select Plugins on the left navigation bar.
3. Under Install Geronimo Plugins, choose the plugin's repository address.

How do I choose the repository address?

a. If the correct address does not appear in the Repository list, select Add Repository.
b. In the New Repository textbox, enter the repository's address. Don't forget to include the forward-slash (/) at the end of the address. Select Add Repository.
c. Select Update Repository List. The new address will now appear in the Repository list.

4. With the correct repository displayed in the Repository box, select Show Plugins in selected repository.
5. Choose the plugin from the Available Plugins list by clicking directly on the plugin name.

Where's the plugin I want?

a. If the desired plugin is not a hyperlink, there are a number of potential reasons:
i. If already installed is displayed next to the plugin name, the plugin may already be installed on your server. Check for its name on the System Modules tab to make sure it is running, and to start or uninstall it if necessary.
ii. If Not available is displayed next to the plugin name, it may require a different version of Geronimo or a different web container (Tomcat or Jetty). Select View Details for more information.
b. If the list of Available Plugins does not display, or the desired plugin is not listed, you may have entered the wrong repository address. Try entering it again. If problems continue, contact the plugin provider or email the user mailing list at user@geronimo.apache.org.

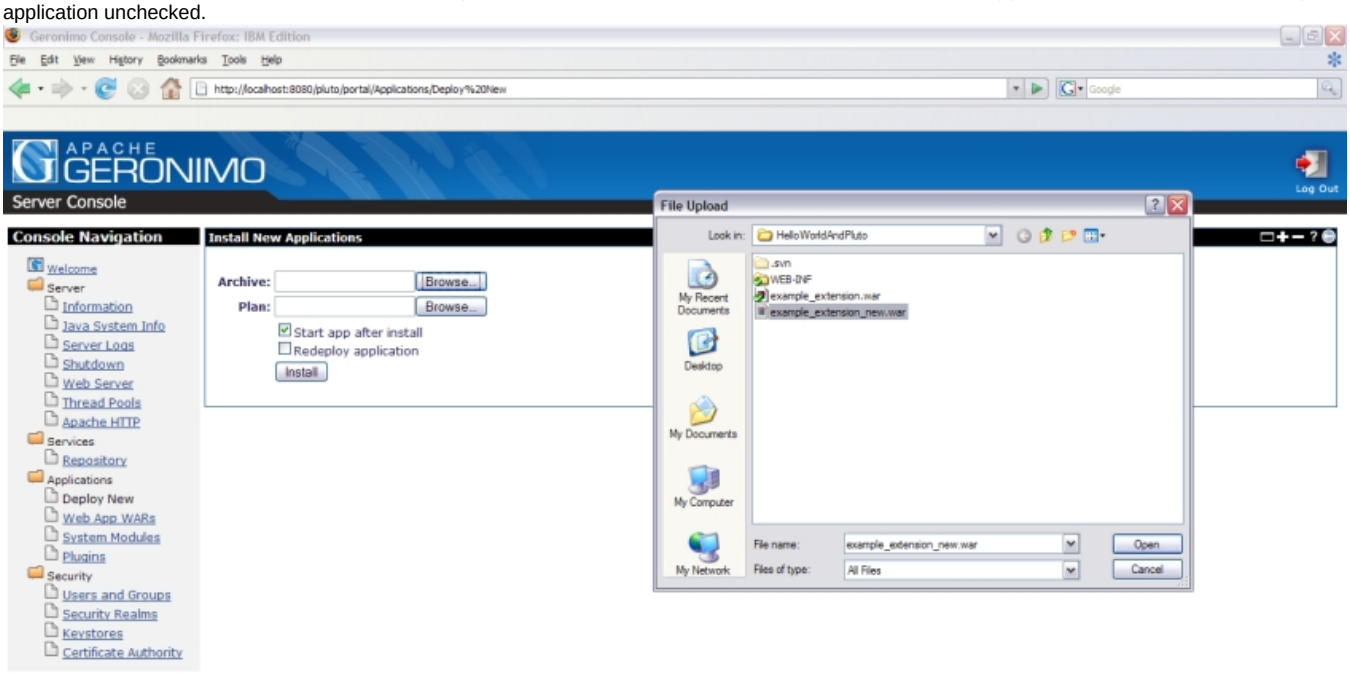
6. Select Install on the next screen if the information is correct, or Return to return to the previous screen.
7. When the plugin installation is complete, refresh your browser. A new menu item will appear on the left. You can now configure this plugin's settings from the newly installed portlets.

