Publisher

Alexis Grandemange

16 February, 2001

1 Foreword

Four actors are involved in the PageBox architecture:

- End user accessing presentation using browsers
- Content **providers** that write web applications and business lo gic. They can also host central application and databases
- ISP/ASP or companies **hosting** the presentation. These hosts **subscribe** to receive available presentations.
- Some of these ISP/ASP also act as **editors**. They allow content providers to **publish** their presentations.

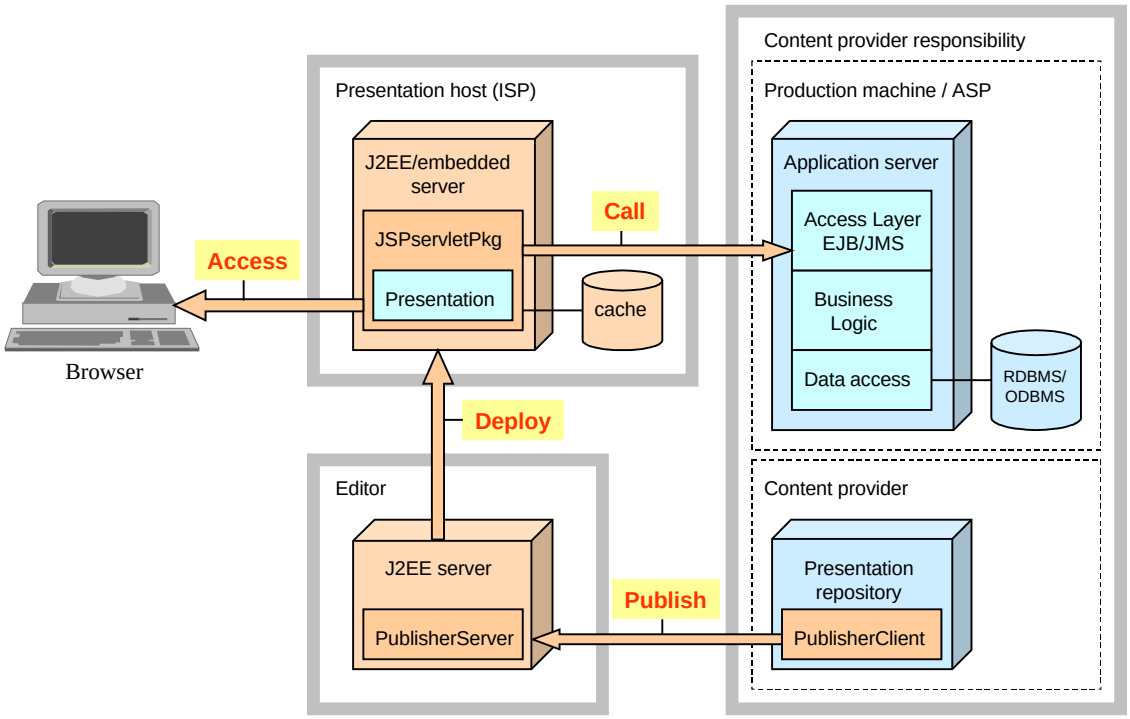

Figure 1: PageBox actors

The Content provider **publishes** the presentation to the Editor. The Editor **deploys** the archive to Presentation hosts. The End User **accesses** the Presentation that **calls** the business logic using for instance EJB or JMS.

In this document, we describe the Publication tool. It involves two parts:

- A PublisherServer web archive
- A PublisherClient Java application the content provider can download from the Editor site.

2 Protocol

The communication betwee n the PublisherClient and the PublisherServer uses HTTP.
It supports publishing and unpublishing, subscribe and unsubscribe.

Note that this protocol is designed to be used between PublisherClient and PublisherServer or from batch tools and not from a brow ser.

