Using Cernunnos in Servlets & Portlets

Drew Wills

JA-SIG Summer Conference, Dallas TX

March 2nd, 2009

UNICON

1. The Cernunnos Project
2. Creating Servlets & Portlets
3. Servlets & Portlets in uPortal

The Cernunnos Project

Project History & Cernunnos Basics

Cernunnos at a Glance

Project Home Page:

http://cernunnos.googlecode.com/

Discussion Group:

http://groups.google.com/group/cernunnos-discussion/

Manual:

http://cernunnos.googlecode.com/svn/manual/index.html

Project Status:

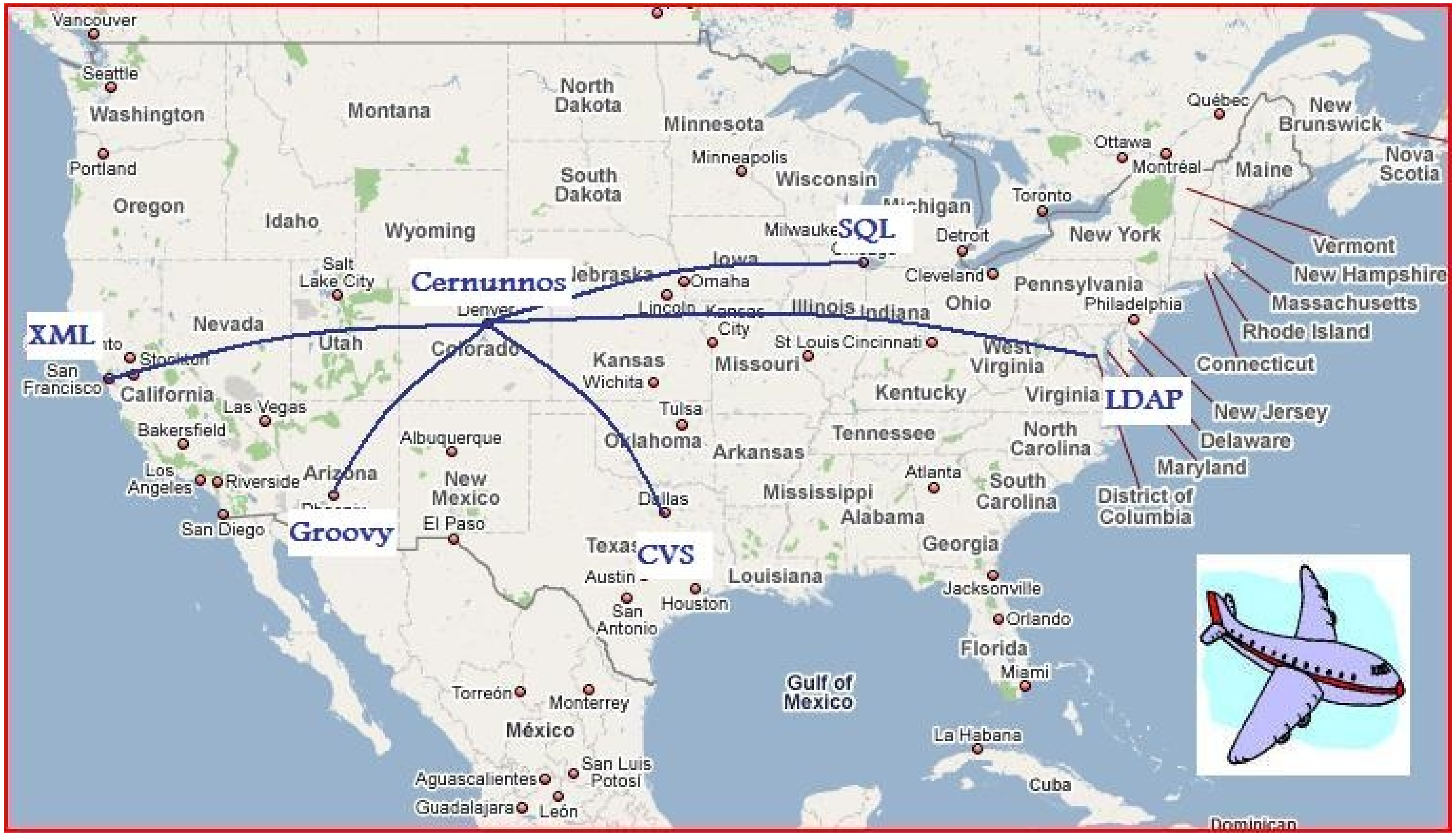
- Version 1.0.0 released September 14th, 2008
- 8 Project Members
- > 20k lines of source (code, comments, blanks)
- > 450 commits since February 2007

What is Cernunnos?

- Cernunnos helps you be more productive
- *Here's how it works...*
  - You don't have to tell components, subsystems, or objects *how* to work together
  - You just have to tell them to do so
  - This simple difference reduces busywork and bulk dramatically
  - It's like a hub airport for code

Welcome to *Java Airways*

Try *Cernunnos Airways* Instead



Jigsaw Puzzles vs. LEGO Bricks

- Consider another example: jigsaw puzzles
  - Puzzle pieces only combine in one way
  - If you want to reuse puzzle pieces, **you have to create new pieces** that will accept their unique shapes
- Each LEGO brick, however, already combines with every other LEGO brick -- past, present, and future
- Cernunnos is like *LEGO-typing for the Java Platform*

Tasks & Phrases

- There are 2 types of components in Cernunnos
- A Task is:
  - A unit of work
  - Like a verb; it describes *what operation will be performed*
  - Represented by an XML element (e.g. `<xslt>`)
- A Phrase is:
  - An expression that evaluates to a value
  - Like a noun; it describes *who performs* an operation and *to, for, or upon whom it will be performed*
  - Usually represented by an XML attribute

```
value="${grovy(new TreeMap())}"
```

Request Attributes

- Cernunnos components do not maintain operational state; they are reusable & thread-safe
- Tasks and Phrases use Request Attributes to manage state and to collaborate
- Request Attributes have scope: they're only visible to descendants, not ancestors or siblings
- Many Tasks can create Request Attributes (e.g. `<with>`, `<with-attribute>`, `<sql-datasource>`)
- The most common way to access an attribute is like this: `${attributeName}`

Cernunnos & JA-SIG Time Line

- (2007/02/14) Cernunnos Project created on GoogleCode
- (2007/04/23) Andrew Petro presents Import/Export at Johns Hopkins University dev meeting
- (2007/07/03) Implemented 'deployPortletApp' Ant target with Cernunnos for uPortal 2.6.0
- (Q4 2007) Yale University sponsored extension and integration of uPortal Import/Export
- (2008/01/10) Import/Export added to uPortal versions 2.5.4, 2.6.2, and 3.0.0

Cernunnos & JA-SIG Time Line (cont.)

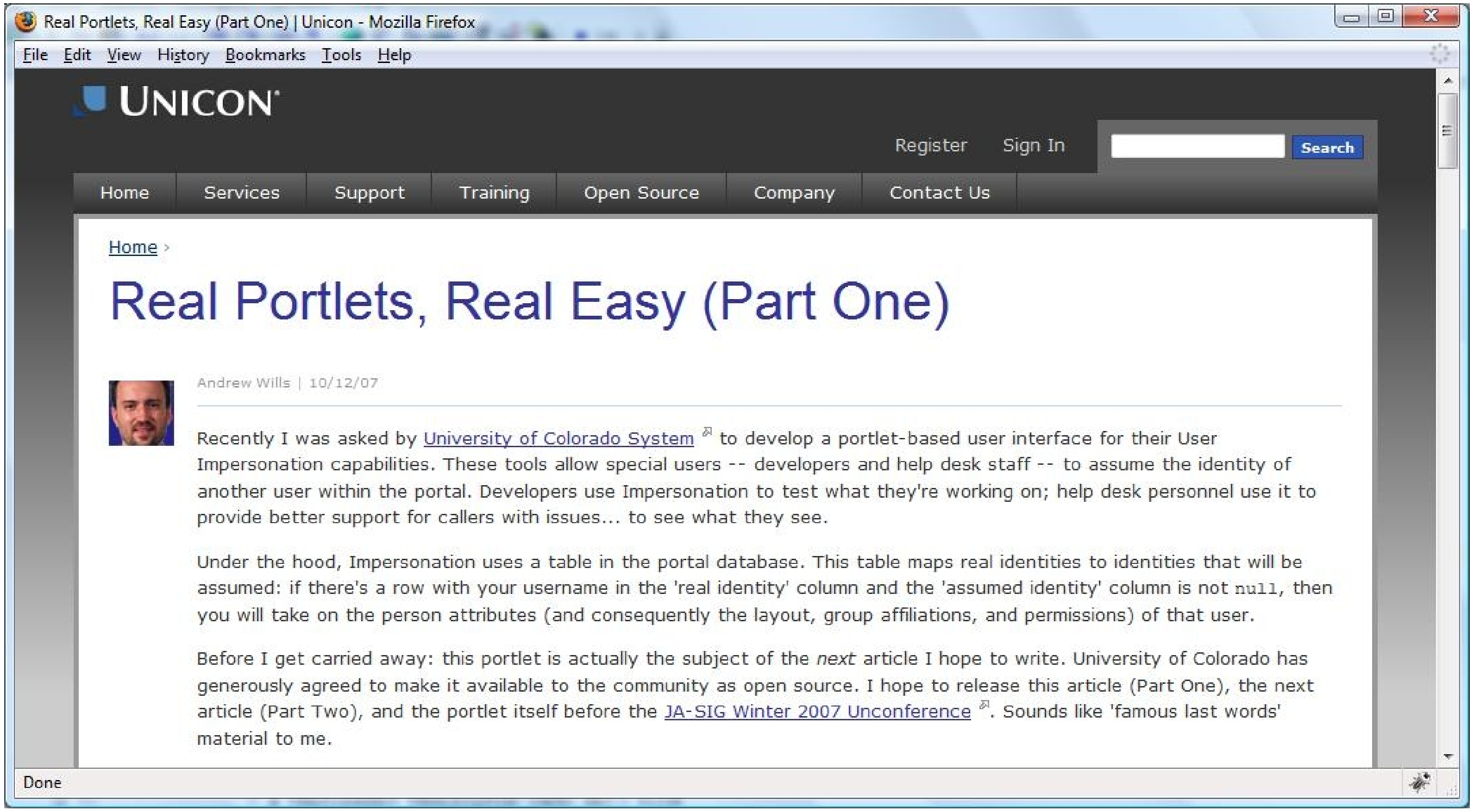
- (Q3 2008) Anthony Colebourne from University of Manchester developes XBEL export for CBookmarks
- (2008/08/29) Johns Hopkins University contributes the SmartLdapGroupStore to uPortal 3.1.0
- (2008/09/14) Cernunnos 1.0.0 released
- (2009/02/20) University of Illinois contributes Import/Export Portlet to uPortal 3.1.0
- (Q2? 2009) Cernunnos 1.1.0 released