Examples

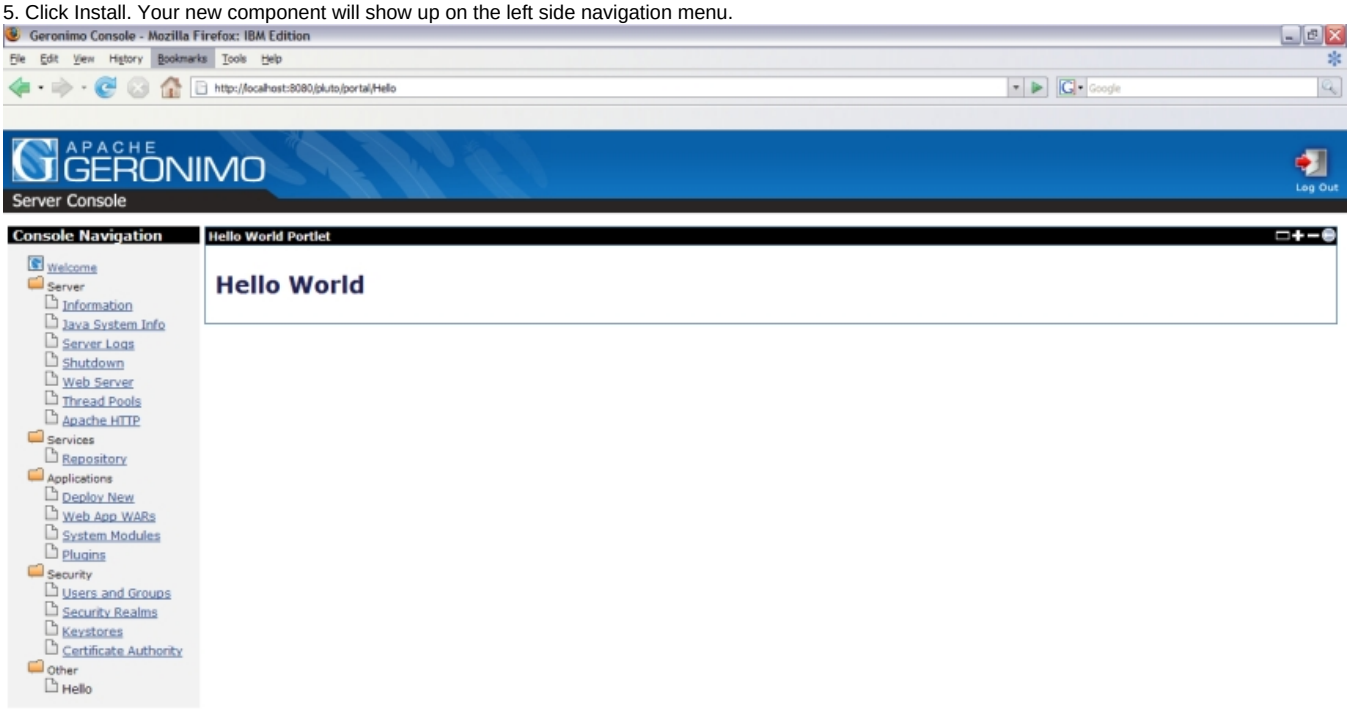
Installing the HelloWorld ACE (.war file)

This is an example of a simple ACE file. It is not attached to any components - it simply adds a new "Hello World" portlet to the Extensible Administration Console.

1. Download example-extension-new.war example_extension_new.war
2. Access the Extensible Administration Console by pointing your web browser to http://localhost:8080/console.
3. Select Deploy New on the left navigation bar.
4. Click browse next to Archive and select the example-extension-new.war file. Leave the Plan box blank, Start app after install checked, and Redeploy application unchecked.



5. Click Install. Your new component will show up on the left side navigation menu.



Customization

Often times when a user installs Geronimo, the Administration Console will be the first place they go in order to configure their application server. For this reason, it is important that the administration console be a feature that provides users with the necessary functionality to easily manage plugins and components of the server. The administration console allows dynamic control over administration features. As such, it reflects the highly flexible and pluggable underlying architecture of Geronimo. What we have done here is to provide a framework for plugin developers to include administration console extensions for their plugins. The aim has been to provide an easy and flexible solution with a simple architecture such that users of Geronimo can intuitively configure components, and developers can easily expand upon the foundation laid out here.