2.1 Publishing

Publishing involves two steps:

1. A sort of conditional GET to check if the archive is already published in format http://myEditorURL?todo=check&arch=myArchiv e&lm=time&size=archive_size where myArchive includes the extension .war or .jar and time is the time when the archive was modified in milliseconds since the epoch and archive_size is the size of the archive in bytes.
2. A POST including:
   a) The archive name
   b) The time when the archive was modified in milliseconds since the epoch
   c) The length of the archive
   d) The archive itself
   e) A variable area ended by a byte set at –1 that can contain a certificate and a policy
   f) A certificate area contains a byte set at 33, the length of the certificate and the certificate itself in a format that can be imported in a keystore
   g) A policy area contains a byte set a 22, the number of permissions and the permissions themselves

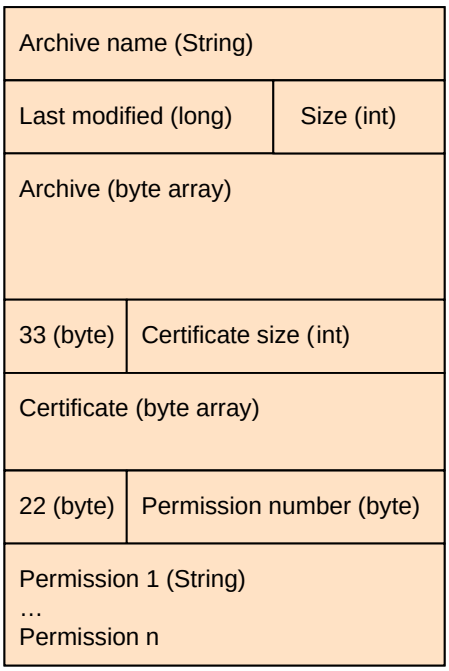

Figure 2: POST format

2.2 Unpublishing

Unpublishing involves a single GET request in format http://myEditorURL?todo=delete&arch=myArchive , where myArchive includes the extension .war or .jar.

2.3 Subscribe

Subscribe involves a single GET req uest in format http://myeditorurl/?todo=subscribe&deployURL=myJSPservlet , where myJSPservlet is a newly subscribed JSPservlet that must receive deployed archives.

2.4 Unsubscribe

Unsubscribe involves a single GET request in format http://myeditorurl/?todo=unsubscribe&deployURL=myJSPservlet , where myJSPservlet is a subscribed JSPservlet that must be removed.

3 PublisherServer

3.1 Configuration

```
<servlet>
        <servlet-name>publish</servlet-name>
        <servlet-class>PublisherServer.PublisherServlet</servlet -class>
        <init-param>
        <param-name>Repository</param-name>
        <param-value>D:/Inetpub/wwwroot/Tomcat</param-value>
        <description>Archive repository</description>
        </init-param>
        <init-param>
        <param-name>Maxlength</param-name>
        <param-value>155648</param-value>
        <description>Max archive size</descripti on>
        </init-param>
        <init-param>
        <param-name>toTrace</param-name>
        <param-value>TRUE</param-value>
         <description>Trace mode TRUE: trace</description>
         </init-param>
        <init-param>
        <param-name>Logfile</param-name>
        <param-value>C:/TEMP/publish.log</param-value>
        <description>Path of log file</description>
        </init-param>
        <init-param>
        <param-name>PageBoxURLs</param-name>
        <param-value>C:/TEMP/PageBoxURLs.ser </param-value>
        <description>Serialized state of publications and subscriptions</description>
        </init-param>
        <init-param>
        <param-name>repositoryURL</param-name>
        <param-value>localhost/Tomcat</param-value>
        <des cription>URL where to retrieve archives</description>
        </init-param>
    </servlet>

    <servlet>
        <servlet-name>publishLog</servlet-name>
        <servlet-class>PublisherServer.PublisherLog</servlet -class>
    </servlet>

    <servlet>
        <servlet-name>PublisherArchive</servlet-name>
        <servlet-class>PublisherServer.PublisherArchive</servlet -class>
    </servlet>
```

```
    <servlet-mapping>
        <servlet-name>publish</servlet-name>
        <url-pattern>/publish</url-pattern>
    </servlet-mapping>

    <servlet-mapping>
        <servlet-name>publishLog</servlet-name>
        <url-pattern>/publishlog</url-pattern>
    </servlet-mapping>

    <servlet-mapping>
        <servlet-name>PublisherArchive</servlet-name>
        <url-pattern>/archiveList</url-pattern>
    </servlet-mapping>
```

PublisherServer must know:
a) where to write the archive: *Repository*
b) where to deploy the archive: *PageBoxURLs*
c) how JSPservletPkg can access the archive: *RepositoryURL*

RepositoryURL is the URL to use to access Repository files.
Both *Repository* and *RepositoryURL* are managed by the Editor.

3.1.1 Repository

Repository is the path of the directory where PublishServer writes archives.
The Editor must create it before starting PublisherServer.

