# JSPservletPkg

Alexis Grandemange

Nov 20 2000

# TABLE OF CONTENT

# TABLE OF FIGURES

# 1 Foreword

JSPservletPkg is a complete implementation for servlets and JSP handling from a remot e repository and with dynamic update.

# 2 Parameters

JSPservletPkg is designed to be deployed in .war files.
There is no limitation to the number of war files, which can be configured, in a given Application Server.

Example:

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 1.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
   <servlet>
      <servlet-name>JSPservlet</servlet-name>
      <servlet-class>JSPservletPkg.JSPservlet</servlet-class>
      <init-param>
      <param-name>cachePath</param-name>
      <param-value>C:/temp/cachePath</param-value>
      <description>local cache</description>
      </init-param>
      <init-param>
      <param-name>toTrace</param-name>
      <param-value>TRUE</param-value>
      <description>trace activation</description>
      </init-param>
      <init-param>
      <param-name>toStat</param-name>
      <param-value>TRUE</param-value>
      <description>stat activation</description>
      </init-param>
      <init-param>
      <param-name>ID</param-name>
      <param-value>1</param-value>
      <description>unique identifier (PageBoxAPI)</description>
      </init-param>
      <init-param>
      <param-name>logfile</param-name>
      <param-value>C:/temp/cachePath/log.txt</param-value>
      <description>trace location</description>
      </init-param>
      <init-param>
      <param-name>remoteLocations</param-name>
      <param-value>C:/temp/cachePath/jdj2.properties</param-value>
      <description>jar remote location</description>
      </init-param>
      <init-param>
      <param-name>Publishers</param-name>
      <param-value>C:/temp/jdj2-publisher.properties</param-value>
      <description>publisher location per archive</description>
      </init-param>
```

```
<init-param>
        <param-name>expiration</param-name>
        <param-value>30</param-value>
        <description>resource expiration time</description>
        </init-param>
        <init-param>
        <param-name>allPermissionPolicy</param-name>
        <param-value>C:/temp/Resin/allPermission.policy</param -value>
        <description>policy file granting all rights</description>
        </init-param>
        <init-param>
        <param-name>defaultPolicy</param-name>
        <param-value>C:/temp/Resin/default.policy</param -value>
        <description>policy file for archives without explicit policy</description>
        </init-param>
        <init-param>
        <param-name>keystore</param-name>
        <param-value>keystore</param-value>
        <description>keystore location + download policy</description>
        </init-param>
        <init-param>
        <param-name>keystorePassword</param-name>
        <param-value>keystorePswd</param-value>
        <description>keystore password</description>
        </init-param>
        <init-param>
        <param-name>CAURL</param-name>
        <param-value>ldap://localhost/…</param -value>
        <description>CA Certificate LDAP URL</descriptio n>
        </init-param>
        <init-param>
        <param-name>CRLURL</param-name>
        <param-value>ldap://localhost/…</param -value>
        <description>Certificate Revocation List LDAP URL</description>
        </init-param>
        <init-param>
        <param-name>CRLLDAPuser</param-name>
        <param-value>…</param-value>
         <description>CRL LDAP user</description>
         </init-param>
        <init-param>
        <param-name>CRLLDAPpasswd</param-name>
        <param-value>…</param-value>
         <description>CRL LDAP password</description>
         </init-param>
        <init-param>
        <param-name>CALDAPuser</param-name>
        <param-value>…</param-value>
         <description>CA certificate LDAP user</description>
         </init-param>
        <init-param>
        <param-name>CALDAPpasswd</param-name>
        <param-value>…</param-value>
        <description> CA certificate LDAP password</description>
        </init-param>
        <init-param>
```