The purpose of the remainder of this document is to provide developers with the necessary instructions to utilize the infrastructure provided by the administration console to add customized console extensions for their Apache Geronimo plugins. The first portion will describe the plumbing of the system and how it works, and the latter portions will provide a solid example of what to expect in order to actually create your own functioning console extension.

Architecture

Below we have a high level view of the interactions between a new plugin and Pluto. This is essentially the 'plumbing' beneath the administration console.

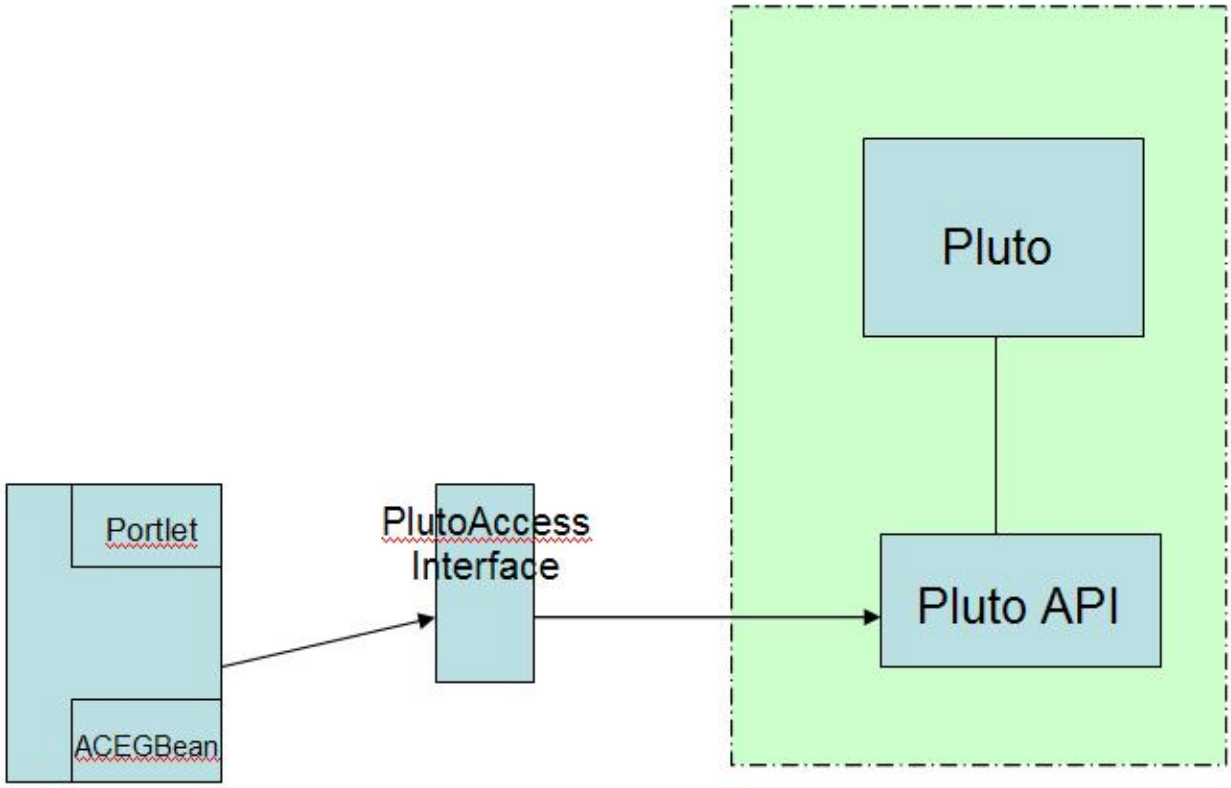


Figure 1 - High level view of Pluto's interaction with a deployable WAR

We have a WAR file containing portlet information and the definition of an Administration Console Extension (ACE) GBean on the left. The ACE GBean is the way in which we can invoke operations on Pluto (this will be covered shortly).

The portlet container that we use is Apache Pluto 1.2. Pluto allows for dynamically adding and removing pages and portlets without restarting the portlet container. The approach is important from a user interface standpoint, because we are not forced into performance bottle necks by restarting server components unnecessarily.

The dashed box around Pluto and its Pluto's API is intended to describe the approach taken to decouple Pluto's implementation from our own. This is a design decision to allow Pluto to be replaced by a different portlet engine for possible future development.

Understanding Through Usecases

Let's dig into the architecture a bit more now. It should be noted that the intended purpose of this architecture is to leverage Geronimo's flexible nature, as well as have the actual administration console be reflective of Geronimo's pluggability. The infrastructure should be simple to understand and easy to build on top of.