Default: none.
This parameter is mandatory.

3.1.2 PageBoxURLs

PageBoxURLs is the serialized state of archive publications and subscriptions.
It is a HashMap whose keys are JSPservlet URLs and values are of a PageBoxEntry class.

Implementation details:

A JSPservlet URL is prefixed by http:// and postfixed by /ServletUpdate.
PageBoxEntry is defined:

```
class PageBoxEntry {
    boolean toRemove = false;
    HashMap hm;
    PageBoxEntry() {
        hm = new HashMap();
    }
    PageBoxEntry(HashMap map) {
        hm = map;
    }
}
```

It contains two fields:
?? hm, a HashMap whose keys are archive names and values Byte fields

- toRemove. This field is used when an entity unsubscribes a JSPservlet. PublisherServlet tries to undeploy the archives it manages on that JSPservlet before removing the JSPservlet itself from PageBoxURLs. If that JSPservlet is not running, it cannot. In this case, it sets the toRemove flag. Then periodically it tries again to undeploy archives and remove JSPservlet. If a new archive is published, it doesn't deploy it to t he "toRemove" JSPservlet.

The Byte field can take three values:
- 1: the archive is deployed on the JSPservlet
- 0: the archive deployment is pending – for instance because the JSPservlet is not running
- -1: the archive undeployment is pending - for instance because the JSPservlet is not running

Default: none.
This parameter is mandatory.

3.1.3 RepositoryURL

URL where JSPservletPkg can retrieve the archive.

Default: none.
Mandatory.

3.1.4 Maxlength

Max archive length. If the archive is bigger, PublisherServer rejects the upload request.

Default: 8192.

3.1.5 ToTrace

If set to true or TRUE, PublisherServer logs intensively.

3.1.6 LogFile

Path of the file where PublisherServer logs.

3.2 Packaging

3.2.1 Sources

Package PublisherServer.
PublisherServlet.java: Servlet implementing PublisherServer.
PublisherLog.java: tool to display PublisherLog.
PublisherArchive.java: tool to display published archives.

3.2.2 Classes

PublisherArchive.class
PublisherLog.class
PublisherServlet$Log.class
PublisherServlet$PageBoxEntry.class
PublisherServlet$PageBoxURLHandler. class
PublisherServlet$Scanner.class
PublisherServlet.class

3.3 Troubleshooting

You can use PublisherLog to remotely display the PublisherServer log.