Creating Servlets & Portlets

How-To

On Line Article

- There's a helpful article on Cernunnos Portlets at http://www.unicon.net/node/822


Real Portlets, Real Easy (Part One) | Unicon - Mozilla Firefox
File Edit View History Bookmarks Tools Help
UNICON
Register Sign In Search
Home Services Support Training Open Source Company Contact Us
Home ›
Done

Real Portlets, Real Easy (Part One)

Andrew Wills | 10/12/07

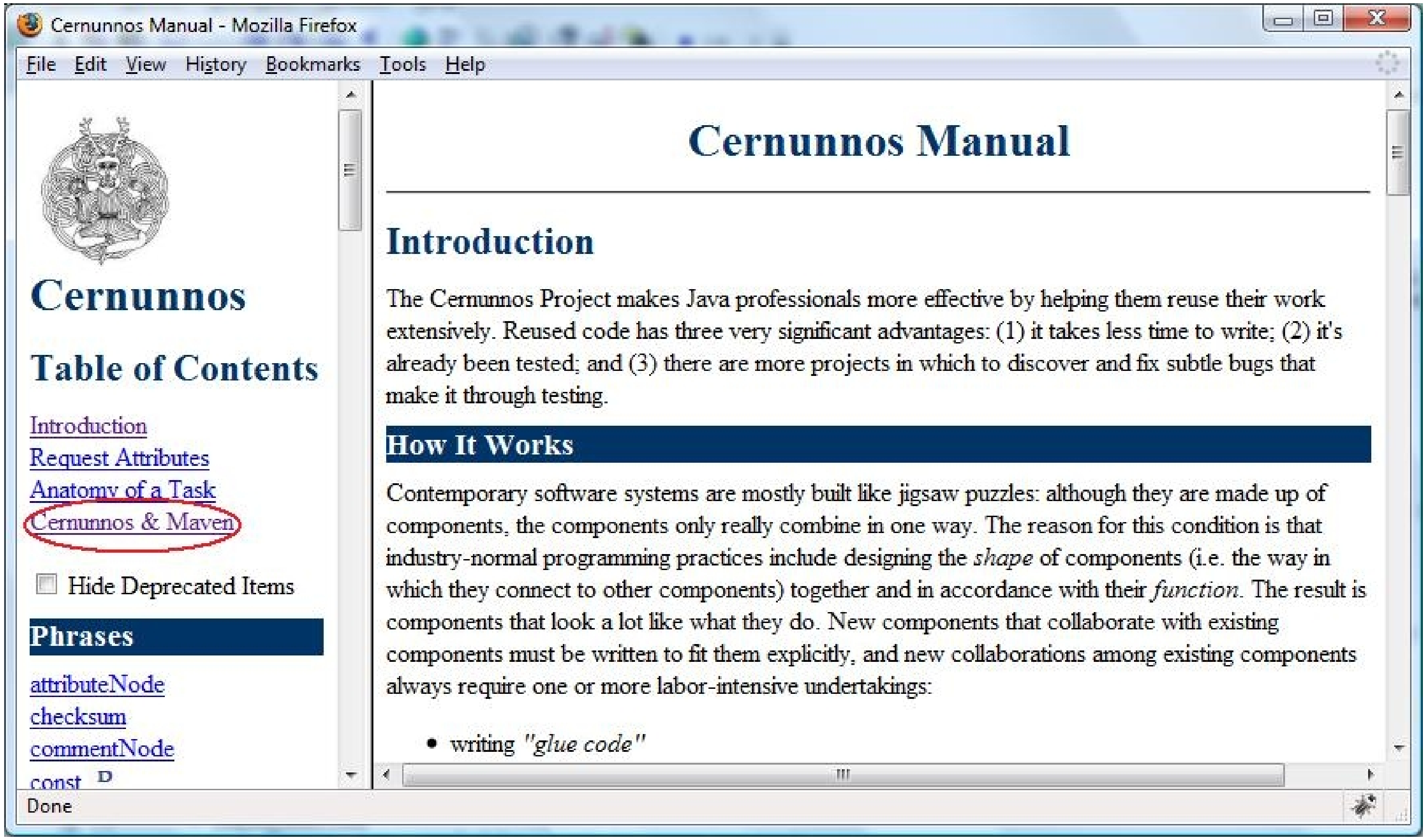
Recently I was asked by University of Colorado System to develop a portlet-based user interface for their User Impersonation capabilities. These tools allow special users -- developers and help desk staff -- to assume the identity of another user within the portal. Developers use Impersonation to test what they're working on; help desk personnel use it to provide better support for callers with issues... to see what they see.

Under the hood, Impersonation uses a table in the portal database. This table maps real identities to identities that will be assumed: if there's a row with your username in the 'real identity' column and the 'assumed identity' column is not `null`, then you will take on the person attributes (and consequently the layout, group affiliations, and permissions) of that user.

Before I get carried away: this portlet is actually the subject of the *next* article I hope to write. University of Colorado has generously agreed to make it available to the community as open source. I hope to release this article (Part One), the next article (Part Two), and the portlet itself before the JA-SIG Winter 2007 Unconference. Sounds like 'famous last words' material to me.

Cernunnos Manual

- Defining Servlets & Portlets is discussed on the Cernunnos & Maven page in the manual



Cernunnos Manual

Introduction

The Cernunnos Project makes Java professionals more effective by helping them reuse their work extensively. Reused code has three very significant advantages: (1) it takes less time to write; (2) it's already been tested; and (3) there are more projects in which to discover and fix subtle bugs that make it through testing.

How It Works

Contemporary software systems are mostly built like jigsaw puzzles: although they are made up of components, the components only really combine in one way. The reason for this condition is that industry-normal programming practices include designing the *shape* of components (i.e. the way in which they connect to other components) together and in accordance with their *function*. The result is components that look a lot like what they do. New components that collaborate with existing components must be written to fit them explicitly, and new collaborations among existing components always require one or more labor-intensive undertakings:

- writing *"glue code"*

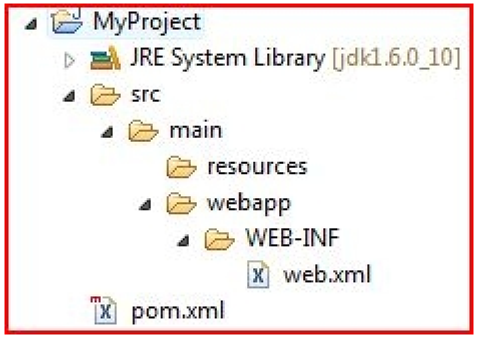
Getting Started

- Things you *probably* want in your project:
  - Cernunnos jar and dependencies
  - JSTL jar
  - Jakarta Standard Taglib jar
  - A build file (Maven, Ant, *etc*.)
  - A Deployment Descriptor (web.xml) file

Getting Started (cont.)

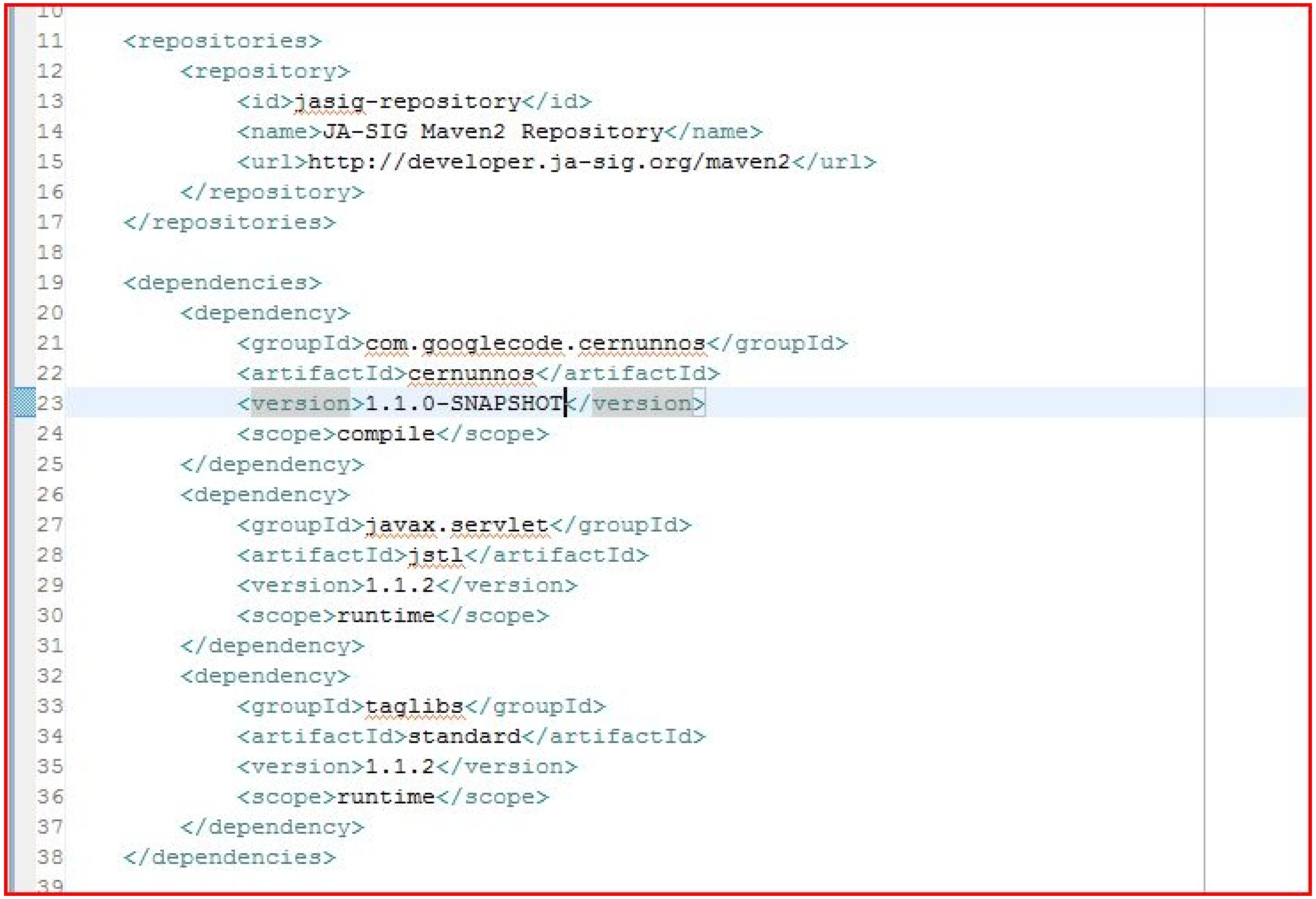
- Get all of these with this Maven command:

```
mvn archetype:create
   -DarchetypeGroupId=com.googlecode.cernunnos
   -DarchetypeArtifactId=cernunnos-webapp
   -DarchetypeVersion=1.1.0-SNAPSHOT
   -DgroupId=<your.groupId>
   -DartifactId=<your.artifactId>
```