The first use case we will look at will also be used to describe the individual pieces that make administration console function.

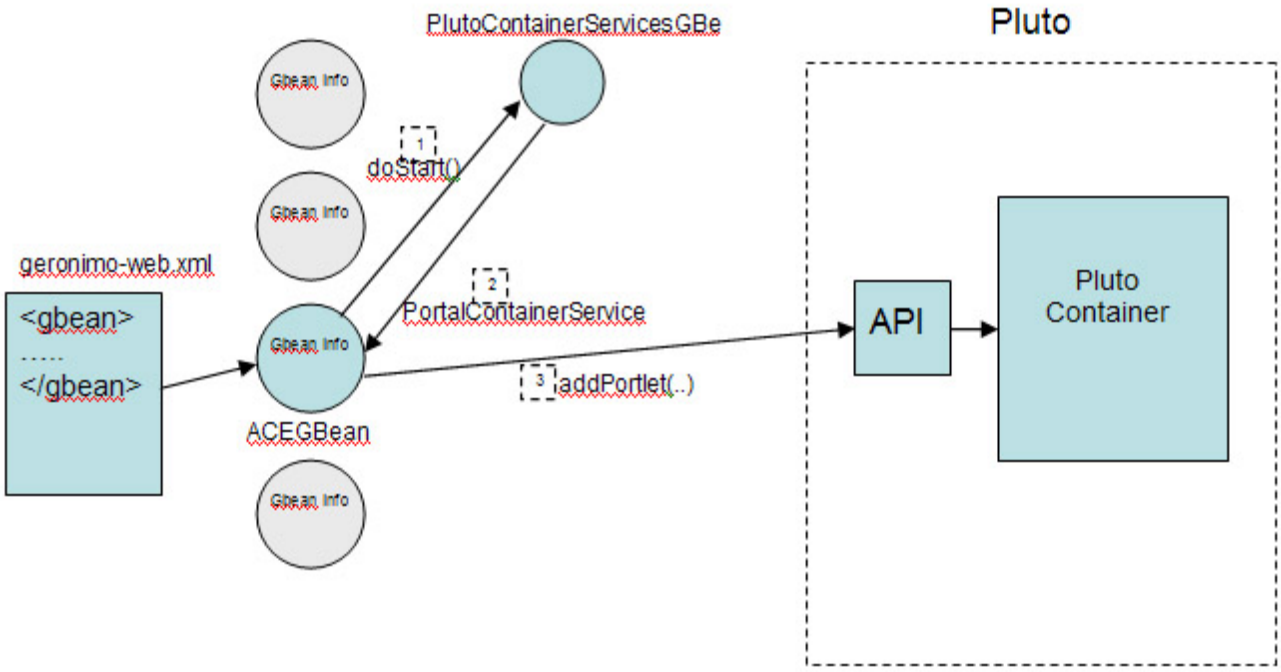


Figure 2 - Use case for adding features to the console

On the left, we begin with a simple Geronimo deployment plan (geronimo-web.xml, for instance). This deployment plan is the same deployment plan included in any archive that you may wish to deploy into Geronimo. In this case, the deployment plan includes the definition for a special AdminConsoleExtensionGBean. The specifics of what goes into an AdminConsoleExtensionGBean will be described in more detail later.

On the right, we have Pluto. When Pluto starts, a PortalContainerServicesGBean is started. This is how we access Pluto. When we deploy our archive, an AdminConsoleExtensionGBean containing portlet configuration information is started (based on the configuration specified in the deployment descriptor). The AdminConsoleExtensionGBean is where the developer specifies the information for the portlets and pages they are adding to the administration console.

The AdminConsoleExtensionGBean implements GBeanLifecycle. When the AdminConsoleExtensionGBean is started, it asks the kernel for the PortalContainerServicesGBean (1). The AdminConsoleExtensionGBean gets the service (2) - now the AdminConsoleExtensionGBean can access Pluto through its API. We want to add a portlet to our administration console, so we invoke Pluto's addPortlet method, which will update Pluto's internal model.

In the above image, each of the grayed out GBeans is an AdminConsoleExtensionGBean instantiated by a deployment descriptor. Each AdminConsoleExtensionGBean modifies or adds exactly one page in the administration console. This essentially means that portlets may be added to an existing page by specifying the name of an existing page, or a separate page can be created if one with the specified name does not exist.