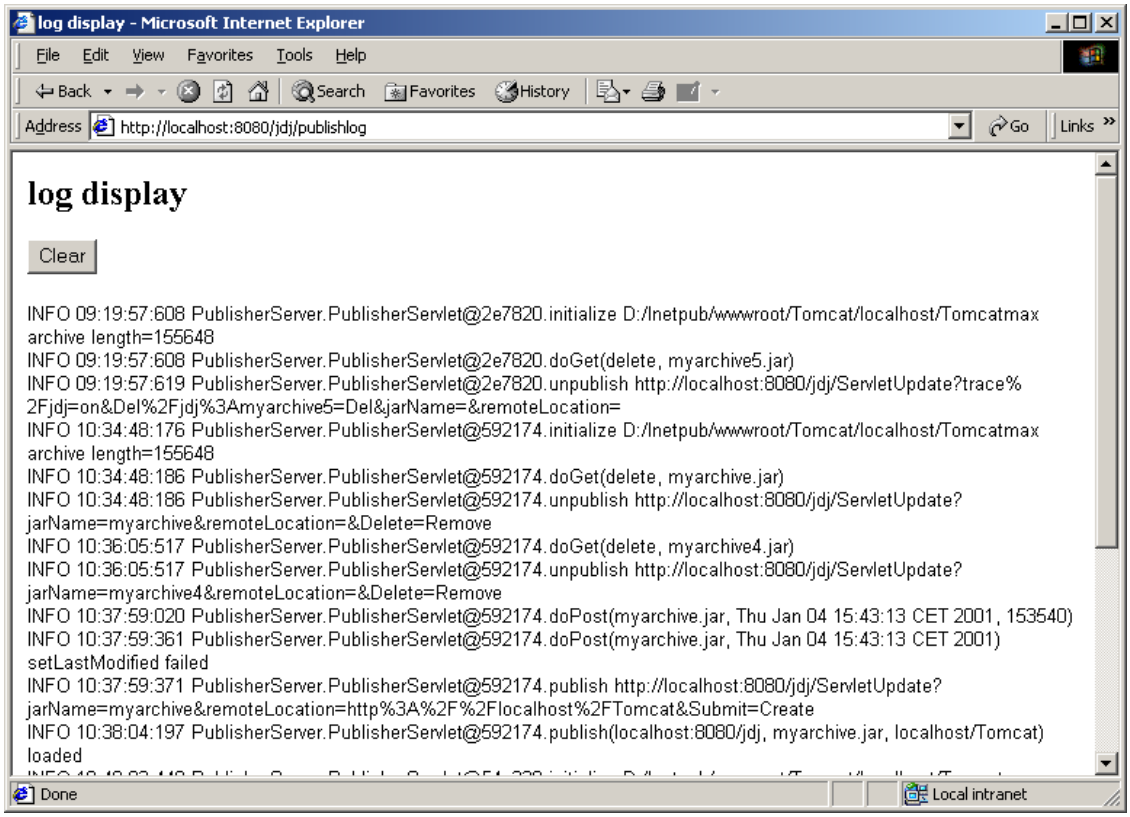

Figure 3: Log display

3.4 Note on implementation

PublisherServer tries to set the date of its local archive with the value sent by the Content provider. To do so, it uses File.setLastModified(). This call is only supported on some platforms. If it fails, it creates a file archive *-ext*.ts, where ext is the archive extension jar or war. This file contains:

1. A cookie, PageBox
2. The time it received from PublisherClient

To compare the date of the archive sent by the content provider with the date of the archive stored by the Editor, PublisherServer:

1. Extracts the last modified time of the local archive from archive*-ext*.ts if it exists
2. Otherwise gets the last modified time using File.getLastModified

All subscribed JSPservlet are defined with the same archives but possibly in different states as described in PageBoxURLs section.

If a JSPservlet is subscribed, it receives the same archives as already subscribed JSPservlet and not only the archives published after its subscription.

PublisherServer MUST be mapped on *publish*.
The reason is PublisherArchive and PublisherLog need to access to PublisherServer initialization parameters. If PublisherServer has not been called, PublisherArchive and PublisherLog use getServletContext().getRequestDispatcher("/ *publish*?todo=init") to get these parameters.

3.5 Subscription/publication display

If you call publish without parameter, you are displayed:

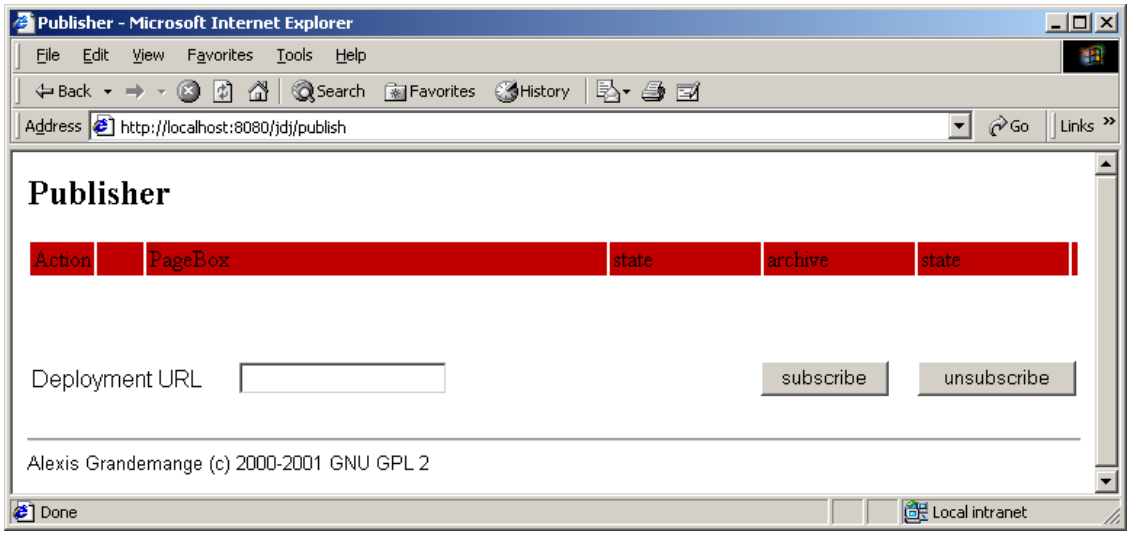

Figure 4: Subscription/publication display

You can subscribe a deployment URL.