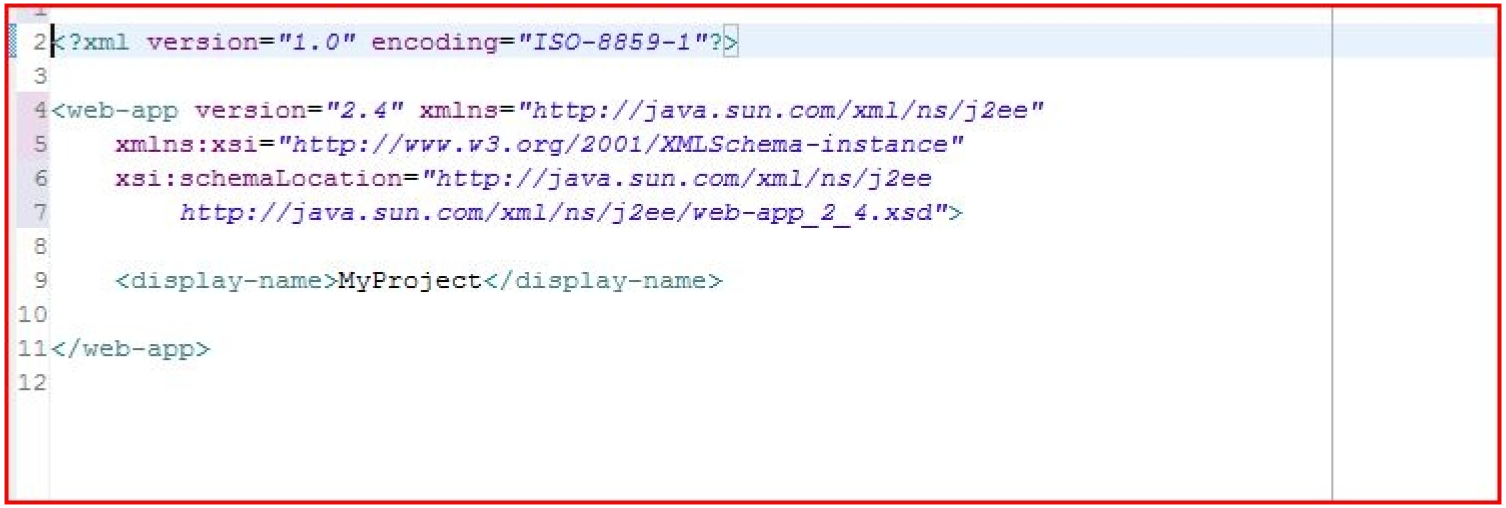
```
MyProject
  JRE System Library [jdk1.6.0_10]
  src
    main
      resources
      webapp
        WEB-INF
          web.xml
  pom.xml
```

pom.xml File



```
    <repositories>
        <repository>
            <id>jasig-repository</id>
            <name>JA-SIG Maven2 Repository</name>
            <url>http://developer.ja-sig.org/maven2</url>
        </repository>
    </repositories>

    <dependencies>
        <dependency>
            <groupId>com.googlecode.cernunnos</groupId>
            <artifactId>cernunnos</artifactId>
            <version>1.1.0-SNAPSHOT</version>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.1.2</version>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>taglibs</groupId>
            <artifactId>standard</artifactId>
            <version>1.1.2</version>
            <scope>runtime</scope>
        </dependency>
    </dependencies>
```

web.xml File



```
<?xml version="1.0" encoding="ISO-8859-1"?>

<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <display-name>MyProject</display-name>

</web-app>
```

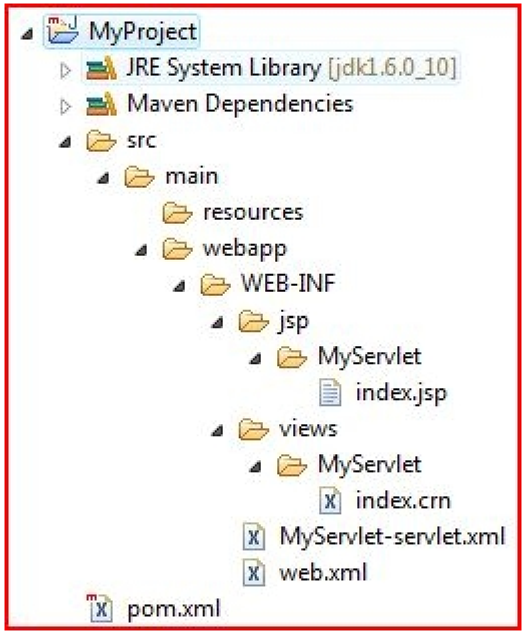
Creating a New Servlet

- For a minimal Servlet example you may:
  - Define `<servlet>` and `<servlet-mapping>` in web.xml
  - Provide a JSP file for HTML markup
  - Provide a CRN file with a `<request-dispatcher>` that invokes your JSP
- You can get all of these with this Cernunnos command (requires 1.1.0):

```
>crn define-servlet.crn <servlet.name>
```

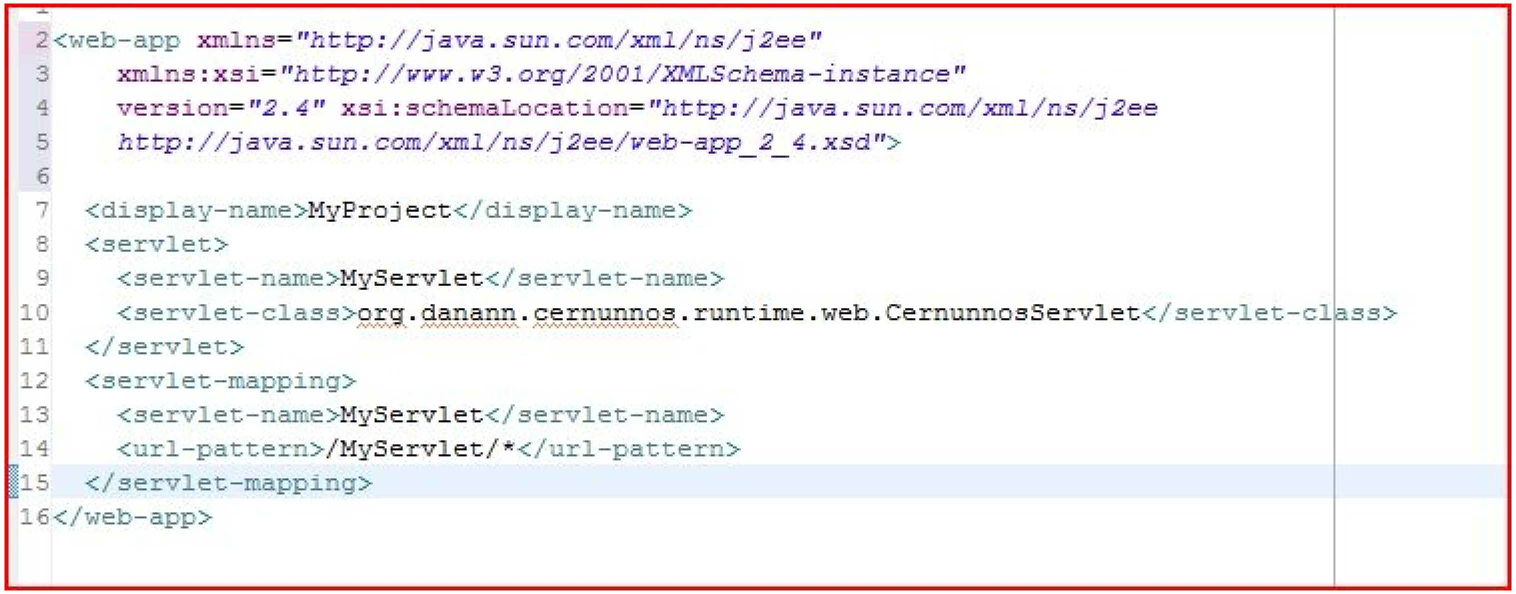
- You will also get a *-servlet.xml context file

Project Directory Structure



```
MyProject
    JRE System Library [jdk1.6.0_10]
    Maven Dependencies
    src
        main
            resources
            webapp
                WEB-INF
                    jsp
                        MyServlet
                            index.jsp
                    views
                        MyServlet
                            index.crn
                    MyServlet-servlet.xml
                    web.xml
    pom.xml
```

web.xml File



```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    version="2.4" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

  <display-name>MyProject</display-name>
  <servlet>
    <servlet-name>MyServlet</servlet-name>
    <servlet-class>org.danann.cernunnos.runtime.web.CernunnosServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyServlet</servlet-name>
    <url-pattern>/MyServlet/*</url-pattern>
  </servlet-mapping>
</web-app>
```

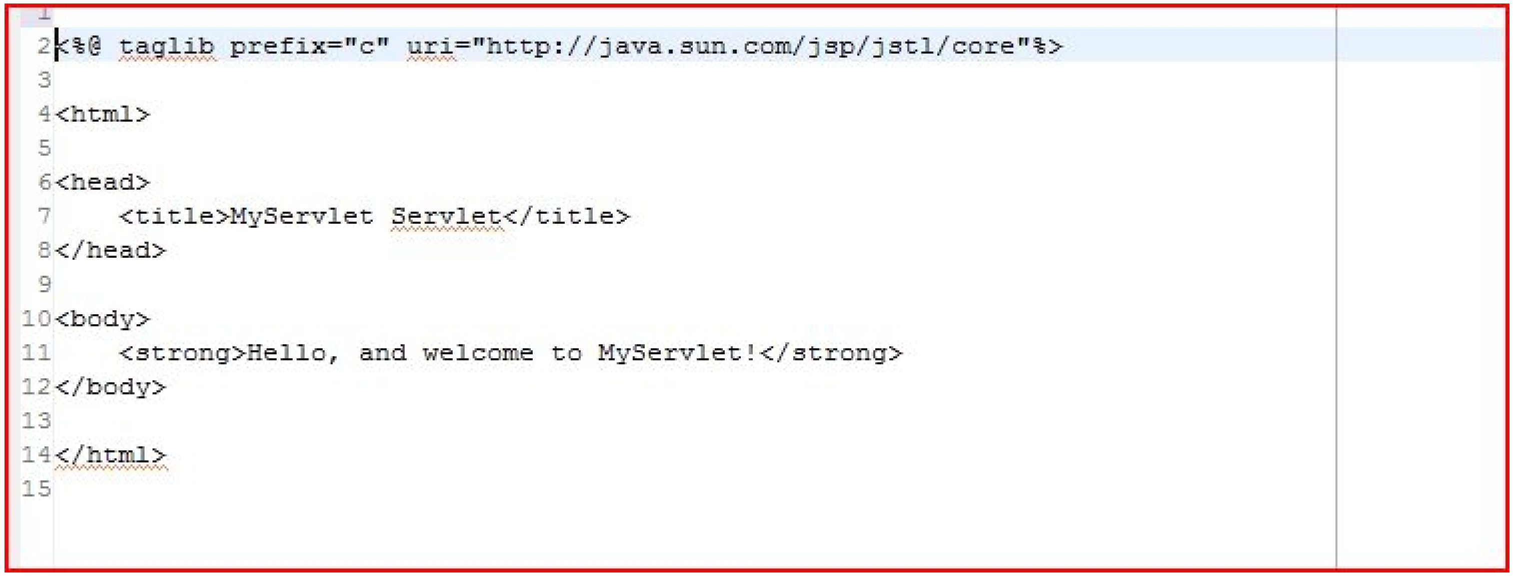
Notes on Defining a Servlet

- Specify `CernunnosServlet` for `<servlet-class>`
- You can specify `contextConfigLocation` as an init parameter
- If you don't specify a context file, the servlet will:
  - Look for a context in the default location (WEB-INF/*-servlet.xml)
  - Use all default settings

Notes on Defining a Servlet (cont.)