A deployment plan may also include definitions for multiple AdminConsoleExtensionGBeans. This means that a deployment plan, if it chooses to do so, may add as many pages to the console as it wants to.

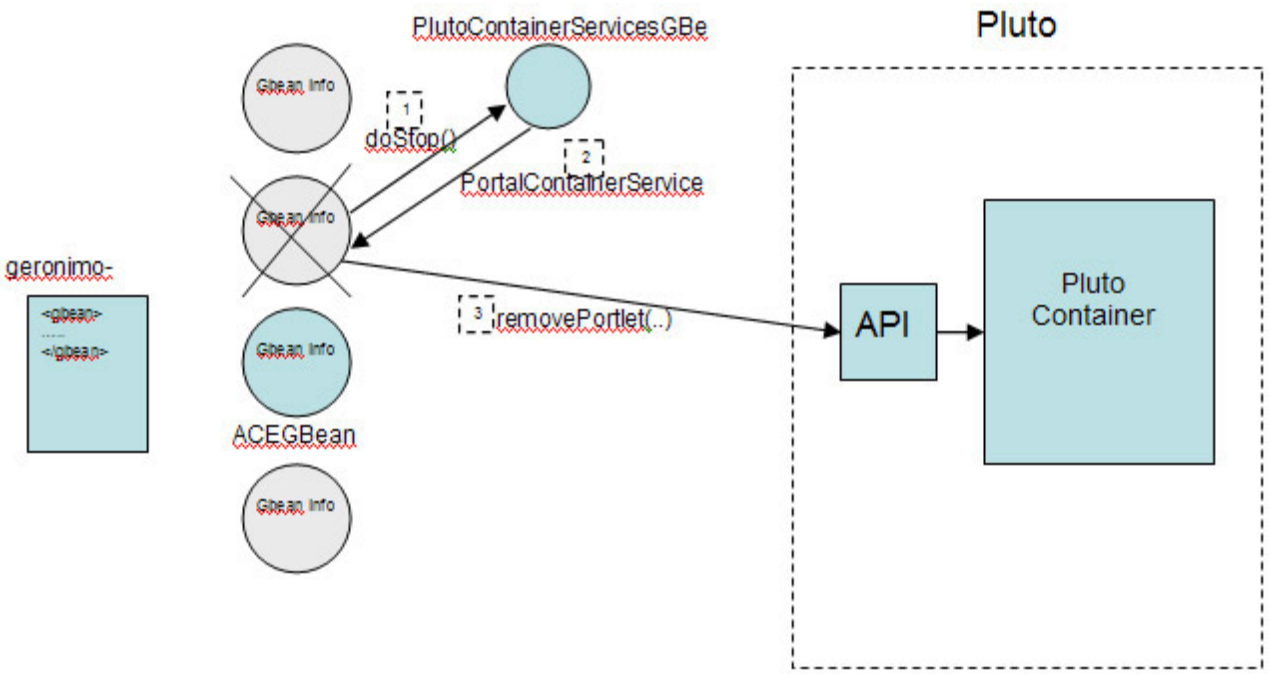


Figure 3 - Use case for removing features from the console

Removing pages (Figure 3) from the console follows a similar process. Because the AdminConsoleExtensionGBean implements the GBeanLifecycle, when the installed component is stopped, its AdminConsoleExtensionGBean will also be stopped. When this happens, we once more ask the kernel for the PortalContainerServiceGBean, which is used to call removePortlet on Pluto through its API. From the view of the administration console, the user will see the page has disappeared from the navigation menu (Figure 4). If the user starts the component again, the GBeanLifecycle's doStart calls addPortlet on Pluto Container, and the administration console extension reappears on the navigation menu (Figure 5).