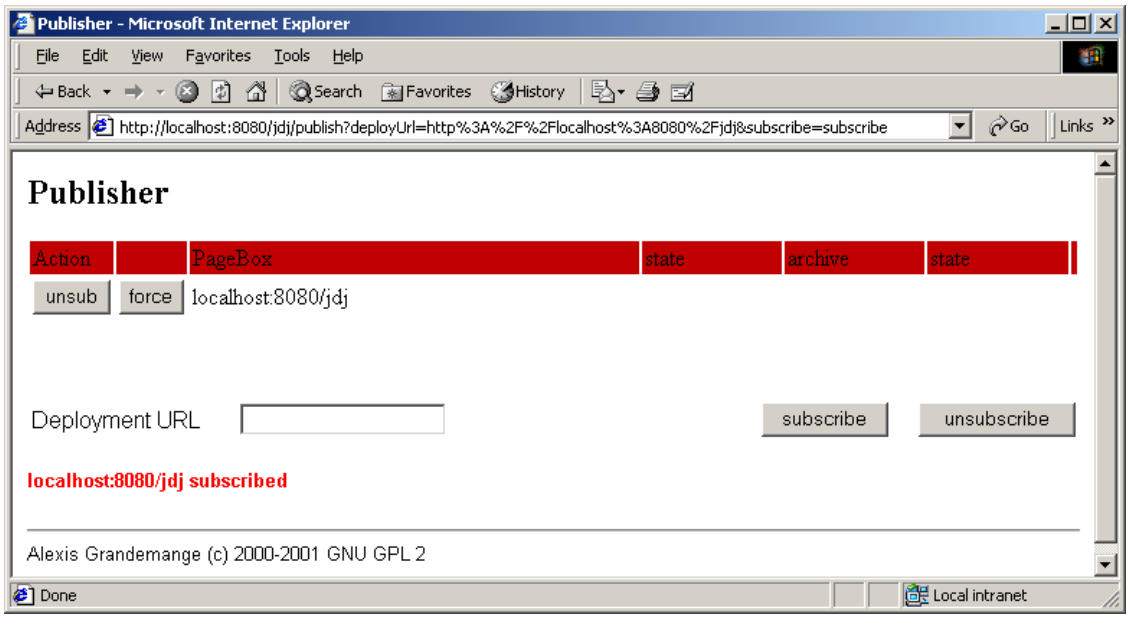

Figure 5: subscription display

You are displayed the screen above.
Note that if you prefix ed your deployment URL by http://, PublisherServer automatically removes it. You can now unsubscribe the deployed PageBox.

The difference between unsubscribe and force is:

?? Unsubscribe allows unsubscribing a valid PageBox that no longer subscribe to that p articular repository. In case of unsubscribe, the PageBox is contacted to remove the archives

?? Force removes the PageBox from the repository without contacting it. Use it if the PageBox no longer exist or if you misspelled its URL.

Consider the following case:

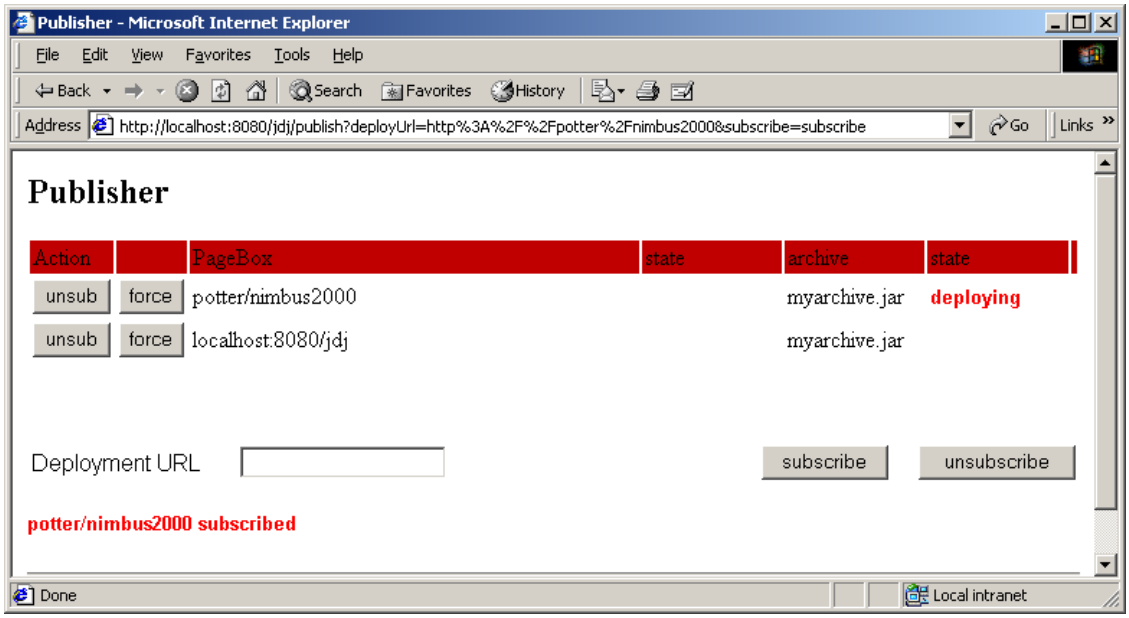

Figure 6: publications

We have two subscribers, localhost:8080/jdj and potter/nimbus2000 and an archive, myarchive.jar is published.

Potter/nimbus2000 doesn't exist, so myarchive is marked as publishing on potter/nimbus200 0. As PublisherServer cannot make a difference between a temporary unavailable PageBox and a PageBox that doesn't exist, it periodically tries to deploy the archive.

If we try to unsubscribe potter/nimbus2000, we get this:

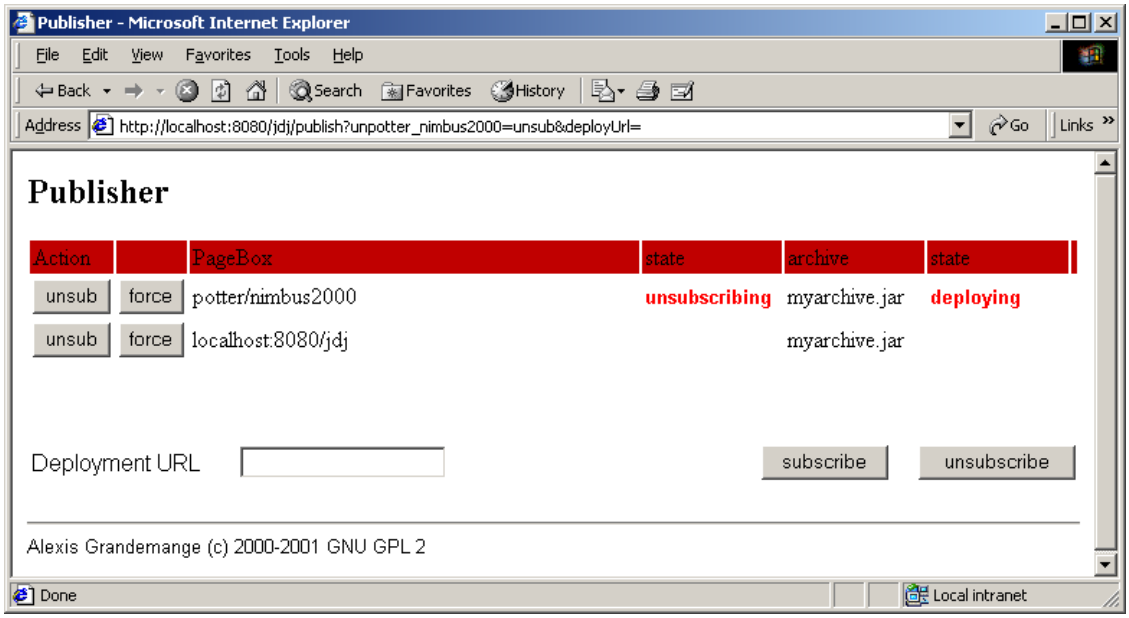

Figure 7: unsubscribing

In this case, use force.

Note:
This display shows information stored in PageBoxURLs's PageBoxEntries.

3.6 Archive display

PublisherArchive allows you to display archives deployed on a repository.