```
        <param-name>CRLperiod</param-name>
        <param-value>30</param-value>
        <description>Certificate Revocation List scan period</description>
        </init-param>
    </servlet>
    <servlet>
        <servlet-name>ServletLog</servlet-name>
        <servlet-class>JSPservletPkg.ServletLog</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ServletStat</servlet-name>
        <servlet-class>JSPservletPkg.ServletStat</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>ServletUpdate</servlet-name>
        <servlet-class>JSPservletPkg.ServletUpdate</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>JSPupdate</servlet-name>
        <servlet-class>JSPupdate</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>ServletUpdate</servlet-name>
        <url-pattern>/ServletUpdate</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ServletLog</servlet-name>
        <url-pattern>/ServletLog</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ServletStat</servlet-name>
        <url-pattern>/ServletStat</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>JSPupdate</servlet-name>
        <url-pattern>/JSPupdate</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>JSPservlet</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

It includes five servlet definitions, ServletUpdate, ServletLog, ServletStat, JSPservlet and JSPupdate.
JSPupdate is a compiled JSP stored outside the JSPservlet package. It must be configured with an url-pattern /JSPupdate.
ServletUpdate is a more advanced administration tool with the ability to display installed, loaded archives and to set trace mode. It must be configured with an url-pattern /ServletUpdate.
ServletLog allows browsing remotely the log and to clear it.
ServletStat allows displaying statistics about servlets invocations

The package must contain:

```
WEB-INF
   Web.xml  // as the file above
   classes
      JSPupdate.class
      JSPservletPkg
         CRLchecker.class
         JSPhandler$ClassEntry$ServletInfo.class
         JSPhandler$ClassEntry$Stat.class
         JSPhandler$ClassEntry.class
         JSPhandler$Log.class
         JSPhandler$Scanner.class
         JSPhandler.class
         JSPloader$ClassInfo.class
         JSPloader$ProtectionDomainInfo.class
         JSPloader$ResourceEntry.class
         JSPloader$1
         JSPloader.class
         JSPloaderException.class
         JSPservlet$IncludeWriter.class
         JSPservlet.class
         JSPservlet$1
         PageBoxAPI.class
         ServletLog.class
         ServletStat.class
         ServletUpdate.class
         ResourcePrivilegedAction.class
         SoapUpdate.class (Tomcat version only)
```

We detail below the parameters it handles.

## *2.1 cachePath*

cachePath is the location where jars are locally stored after been retrieved from remote location.
Default value: C:/temp.

## *2.2 toTrace*

Tells if the tool must write diagnostic messages.
Default value: false.

## *2.3 toStat*

Tells if the tool must record statistics.
Default value: false.
If true, statistics are recorded **per archive** in a file *cachePath*/*archive*.stat, which is a property file, for instance:

```
#Sun Nov 19 23:20:32 CET 2000
TestServlet/FileAccess=3
helloImg=8
TestServlet/OtherServlet=2
SnoopJSP=3
TestServlet/ForwardingServlet=4
SnoopServlet=2
```

### 2.4 ID

Allows the deployer to specify a unique identifier.
PageBoxAPI allows retrieving this ID.
Application: deployment of a large number of instances.

No default value.

### 2.5 logfile

Tells where the tool must write diagnostic messages.
Default value: $CachePath/log.txt.

### 2.6 remoteLocations

Location of a property file containing jar names and associated URLs.
Default value: $CachePath/$ContextPath.properties where ContextPath indicated the name the war file is deployed with.

JSPupdate, ServletUpdate and publication update this file. Though we don't recommend it, you can modify it manually but don't expect to retrieve your comments.

### 2.7 remoteLocations

Location of a property file containing for each archive the URL of the publisher that published it.
Default value: $CachePath/$ContextPath -publisher.properties where ContextPath indicated the name the war file is deployed with.

### 2.8 expiration

You can set this parameter to minimize the round trip number between the browser and the server.
JSPservlet sets the *Expire* header field of static content (content with an extension different of *class*). It computes the *Expire* as current_time + *expiration*.
*expiration* unit is second.

Default value: 5 seconds.

### 2.9 allPermissionPolicy

If it is set, *allPermissionPolicy* is the path to a policy file with syntax conforming to the Java 2 security specification. If *defaultPolicy* is also set it means:

1. JSPservletPkg will implement sandboxes. So every archive will run with the permissions defined either in *cachePath*/*archive*.policy or in *cachePath*/java.policy where:
   - cachePath is the cachePath initialization parameter value
   - archive is the archive name without suffix
2. The Java server itself will run with the permission described in *allPermissionPolicy*.