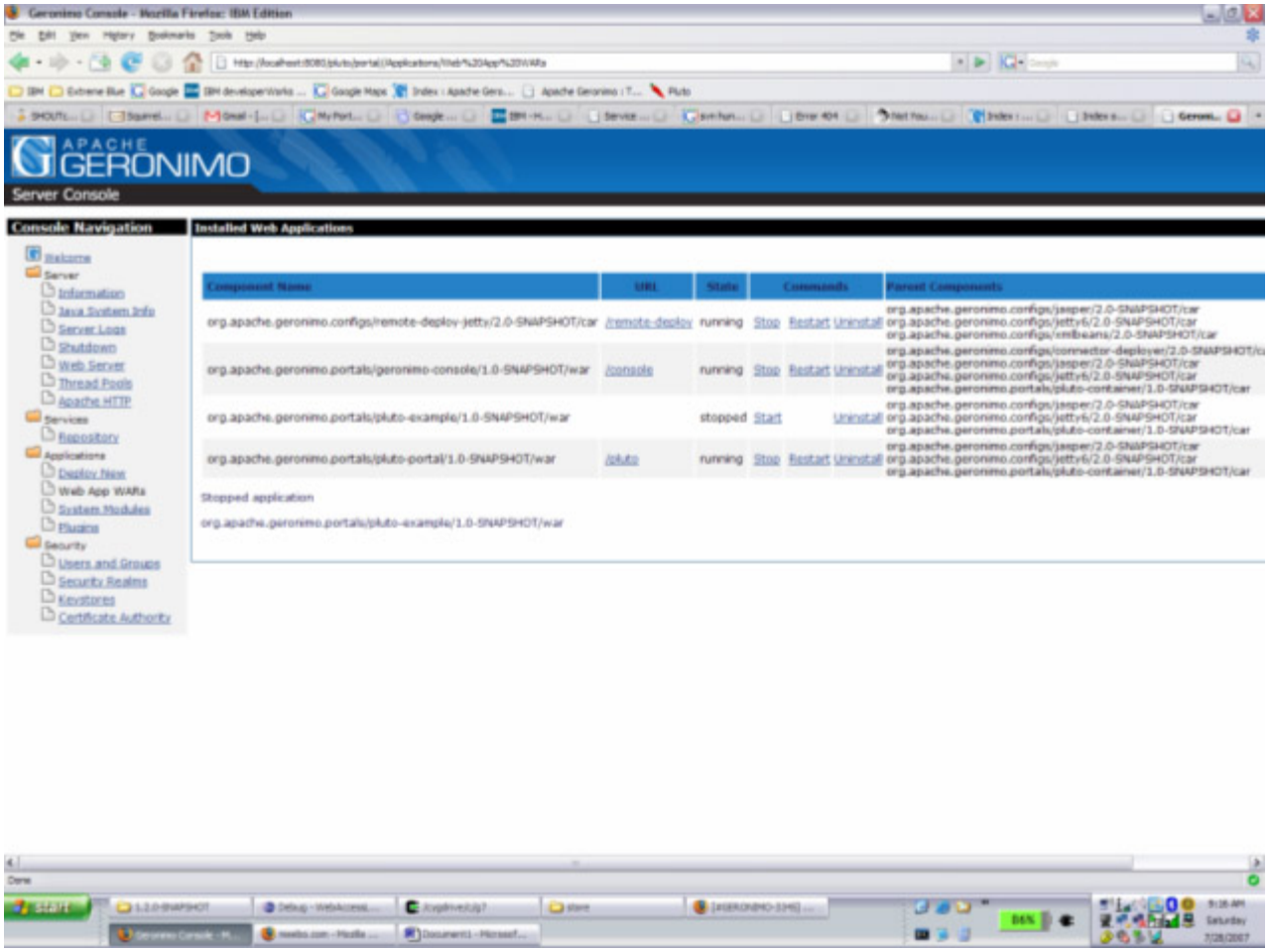


Figure 4 - We see no administration extensions to the left for the stopped component