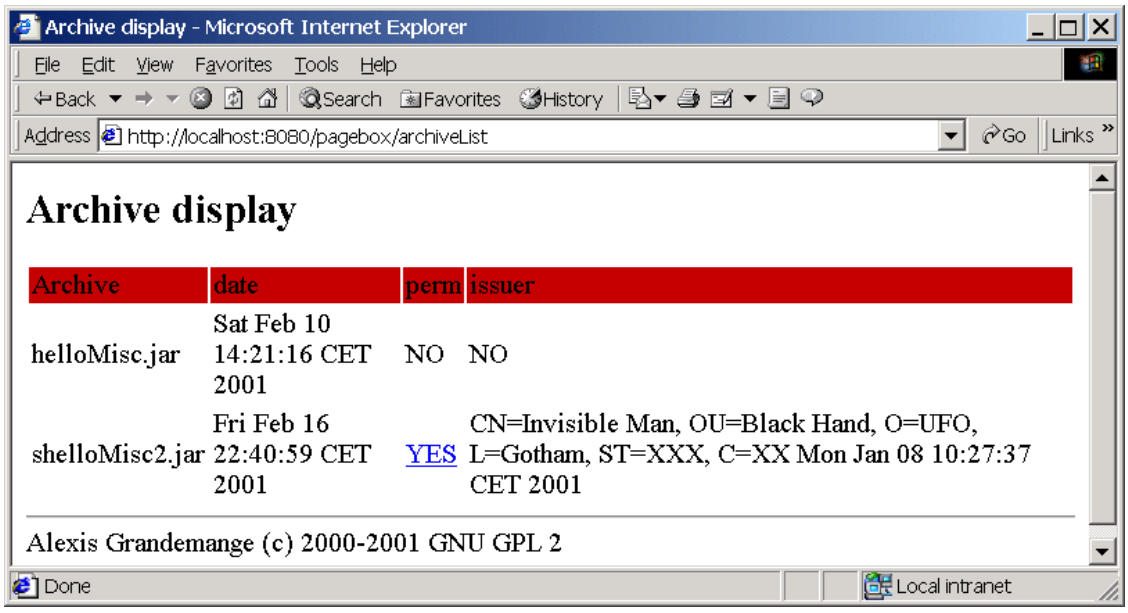

Figure 8: Archive list

You get the following information:
?? Archive name

- The last modified time of the archive on the publisher machine. If the publisher tries to publish again an archive with the same timestamp, PublisherClient will return “already published”
- Archive permissions if provided
- Issuer ID retrieved from the archive certificate if it exists

Note:
This display shows information retrieved from the repository directory.

If permission is set to yes, you can click on it to display the permission list:

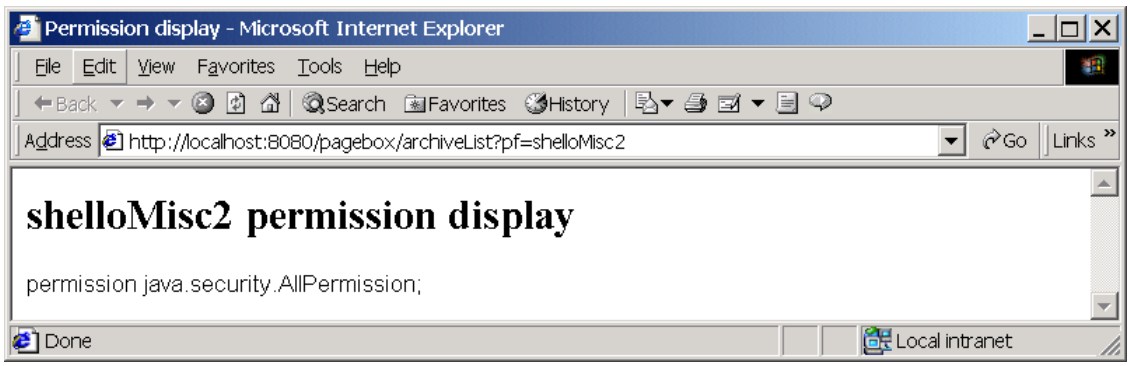

Figure 9: Permission list

4 PublisherClient

4.1 Use

PublisherClient is a Java application. Your CLASSPATH and PATH being properly set, you invoke it with:
java PublisherClient.PublisherClient.