- You can specify `scriptLocation` or `getScriptLocation/postScriptLocation` as init parameters
- If you don't specify scripts as init parameters the servlet will invoke:
  - One (optional) action (at `/WEB-INF/actions/${action}.crn` by default)
  - Followed by one view (at `/WEB-INF/views/${view}.crn` by default)
- The "default default" view name is 'index'; you can specify a different default in the context file

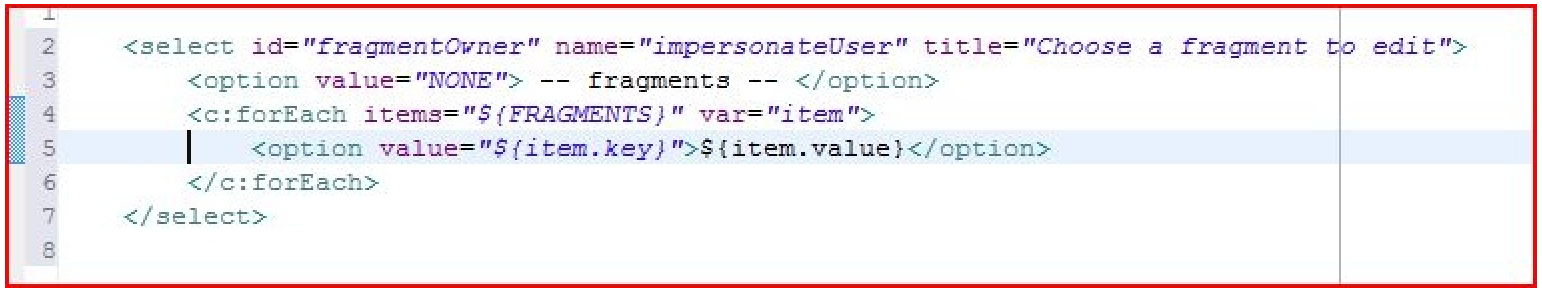
index.jsp File



```

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<html>

<head>
    <title>MyServlet Servlet</title>
</head>

<body>
    <strong>Hello, and welcome to MyServlet!</strong>
</body>

</html>

```

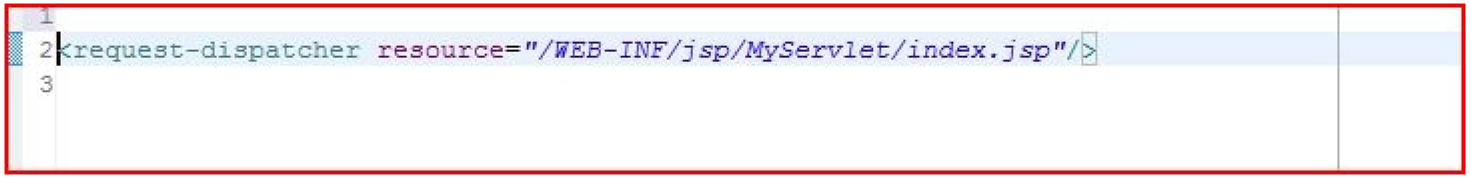
Notes on JSP Files

- The JSTL Core <taglib> is pre-defined by define-servlet.crn
- Request attributes that are in scope when `<request-dispatcher>` is invoked will be available in EL expressions



```
<select id="fragmentOwner" name="impersonateUser" title="Choose a fragment to edit">
    <option value="NONE"> -- fragments -- </option>
    <c:forEach items="${FRAGMENTS}" var="item">
        <option value="${item.key}">${item.value}</option>
    </c:forEach>
</select>
```

index.crn File



```
<request-dispatcher resource="/WEB-INF/jsp/MyServlet/index.jsp"/>
```

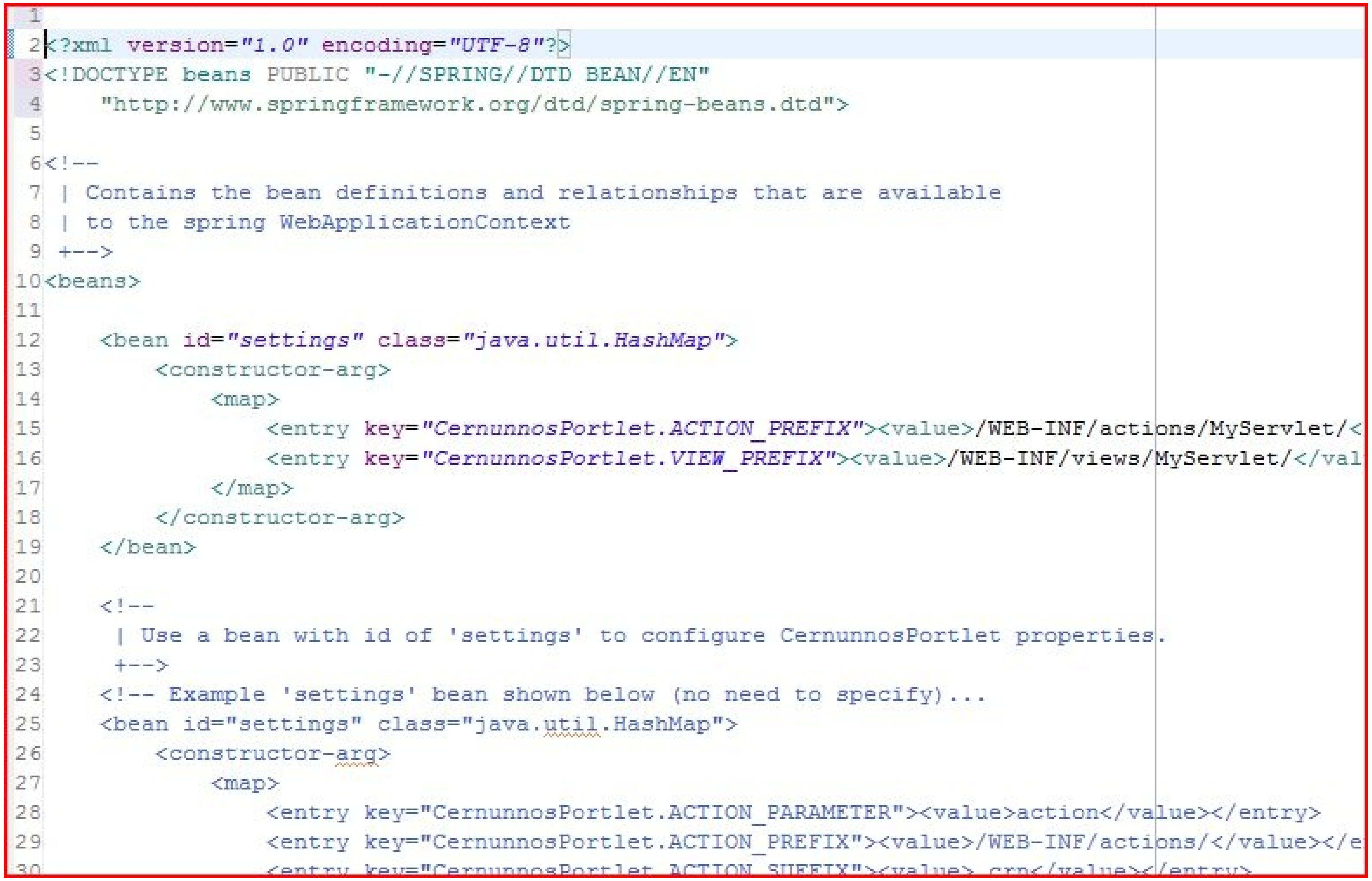
Notes on CRN Files

- In MVC-mode, the Servlet uses two types of Cernunnos XML: *actions* (optional) and *views*
- Leverage all the features of Cernunnos (*e.g.* Groovy expressions, XML, Spring, *etc*.)
- Seemlessly interact with Java code, RDBMS, LDAP, Web Services, CVS, Facebook, *etc*.

Notes on CRN Files (cont.)

- CernunnosServlet provides some important request attributes:
  - WebAttributes.REQUEST: `HttpServletRequest`
  - WebAttributes.RESPONSE: `HttpServletResponse`
- Use one of these methods to write Servlet output:
  - `<request-dispatcher>`: Renders the specified JSP
  - `<download>`: Sends the specified String, byte[], or InputStream; you can specify content-type and even suggest a file name
  - `<xslt>`: Transform to HTML, then write to the HttpServletResponse

MyServlet-servlet.xml



```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
    "http://www.springframework.org/dtd/spring-beans.dtd">

<!--
 | Contains the bean definitions and relationships that are available
 | to the spring WebApplicationContext
 +-->
<beans>

    <bean id="settings" class="java.util.HashMap">
        <constructor-arg>
            <map>
                <entry key="CernunnosPortlet.ACTION_PREFIX"><value>/WEB-INF/actions/MyServlet/<
                <entry key="CernunnosPortlet.VIEW_PREFIX"><value>/WEB-INF/views/MyServlet/</val
            </map>
        </constructor-arg>
    </bean>

    <!--
     | Use a bean with id of 'settings' to configure CernunnosPortlet properties.
     +-->
    <!-- Example 'settings' bean shown below (no need to specify)...
    <bean id="settings" class="java.util.HashMap">
        <constructor-arg>
            <map>
                <entry key="CernunnosPortlet.ACTION_PARAMETER"><value>action</value></entry>
                <entry key="CernunnosPortlet.ACTION_PREFIX"><value>/WEB-INF/actions/</value></e
                <entry key="CernunnosPortlet.ACTION_SUFFIX"><value>.crn</value></entry>
```

Notes on Servlet Context Files

- These define standard Spring application contexts
- You may use the 'settings' bean to define:

ACTION_PARAMETER: Request param signaling an action (default 'action')

ACTION_PREFIX: Where action files reside (default '/WEB-INF/actions/')

ACTION_SUFFIX: File extension for actions (default '.crn')

VIEW_PARAMETER: Request param signaling a view (default 'view')