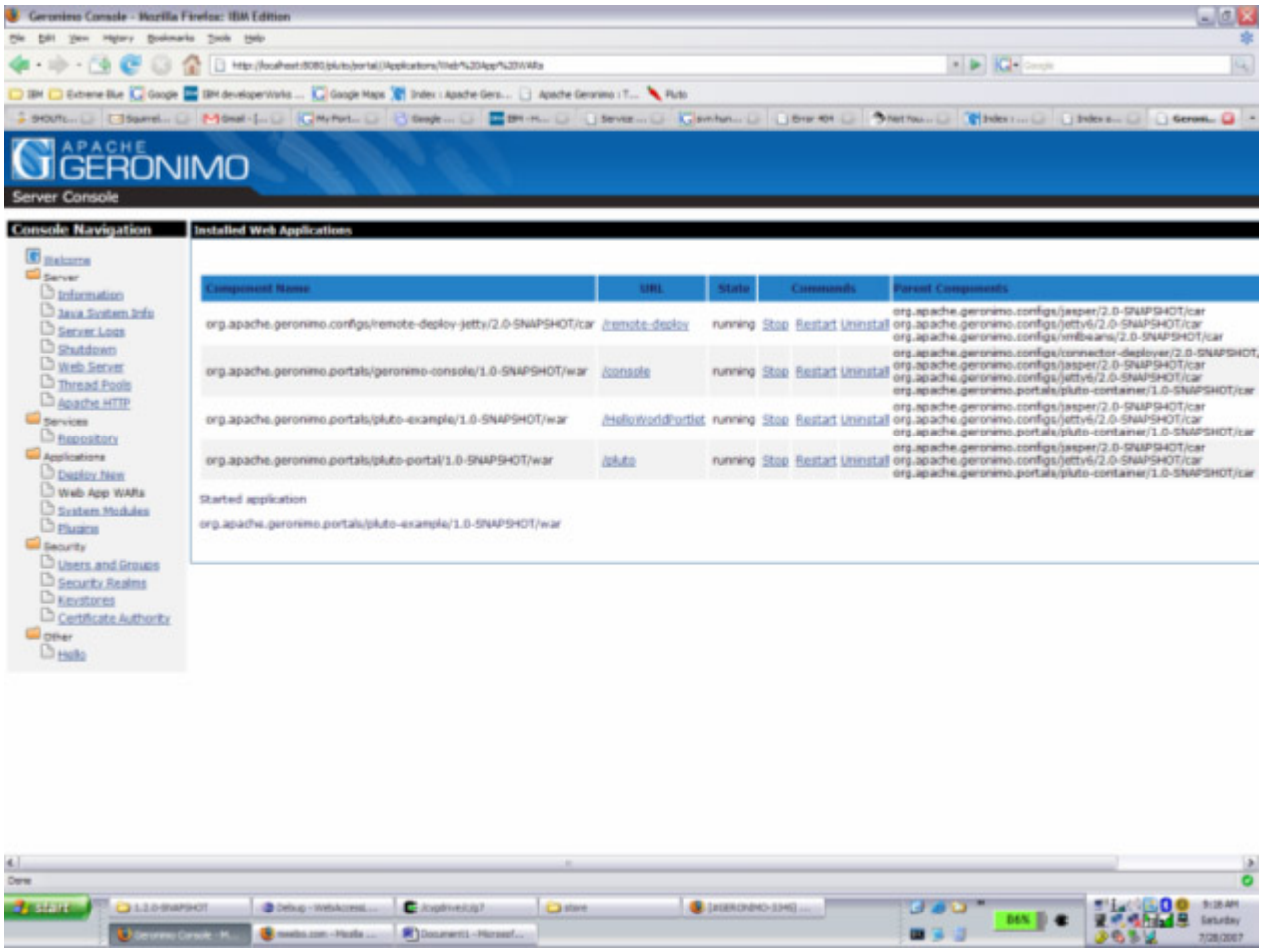


Figure 5 - We see the administration feature for the HelloWorldPortlet appear in the navigation

Class/Dependency Structure

Dependency relationship of the pieces

- PortalContainer
  - PortalContainerServiceGBean
    - AdminConsoleExtensionGBean
  - Pluto Portal
    - Administration Console
      - New Portlet

The PortalContainer is the base requirement for the administration console. The Pluto Portal and the PortalContainerServiceGBean are the base requirements to install anything into the framework of the console. The administration console and the AdminConsoleExtensionGBean depends on the PortalContainerServiceGBean and the Pluto Portal. In order to add new console extensions, all of the described components must be in place.

The key here is that the requirements for adding a console extention is an AdminConsoleExtensionGBean and the New Portlet files. This means that the required dependencies that need to be specified in the deployment descriptor are the Administration Console and the PortalContainerServicesGBean.

How to Develop an Administrator Console Extension (ACE)

In order to develop your own console extension, you will need to perform the following steps:

1. Create a portlet
2. Add an AdminConsoleExtensionGBean to the deployment plan
3. Build the WAR or CAR
4. Deploy the application
5. Verify the installation

Each step is further detailed below.

Different ways of structuring

The whole concept behind creating an ACE for Geronimo is that you are adding an extension in the form of portlets to the administrator console. You can package this however you like. Some options are as follows:

1. Your portlets
2. Your portlets + your related component
3. Your portlets which declares a dependency on your related component

Create a portlet

In their most fundamental state, an ACE defines where to place new/old portlets in your console. Portlets are the fundamental building block of the administrator console. We are using Pluto 2.x as our portal engine since it closely conforms to JSR 168. We will not go into the details of how to develop portlets since you can find many tutorials and tools for this such as Portlet Plugin for Eclipse. You will need the appropriate "web.xml" and "portlet.xml" files defined. As a simple example, refer to the HelloWorldPortlet defined at the end of this section.