The following no-brainer *allPermissionPolicy* will work in all cases:

```
grant {
        permission java.security.AllPermission;
};
```

**Note that you should never grant permissions in *cachePath*/*archive* policy as these files are downloaded from the archive location.**

Default: the parameter has no default value.

### 2.10 defaultPolicy

If it is set, *defaultPolicy* is the path to a policy file with syntax conforming to the Java 2 security specification. If *allPermissionPolicy* is also set it means:

1. JSPservletPkg will implement sandboxes. So every archive will run with the permissions defined either in *cachePath*/*archive*.policy or in *cachePath*/java.policy where:
   - cachePath is the cachePath initialization parameter value
   - archive is the archive name without suffix

   If no policy applies to the archive, then *defaultPolicy* is used
2. The Java server itself will run with the permission described in *allPermissionPolicy*.

The Java server itself will run with the permission described in *allPermissionPolicy*.

Default: the parameter has no default value.

### 2.11 keystore

*keystore* is the name of the key store in cachePath directory, for instance “keystore” but not “/mydir/keystore” or “mydir/keystore”.

If *keystore* is set, when JSPservletPkg downloads an archive, it tries

- To download a certificate from the same location as the archive and named *archive*.cer. If it finds, it adds the certificate to *cachePath*/*keystore*, which has to be in Sun JKS format with an *archive* alias
- To download a permission file from the same location as the archive and named *archive*.policy. This file should only contain permission entries. JSPservletPkg adds a keystore line, builds the appropriate grant line and stores it in *cachePath*/*archive*.policy in order to implement a sandbox with the permissions requested by the provider.

The archive user has no longer to administrate security. It is appropriate for trusted providers.

Default: the parameter has no default value.

### 2.12 keystorePassword

Password JSPservletPkg uses to access the keystore.

Default: the parameter has no default value.

### 2.13 CAURL

When JSPservletPkg implements sandboxes, it processes signed archives and retrieve classes certificate chain. If CAURL is set, it connects to this URL and expects to retrieve a Certificate Authority certificate used in the classes certificate chain.

If it fails to connect to the CA or if what it retrieves is not a certificate, it logs an ERROR entry with “directory access failure”. If the certificate is valid but not present in a class certificate chain, it invalidates the class just as if one of its certificates was revoked.

**Note that JSPservletPkg doesn’t load the class and therefore doesn’t raise a security but a class not found exception.**

If you set this parameter, you MUST add JNDI to your Java Server CLASSPATH in JDK 1.2. In JDK 1.3, you don’t have to, as JNDI is included in JDK.

Default: the parameter has no default value.

### 2.14 CRLURL

When JSPservletPkg implements sandboxes, it processes signed a rchives and retrieve classes certificates.
If CRLURL is set, it connects to this URL and expects to retrieve a Certificate Revocation List (CRL) used to check if a class certificate is revoked.

If it fails to connect to the CA or if what it retrieves is n ot a CRL, it logs an ERROR entry with "directory access failure". If the CRL is valid and it finds one of the class certificates in it, it invalidates the class.

**Note JSPservletPkg doesn't load the class and therefore doesn't raise a security but a class not found exception.**

If you set this parameter, you MUST add JNDI to your Java Server CLASSPATH in JDK 1.2. In JDK 1.3, you don't have to, as JNDI is included in JDK.

Default: the parameter has no default value.

### 2.15 CALDAPuser

Principal used to connect to th e Directory server to retrieve CAURL.

Default: the parameter has no default value. If it is not set, the tool connects to the Directory server without credential and password (LDAPpasswd).

### 2.16 CALDAPpasswd

Password used to connect to the Directory server to r etrieve CAURL.

Default: the parameter has no default value.

### 2.17 CRLLDAPuser

Principal used to connect to the Directory server to retrieve CRLURL.

Default: the parameter has no default value. If it is not set, the tool connects to the Directory server without credential and password (LDAPpasswd).