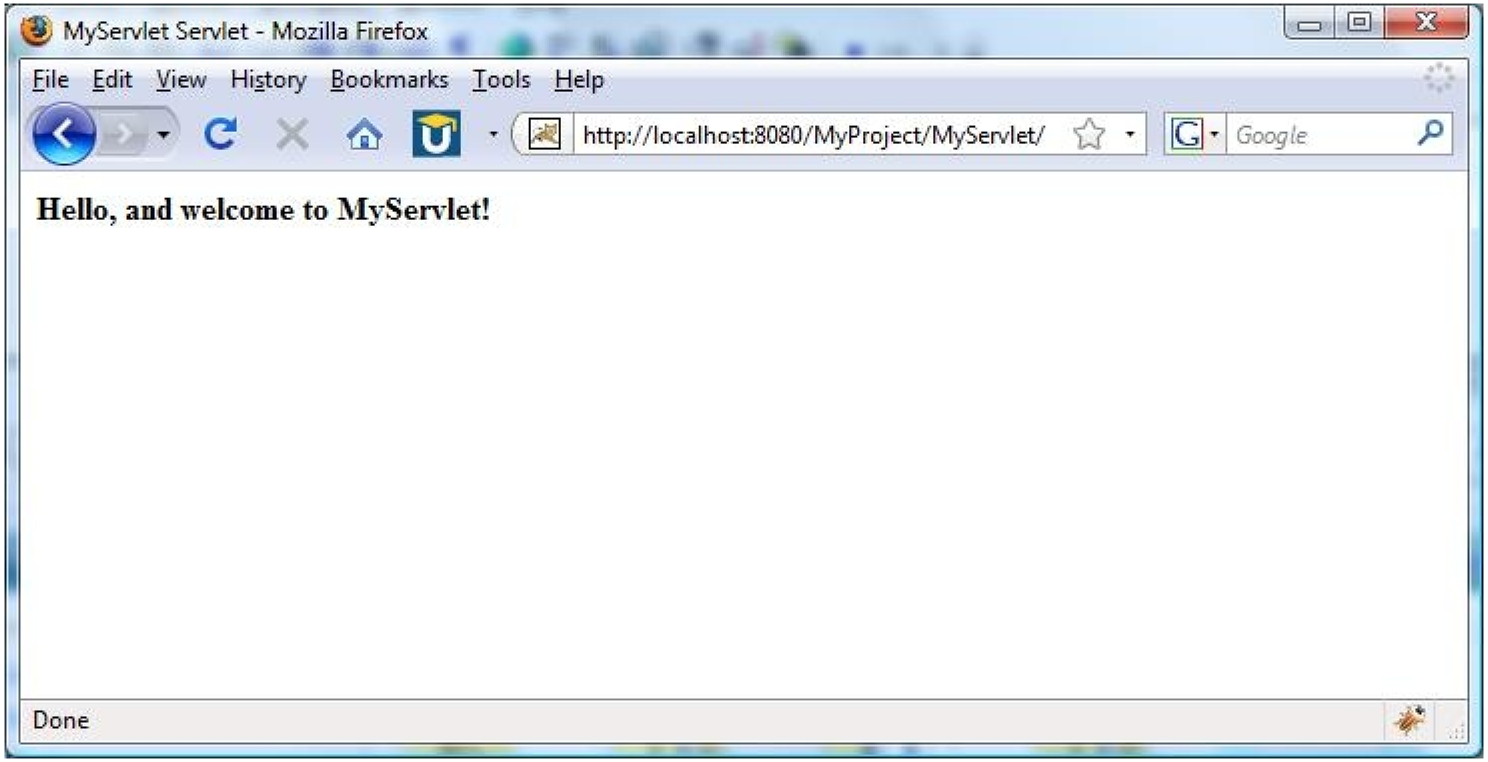
VIEW_PREFIX: Where view files reside (default '/WEB-INF/views/')

VIEW_SUFFIX: File extension for views (default '.crn')

DEFAULT_VIEW: View to display if none is specified (default 'index')

- You may use the 'requestAttributes' bean to define request attributes that apply to every action & view (e.g. 'dataSource')

MyServlet Screen Shot


MyServlet Servlet - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/MyProject/MyServlet/
Google

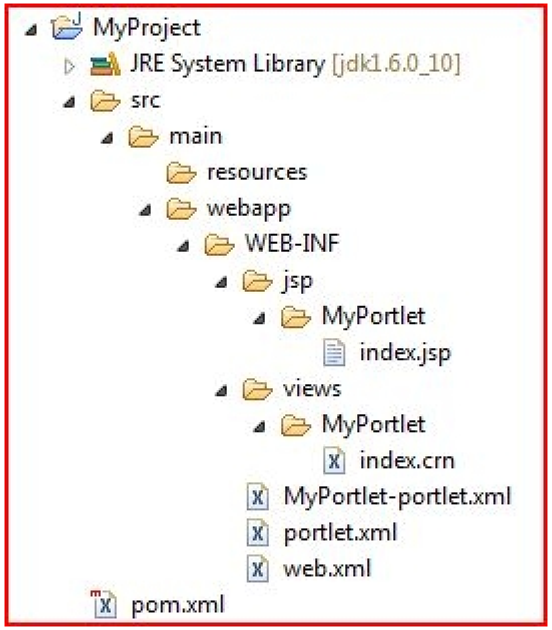
Hello, and welcome to MyServlet!

Done

Creating a New Portlet

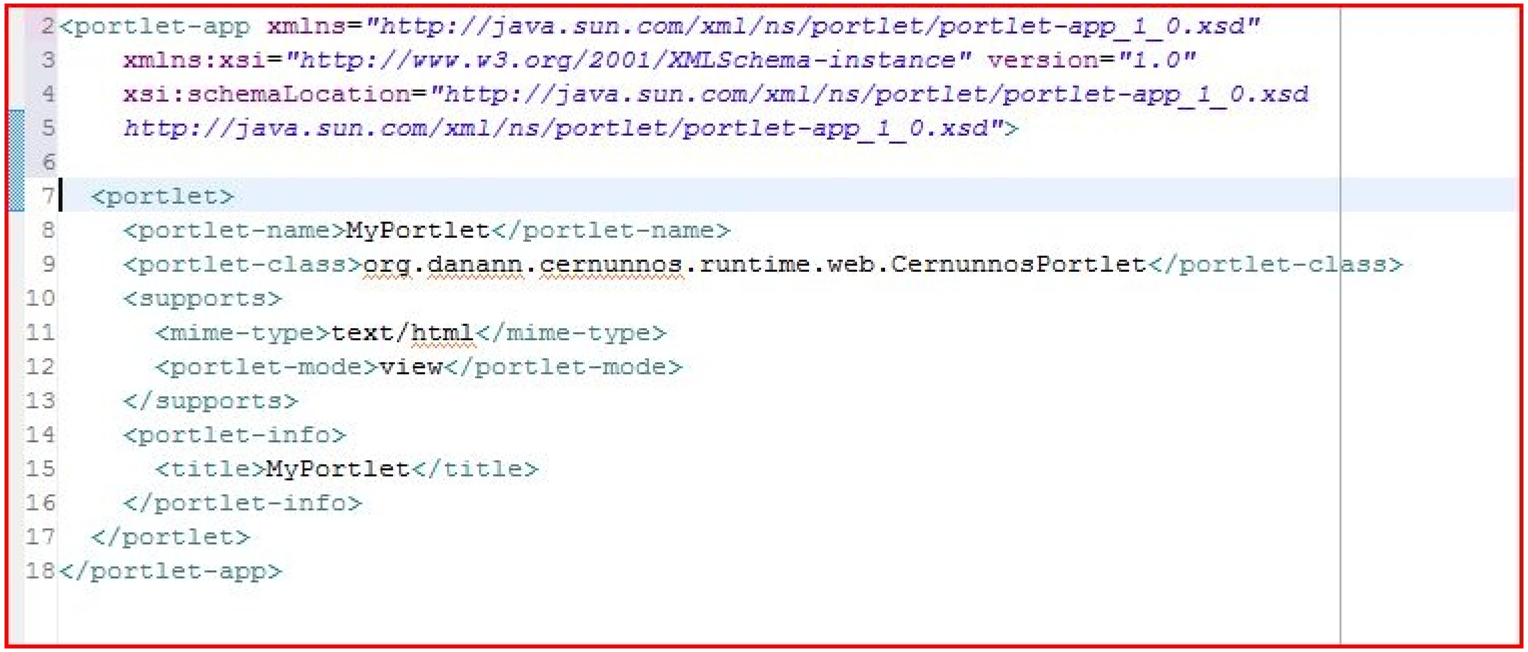
- For a minimal Portlet example you may:
  - **Provide a portlet.xml deployment descriptor**
  - Define a `<portlet>` in portlet.xml
  - Provide a JSP file for HTML markup
  - Provide a CRN file with a `<request-dispatcher>` that invokes your JSP
- You guessed it – there's a Cernunnos command (requires 1.1.0):
  - `>crn define-portlet.crn <portlet.name>`
- You will also get a *-portlet.xml context file (just like a Servlet)

Project Directory Structure



```
MyProject
  JRE System Library [jdk1.6.0_10]
  src
    main
      resources
      webapp
        WEB-INF
          jsp
            MyPortlet
              index.jsp
          views
            MyPortlet
              index.crn
          MyPortlet-portlet.xml
          portlet.xml
          web.xml
  pom.xml
```

portlet.xml File



```
<portlet-app xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
    xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd
    http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">

  <portlet>
    <portlet-name>MyPortlet</portlet-name>
    <portlet-class>org.danann.cernunnos.runtime.web.CernunnosPortlet</portlet-class>
    <supports>
      <mime-type>text/html</mime-type>
      <portlet-mode>view</portlet-mode>
    </supports>
    <portlet-info>
      <title>MyPortlet</title>
    </portlet-info>
  </portlet>
</portlet-app>
```

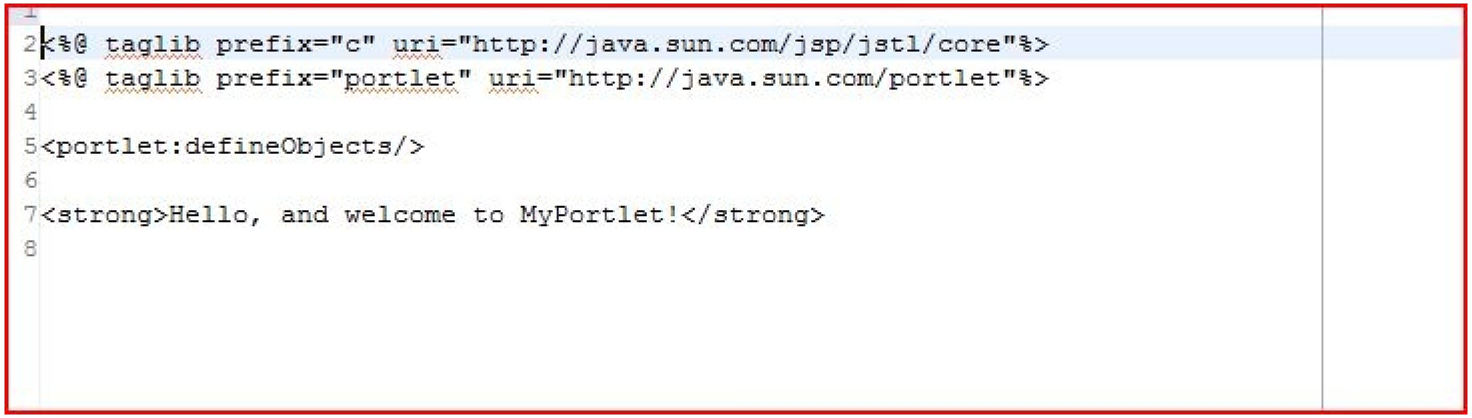
Notes on Defining a Portlet

- Specify `CernunnosPortlet` for `<portlet-class>`
- You can specify `contextConfigLocation` as an init parameter
- If you don't specify a context file, the servlet will:
  - Look for a context in `WEB-INF/*-portlet.xml` (default location)
  - Use all default settings
- You *cannot* specify `scriptLocation` the way Servlets can

Notes on Defining a Portlet (cont.)