Add an AdminConsoleExtensionGBean

To add an AdminConsoleExtensionGBean to your deployable archive file, you need to include the following to your "geronimo-web.xml" deployment plan:

1. Declare a dependency on the "pluto-portal"

```
<dependencies>
    ...
<dependency>
    <groupId>org.apache.geronimo.plugins</groupId>
    <artifactId>pluto-support</artifactId>
</dependency>
</dependencies>
```

2. Define an AdminConsoleExtensionGBean

```
<gbean name="example" class="org.apache.geronimo.pluto.AdminConsoleExtensionGBean">
    <attribute name="pageTitle">Testing</attribute>
    <attribute name="portletContext">/HelloWorldPortlet</attribute>
    <attribute name="portletList">[HelloWorldPortlet]</attribute>
</gbean>
```

   where the attributes are defined as follows:

- pageTitle: The name of the page to add these portlets to (new or existing)
- portletContext: The context of where the portlets are installed.
- portletList: A comma-delimited list of the portlets to be added to this page.

Sample geronimo-web.xml

```
<web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.2">
    <environment>
        <moduleId>
            <groupId>org.apache.geronimo.portals</groupId>
            <artifactId>pluto-example</artifactId>
            <version>1.0-SNAPSHOT</version>
            <type>war</type>
        </moduleId>

        <dependencies>
            <dependency> <!-- Put a dependancy on the hosting portal (pluto) -->
                <groupId>org.apache.geronimo.plugins</groupId>
                <artifactId>pluto-support</artifactId>
            </dependency>
        </dependencies>
    </environment>

    <!-- This is where the files are accessed from. (aka - portletContext) -->
    <context-root>/HelloWorldPortlet</context-root>

    <!-- Start off a ACEGBean, this is the lifecycle for the portlet -->
    <gbean name="PlutoTest" class="org.apache.geronimo.pluto.AdminConsoleExtensionGBean">
        <attribute name="pageTitle">Hello</attribute>
        <attribute name="portletContext">/HelloWorldPortlet</attribute>
        <attribute name="portletList">[HelloWorldPortlet]</attribute>
        <reference name="PortalContainerServices">
            <name>PlutoPortalServices</name>
        </reference>
    </gbean>
</web-app>
```

Putting it together

Add your modified "geronimo-web.xml" deployment plan to your WAR or CAR in the WEB-INF folder. Also, if you are including portlets, your archive file should contain the appropriate class files as well as the "web.xml" and "portlet.xml" files.

What is an example structure for a simple war?

- HelloWorldPortlet.war
  - WEB-INF/
    - classes/
      - (portlet class files)
    - geronimo-web.xml
    - portlet.xml
    - web.xml

Deploying the application

Deploy the archive as you normally would either using the "deploy" command or by using the Administrator Console's "Deploy New" or "Plugins".

1. *Command Line*: **`<geronimo bin folder>/deploy.bat deploy <path to war>`**
   Example:
   `cd c:/g/target/geronimo-tomcat6-minimal-2.0-SNAPSHOT/bin/`
   `deploy.bat deploy c:/HelloWorldPortlet.war`
2. *Using the Admin Console*:

Verifying installation

Go to http://localhost:8080/console/ to verify that the portlets were added to the correct page. (You may need to refresh the browser if you deployed from the command line.)

Sample Extension

This is a working simple example of an Administration Console Extension.
Download the example WAR

HelloWorldPortlet.java

```
package org.apache.pluto.examples;

import java.io.IOException;
import java.io.PrintWriter;

import javax.portlet.GenericPortlet;
import javax.portlet.PortletException;
import javax.portlet.RenderRequest;
import javax.portlet.RenderResponse;

/**
 * A very simple portlet example.
 * This portlet displays 'Hello World' to the end user
 */
public class HelloWorldPortlet extends GenericPortlet {

    // This function is called when a user requests to view this portlet (by
    // navigating to the webpage in Pluto)
    public void doView(RenderRequest request, RenderResponse response)
            throws PortletException, IOException {

        // Set the response to read HTML
        response.setContentType("text/html;charset=UTF-8");

        // Required call for use of getWriter() and getPortletOutputStream()
        PrintWriter out = response.getWriter();
        // Write content to the portlet
        out.println("<h1>Hello World</h1>");
    }
}
```