### 2.18 CRLLDAPpasswd

Password used to connect to the Directory server to retrieve CRLURL.

Default: the parameter has no default value.

### 2.19 CRLperiod

Defines how often the tool will connect to check for CRL updates in seconds.

Default: 7 * 24 * 3600 (1 week).

## 3 Statistics and troubleshooting

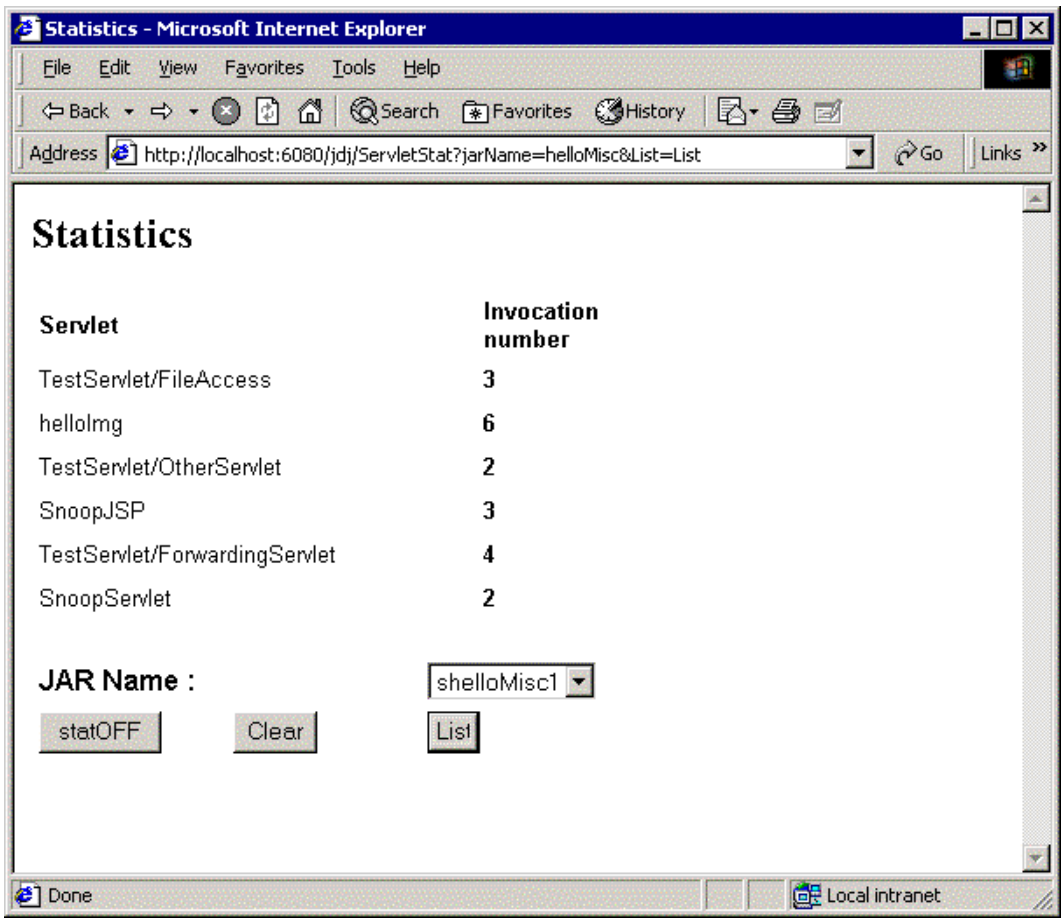


**Figure 1: Statistics using ServletStat**

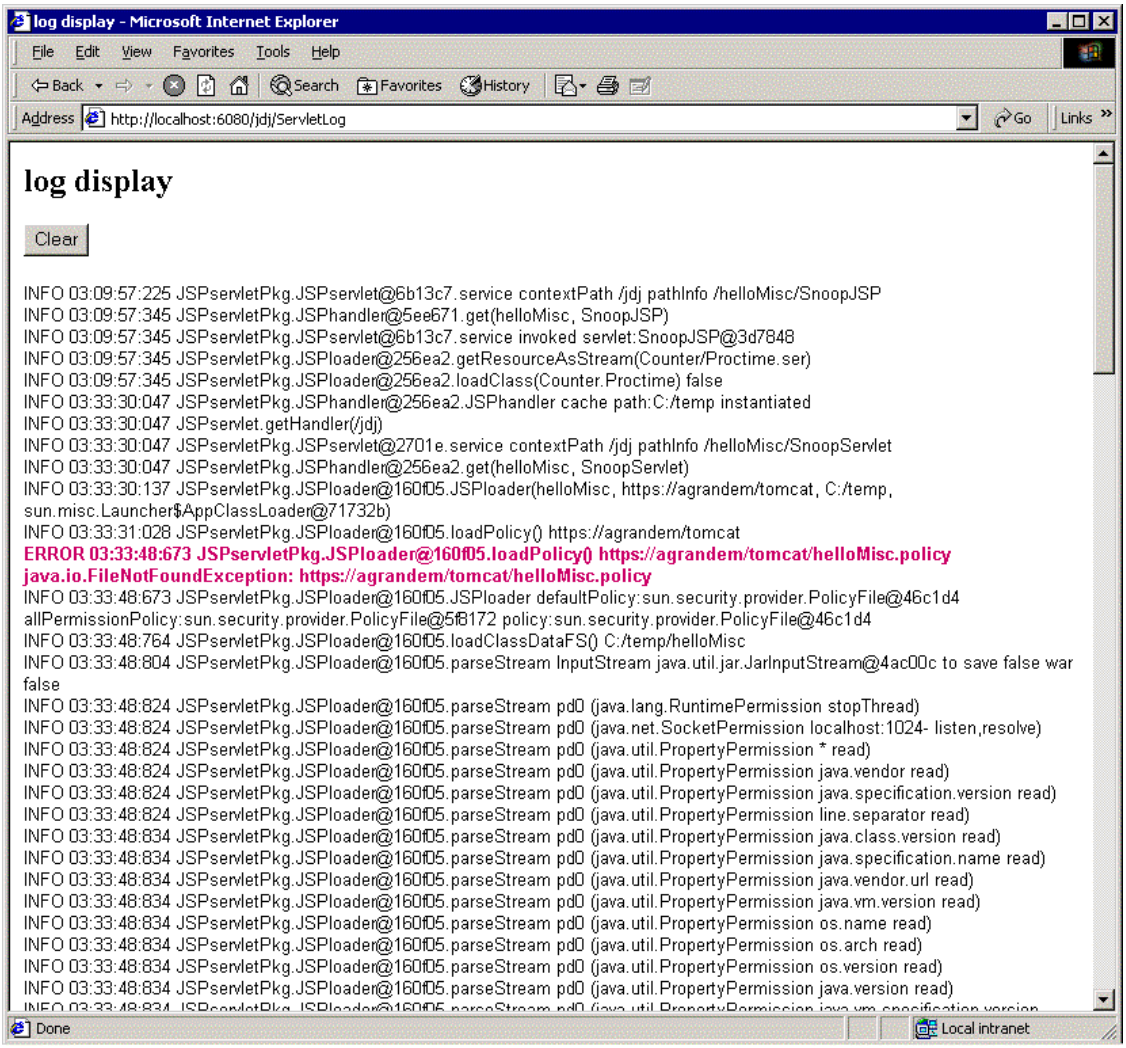


**Figure 2: log using ServletLog**

# 4 Notes on implementation

## *4.1 Dynamic update*

Dynamic update of jar file is supported though JSPupdate.