- Cernunnos Portlets always operate in MVC-mode:
  - One (optional) action script (at `/WEB-INF/actions/${action}.crn` by default)
  - Followed by one view script (at `/WEB-INF/views/${view}.crn` by default)
- The "default default" view name is 'index'; you can specify a different default in the context file

index.jsp File



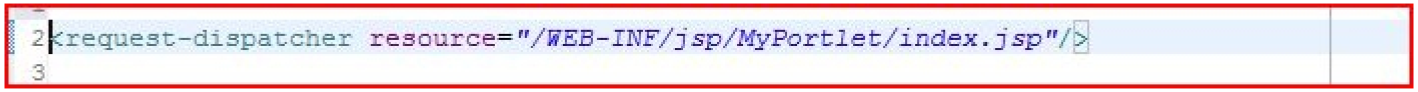
```

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="portlet" uri="http://java.sun.com/portlet"%>

<portlet:defineObjects/>

<strong>Hello, and welcome to MyPortlet!</strong>

```

Notes on JSP Files

- The following resources are pre-defined by define-portlet.crn
  - JSTL Core <taglib>
  - JSR-168 Portlet <taglib>
  - `<portlet:defineObjects/>` at top of page
- Request attributes that are in scope when `<request-dispatcher>` is invoked will be available in EL expressions (just like Servlets)

Notes on CRN Files

- Portlet CRN files work pretty much like Servlet CRN files



```
<request-dispatcher resource="/WEB-INF/jsp/MyPortlet/index.jsp"/>
```

- Exception: `<download>` is not available
- CernunnosPortlet provides some important request attributes:
  - WebAttributes.REQUEST: PortletRequest (Action- or Render-)
  - WebAttributes.REQUEST: PortletResponse (Action- or Render-)

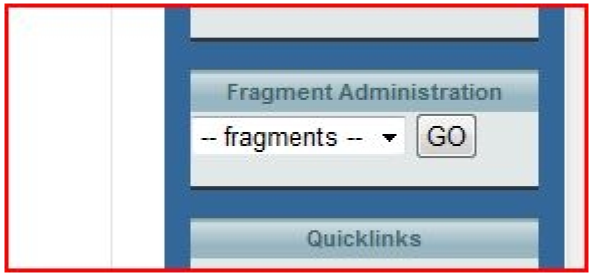
Servlets & Portlets in uPortal

Servlets & Portlets in uPortal 3.1.0

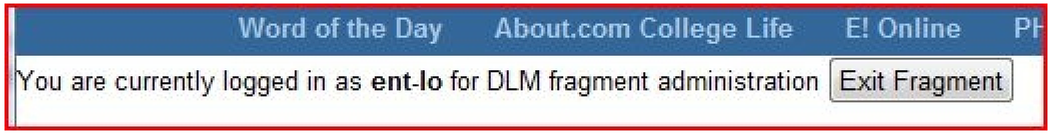
- The uPortal 3.1.0 release includes these Cernunnos-based Servlets & Portlets:
  - FragmentAdministration (Portlet)
  - ExitFragmentAdministration (Portlet)
  - ImportExportPortlet
  - ImportExportServlet

Fragment Administration

- FragmentAdministration allows authorized users to *impersonate* DLM fragment owner accounts with one click

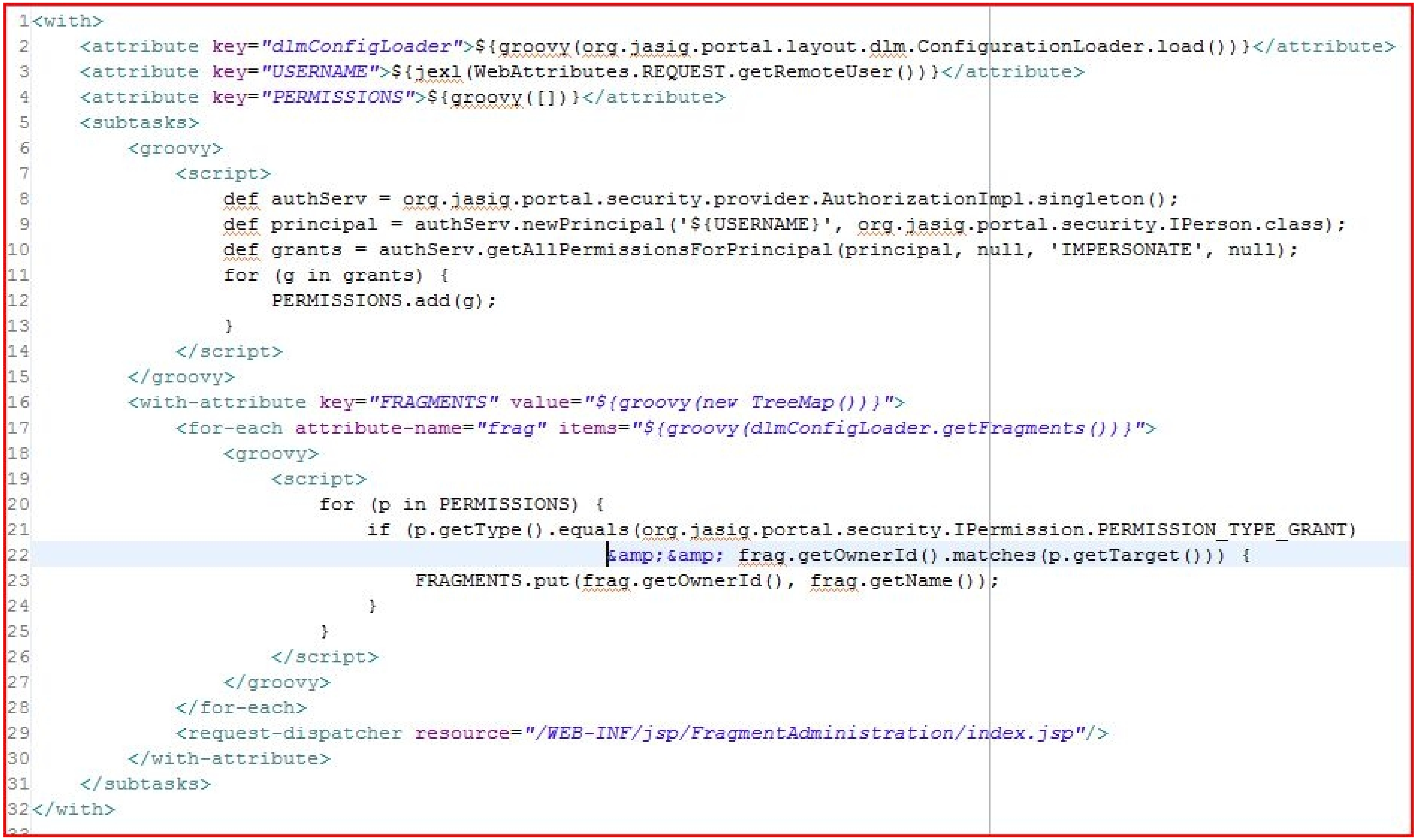


- ExitFragmentAdministration helps them become themselves again



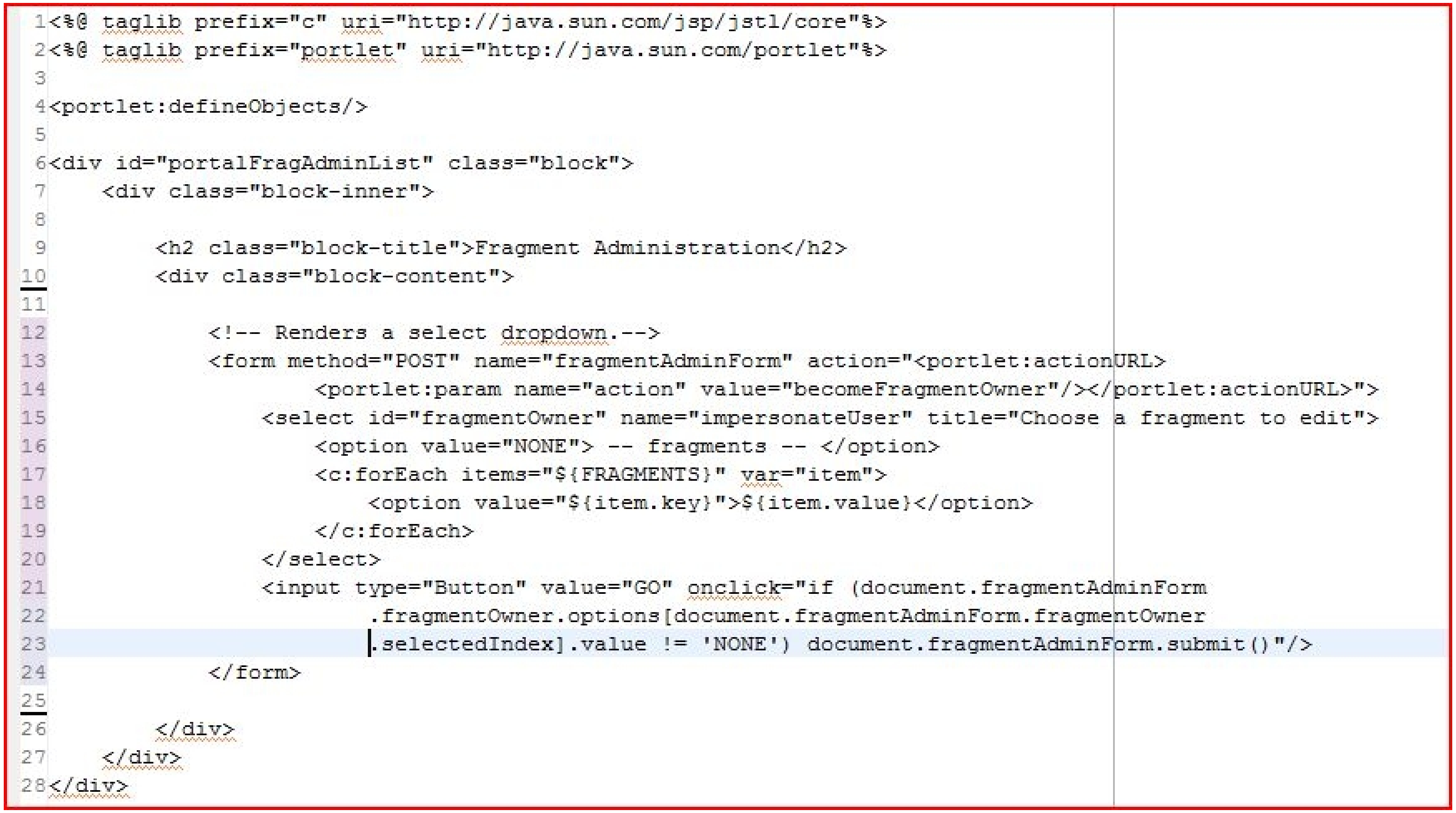
You are currently logged in as **ent-lo** for DLM fragment administration