Then the following window is displayed:

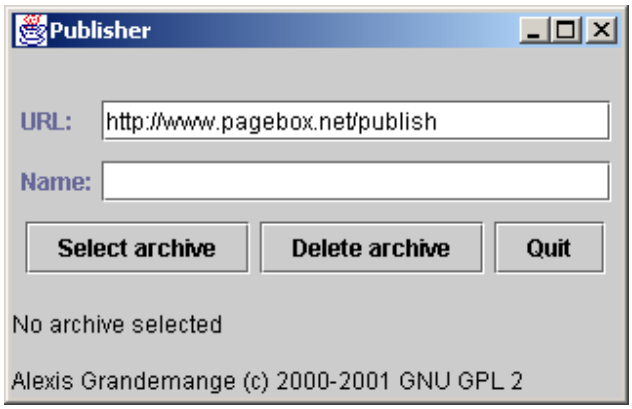

Figure 10: PublisherClient main window

You must set the URL field to the URL of PublisherServer.PublisherServlet on Editor side.

The second field should contain the name of the archive. Its use is mandatory only for unpublishing. If you don’t set the field at publication, it will be automatically set to the last part of

the archive file path. If your archive is stored in mydir/myarchive.jar, then your archive will be published as myarchive.jar.

If you don't set the extension, PublisherClien t assumes the extension is .jar.

There is a status line displaying the status of the last request or "No archive selected" at startup.

You can either unpublish with "Delete archive" button or publish with "Select archive" button. In the latter case, the following FileChooser window is displayed:

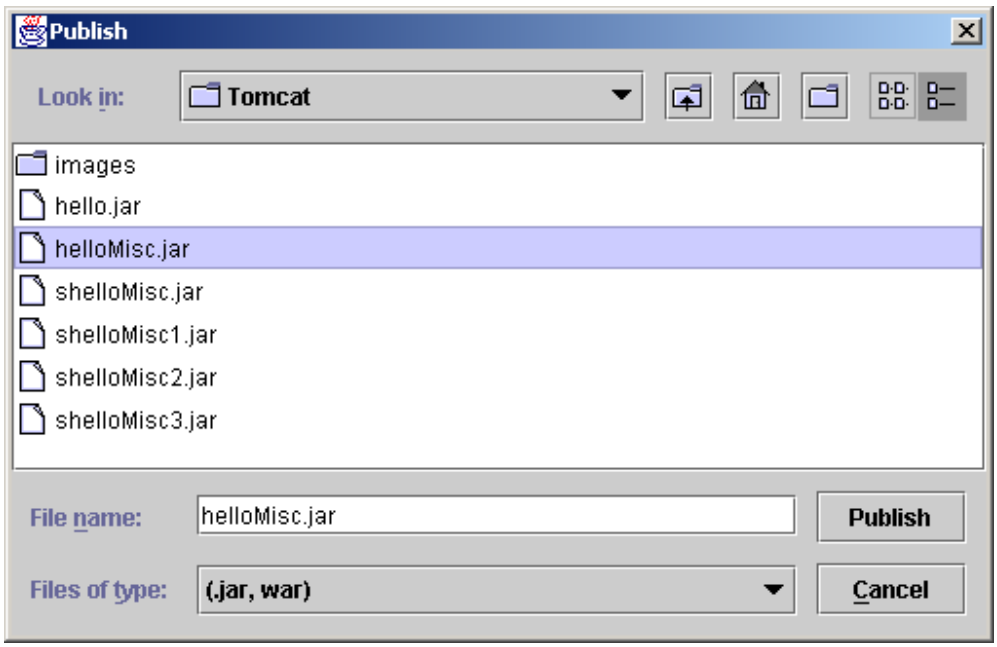

Figure 11: Archive selection

Once you have selected an archive, click on Publish button.

4.2 Packaging

4.2.1 Sources

Package PublisherClient.
PublisherClient.java: Java application.
PublisherFilter.java: a filter in charge to display only jar and war files.

4.2.2 Classes

PublisherClient$1.class
PublisherClient$DeleteArchive.class
PublisherClient$SelectArchive.class
PublisherClient.class
PublisherFilter.class

4.3 Note on implementation

When you select an arc hive, PublisherClient sends to PublisherServer:
1. The archive itself

2. Its certificate if it finds it. It has to be stored in the same directory as the archive and to be named archive_without_extension.cer.
3. Its policy or permission list defined according to Ja va 2 security. It has to be stored in the same directory as the archive and to be named archive_without_extension.policy.