FragmentAdministration index.crn



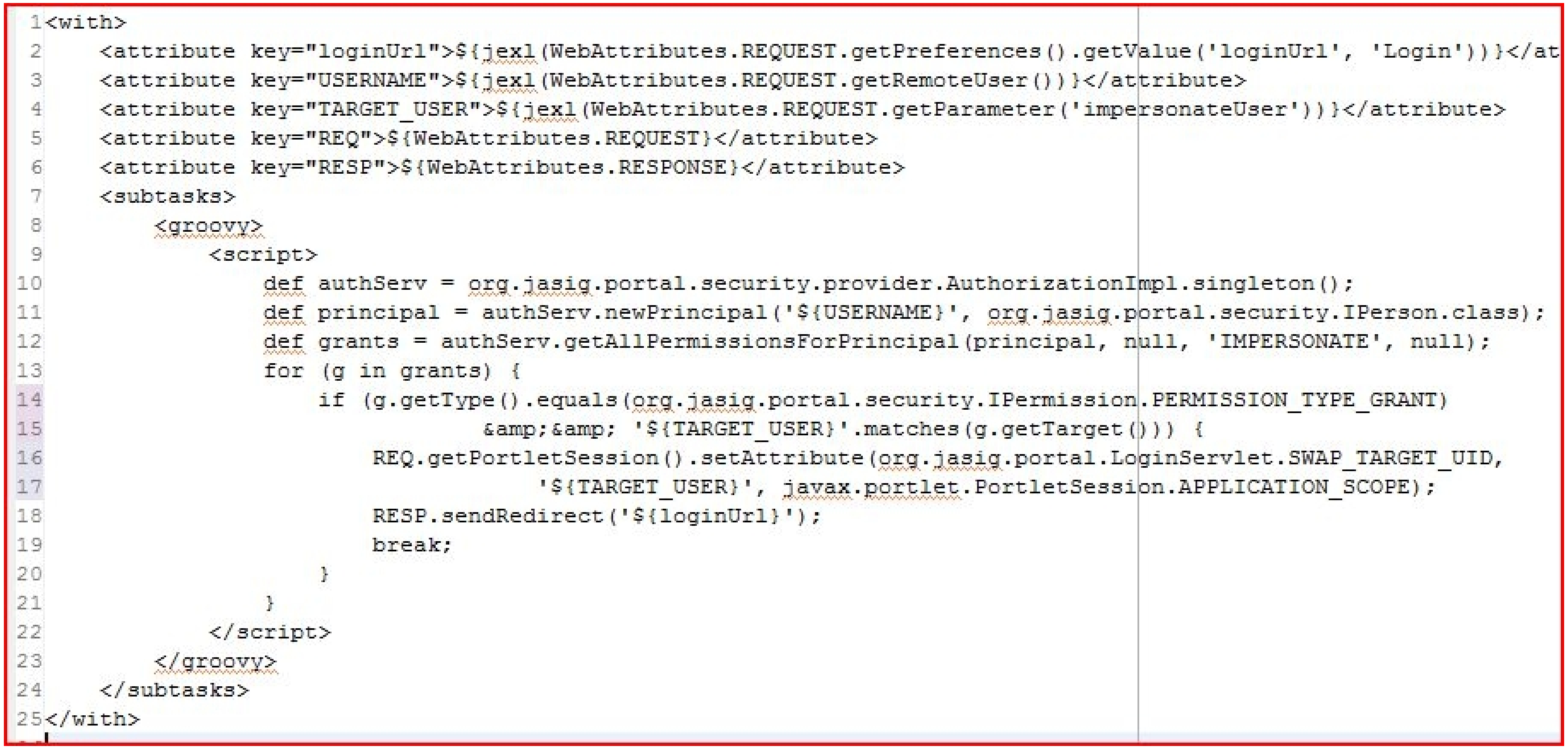
```
<with>
    <attribute key="dlmConfigLoader">${groovy(org.jasig.portal.layout.dlm.ConfigurationLoader.load())}</attribute>
    <attribute key="USERNAME">${jexl(WebAttributes.REQUEST.getRemoteUser())}</attribute>
    <attribute key="PERMISSIONS">${groovy([])}</attribute>
    <subtasks>
        <groovy>
            <script>
                def authServ = org.jasig.portal.security.provider.AuthorizationImpl.singleton();
                def principal = authServ.newPrincipal('${USERNAME}', org.jasig.portal.security.IPerson.class);
                def grants = authServ.getAllPermissionsForPrincipal(principal, null, 'IMPERSONATE', null);
                for (g in grants) {
                    PERMISSIONS.add(g);
                }
            </script>
        </groovy>
        <with-attribute key="FRAGMENTS" value="${groovy(new TreeMap())}">
            <for-each attribute-name="frag" items="${groovy(dlmConfigLoader.getFragments())}">
                <groovy>
                    <script>
                        for (p in PERMISSIONS) {
                            if (p.getType().equals(org.jasig.portal.security.IPermission.PERMISSION_TYPE_GRANT)
                                                &amp;&amp; frag.getOwnerId().matches(p.getTarget())) {
                                FRAGMENTS.put(frag.getOwnerId(), frag.getName());
                            }
                        }
                    </script>
                </groovy>
            </for-each>
            <request-dispatcher resource="/WEB-INF/jsp/FragmentAdministration/index.jsp"/>
        </with-attribute>
    </subtasks>
</with>
```

FragmentAdministration index.jsp



```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="portlet" uri="http://java.sun.com/portlet"%>

<portlet:defineObjects/>

<div id="portalFragAdminList" class="block">
    <div class="block-inner">

        <h2 class="block-title">Fragment Administration</h2>
        <div class="block-content">

            <!-- Renders a select dropdown.-->
            <form method="POST" name="fragmentAdminForm" action="<portlet:actionURL>
                    <portlet:param name="action" value="becomeFragmentOwner"/></portlet:actionURL>">
                <select id="fragmentOwner" name="impersonateUser" title="Choose a fragment to edit">
                    <option value="NONE"> -- fragments -- </option>
                    <c:forEach items="${FRAGMENTS}" var="item">
                        <option value="${item.key}">${item.value}</option>
                    </c:forEach>
                </select>
                <input type="Button" value="GO" onclick="if (document.fragmentAdminForm
                        .fragmentOwner.options[document.fragmentAdminForm.fragmentOwner
                        .selectedIndex].value != 'NONE') document.fragmentAdminForm.submit()"/>
            </form>

        </div>
    </div>
</div>
```

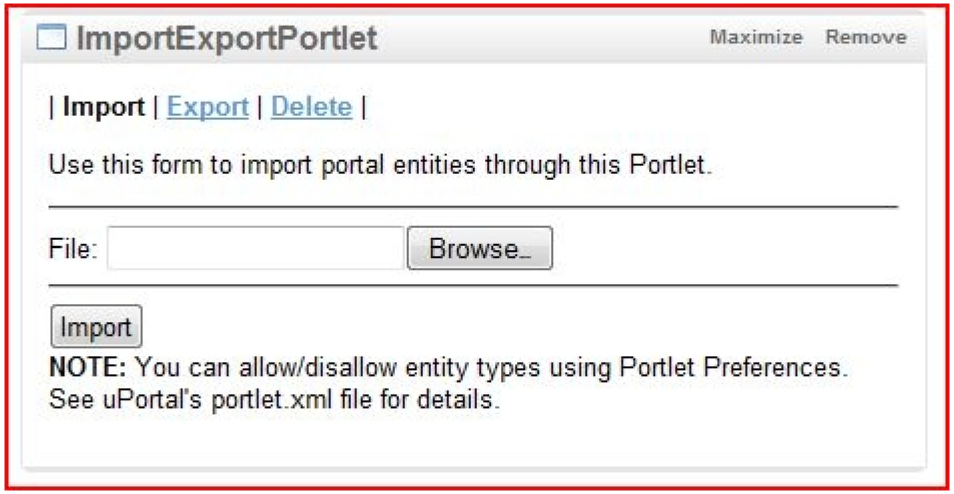
FragAdmin becomeFragmentOwner.crn



```
<with>
    <attribute key="loginUrl">${jexl(WebAttributes.REQUEST.getPreferences().getValue('loginUrl', 'Login'))}</at
    <attribute key="USERNAME">${jexl(WebAttributes.REQUEST.getRemoteUser())}</attribute>
    <attribute key="TARGET_USER">${jexl(WebAttributes.REQUEST.getParameter('impersonateUser'))}</attribute>
    <attribute key="REQ">${WebAttributes.REQUEST}</attribute>
    <attribute key="RESP">${WebAttributes.RESPONSE}</attribute>
    <subtasks>
        <groovy>
            <script>
                def authServ = org.jasig.portal.security.provider.AuthorizationImpl.singleton();
                def principal = authServ.newPrincipal('${USERNAME}', org.jasig.portal.security.IPerson.class);
                def grants = authServ.getAllPermissionsForPrincipal(principal, null, 'IMPERSONATE', null);
                for (g in grants) {
                    if (g.getType().equals(org.jasig.portal.security.IPermission.PERMISSION_TYPE_GRANT)
                                &amp;&amp; '${TARGET_USER}'.matches(g.getTarget())) {
                        REQ.getPortletSession().setAttribute(org.jasig.portal.LoginServlet.SWAP_TARGET_UID,
                                    '${TARGET_USER}', javax.portlet.PortletSession.APPLICATION_SCOPE);
                        RESP.sendRedirect('${loginUrl}');
                        break;
                    }
                }
            </script>
        </groovy>
    </subtasks>
</with>
```

ImportExport Portlet

- This Portlet provides access to uPortal Import/ Export capabilities from the portal UI



ImportExportPortlet

Maximize Remove

| **Import** | Export | Delete |

Use this form to import portal entities through this Portlet.

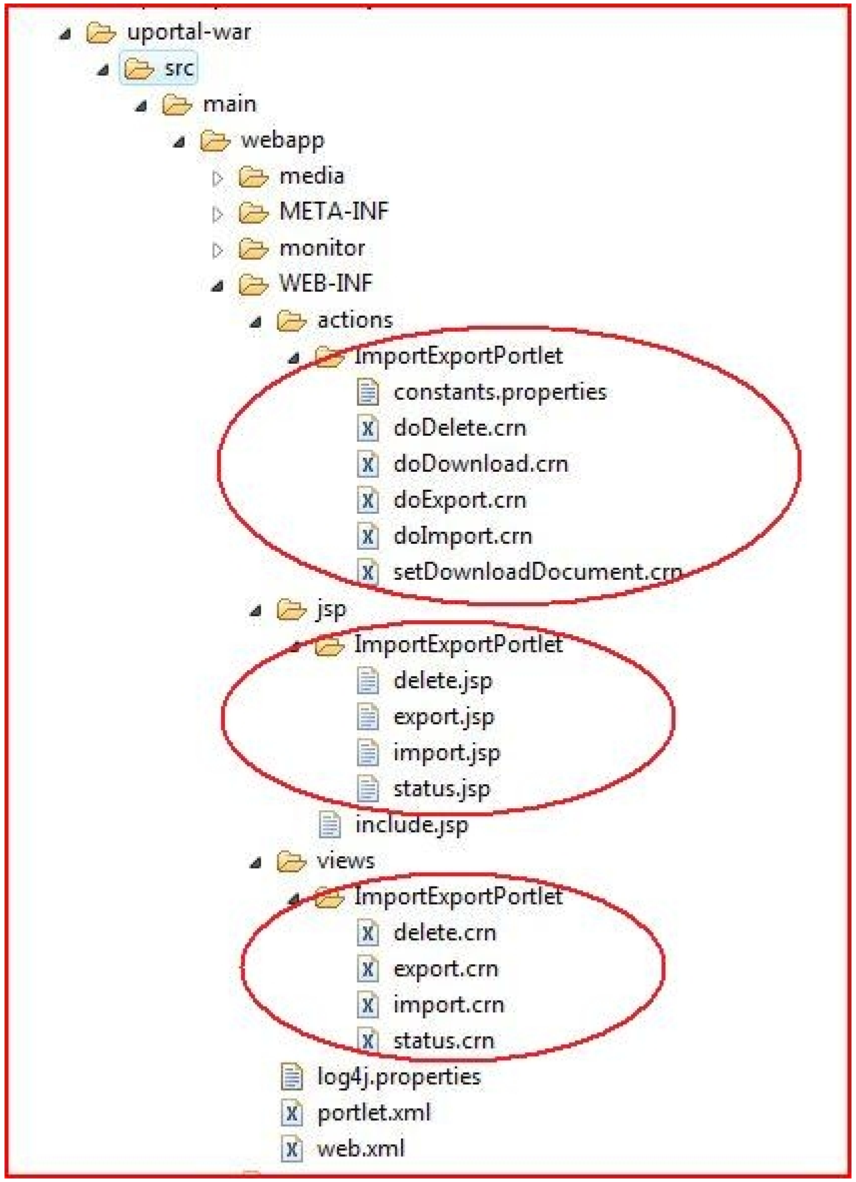
File: Browse...

Import

NOTE: You can allow/disallow entity types using Portlet Preferences. See uPortal's portlet.xml file for details.

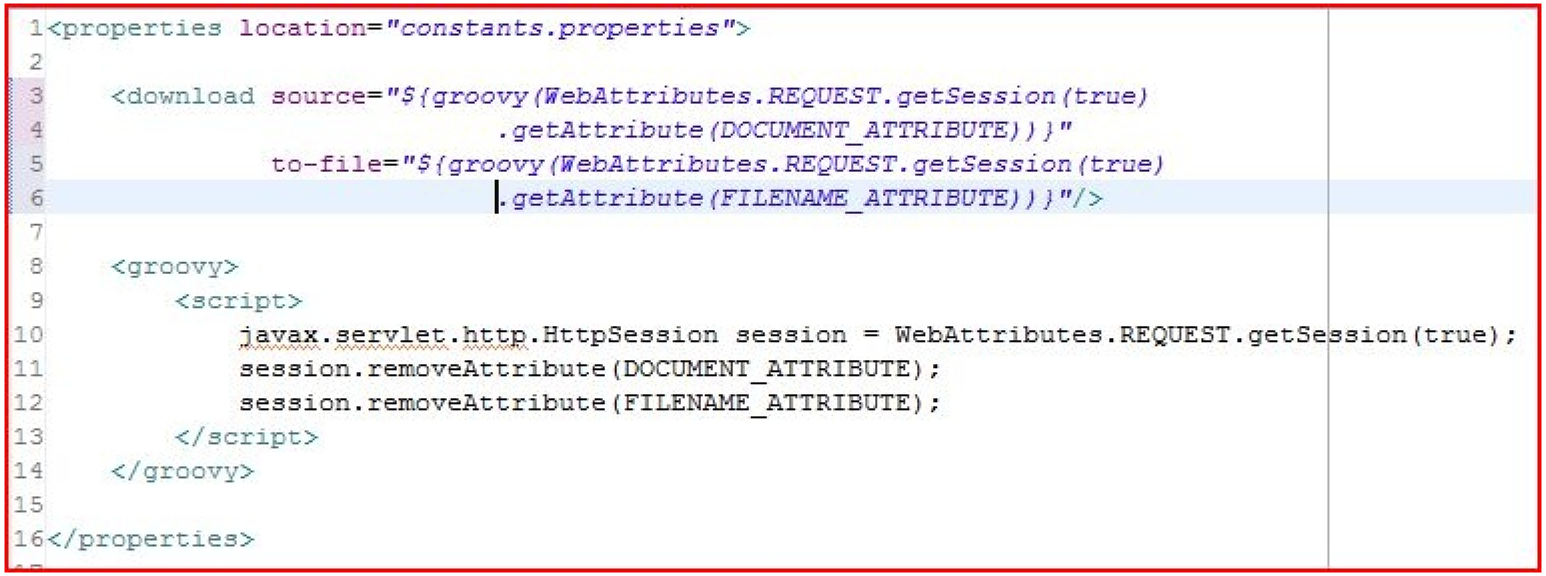
- You can restrict allowable operations at deploy/publish time with Portlet Preferences

ImportExportPortlet Directory Structure



```
uportal-war
  src
    main
      webapp
        media
        META-INF
        monitor
        WEB-INF
          actions
            ImportExportPortlet
              constants.properties
              doDelete.crn
              doDownload.crn
              doExport.crn
              doImport.crn
              setDownloadDocument.crn
          jsp
            ImportExportPortlet
              delete.jsp
              export.jsp
              import.jsp
              status.jsp
            include.jsp
          views
            ImportExportPortlet
              delete.crn
              export.crn
              import.crn
              status.crn
          log4j.properties
          portlet.xml
          web.xml
```

ImportExportServlet doDownload.crn



```
<properties location="constants.properties">

    <download source="${groovy(WebAttributes.REQUEST.getSession(true)
                            .getAttribute(DOCUMENT_ATTRIBUTE))}"
              to-file="${groovy(WebAttributes.REQUEST.getSession(true)
                            .getAttribute(FILENAME_ATTRIBUTE))}"/>

    <groovy>
        <script>
            javax.servlet.http.HttpSession session = WebAttributes.REQUEST.getSession(true);
            session.removeAttribute(DOCUMENT_ATTRIBUTE);
            session.removeAttribute(FILENAME_ATTRIBUTE);
        </script>
    </groovy>

</properties>
```

Some Metrics

I have argued this approach reduces busywork and bulk **dramatically**... is there any way we can test that claim?

- ImportExportPortlet

  Java: 0 files, 0 lines / XML: 5 files, 177 lines / CRN: 10 files, 399 lines

- BookmarksPortlet

  Java: 47 files, 4540 lines / XML: 9 files, 949 lines

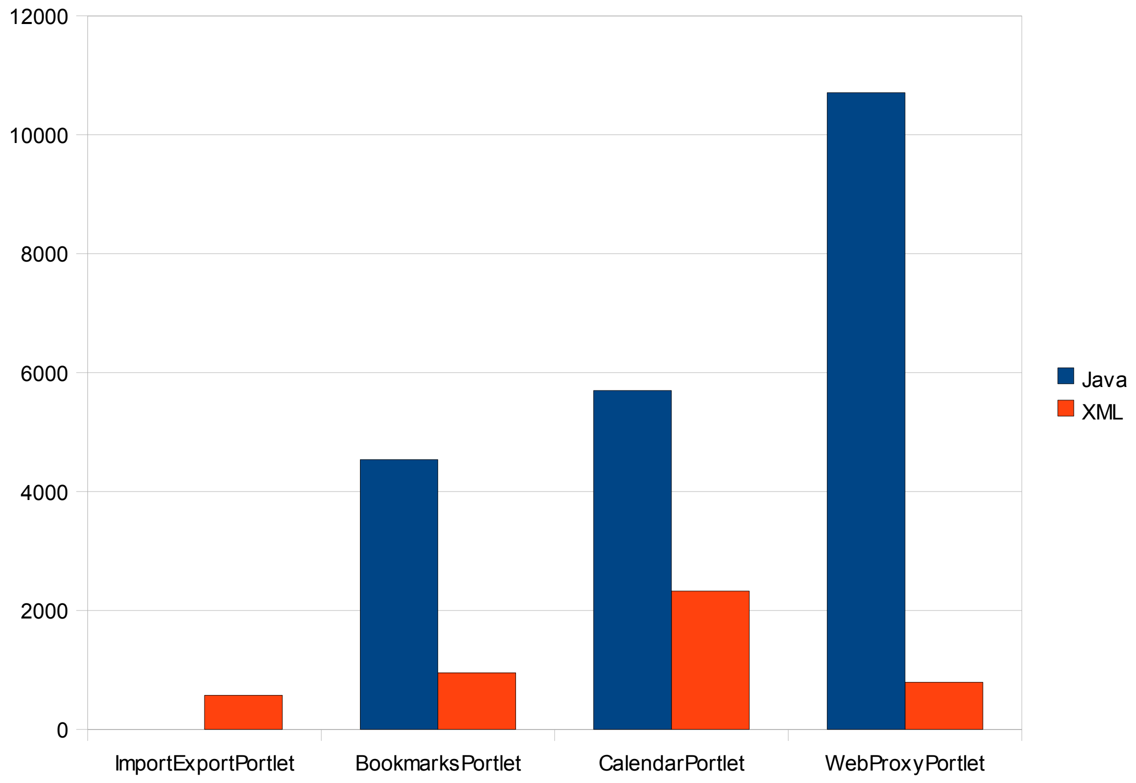
- CalendarPortlet

  Java: 50 files, 5700 lines / XML: 13 files, 2330 lines

- WebProxyPortlet

  Java: 64 files, 10710 lines / XML: 5 files, 793 lines

Some Metrics (cont.)



Questions?

Drew Wills

drew@unicon.net

http://cernunnos.googlecode.com

Hidden Bulk