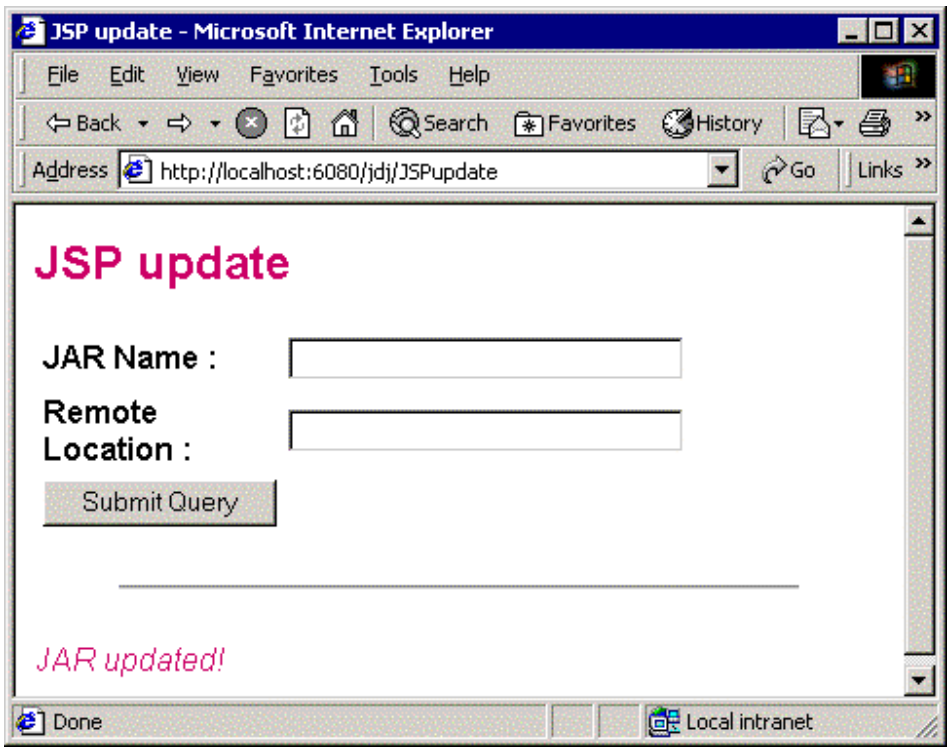


**Figure 3: JSPupdate**

You must fill the JAR Name field without extension. If you don't fill the remote location the current location is reused. It is the JAR file URL minus the file name.
Assuming you specified a jar name myjar and a remote location http://www.mydownloadsite.com, JSPserletPkg will download http://www.mydownloadsite.com/myjar.jar and persist your action in RemoteLocations with a property myjar= http://www.mydownloadsite.com.

Note the URL of JSPupdate http://localhost:6080/jdj/JSPupdate. jdj is the Context Path (name th e JSPservlet war file is deployed with). If you deploy different JSPservlet packages, you must select the appropriate JSPservlet to require an update. When you updates, the local cache is removed and the jar is always loaded from the remote location. You c an also use JSPupdate to add a jar.

Then, you will able to request a servlet with a path servletPath in myjar.jar with http://www.mysite.com/jdj/myjar/servletPath.

Let's look at the JSPupdate cod e:

```
    int index = jarName.lastIndexOf('.');
    if (index != -1)
        withoutExt = jarName.substring(0, index);
    String contextPath = request.getContextPath();
    if ((JSPservlet.JSPhandlers == null) ||
        (!JSPservlet.JSPhandlers.conta insKey(contextPath))) {
        RequestDispatcher rd = getServletContext().getRequestDispatcher(
          "/JSPservlet");
        rd.include(request, response);
    } else {
        JSPhandler handler = (JSPhandler)JSPservlet.JSPhandlers.get(contextPath);
        if (!handler.update(withoutExt, request.getParameter("remoteLocation"))) {
```

If JSPupdate is invoked before the corresponding JSPservletPkg first use, JSPupdate invokes JSPservlet in order to initialize JSPhandler properly. It uses RequestDispatcher.include to achieve this result.
JSPservlet looks if it has been called on behalf of JSPupdate. It is the reason it has to be /JSPupdate.

## *4.2 ServletUpdate*

ServletUpdate is a superset of the aforementioned JSPupdate.
Its display is split in two sections:
?? A *Definition* section at the bottom, similar to JSPupdate
?? A *loaded/defined* section at the top, that lists known and loaded archives

As JSPupdate, ServletUpdate uses GET mode. It is a servlet defined in the JSPservlet package.

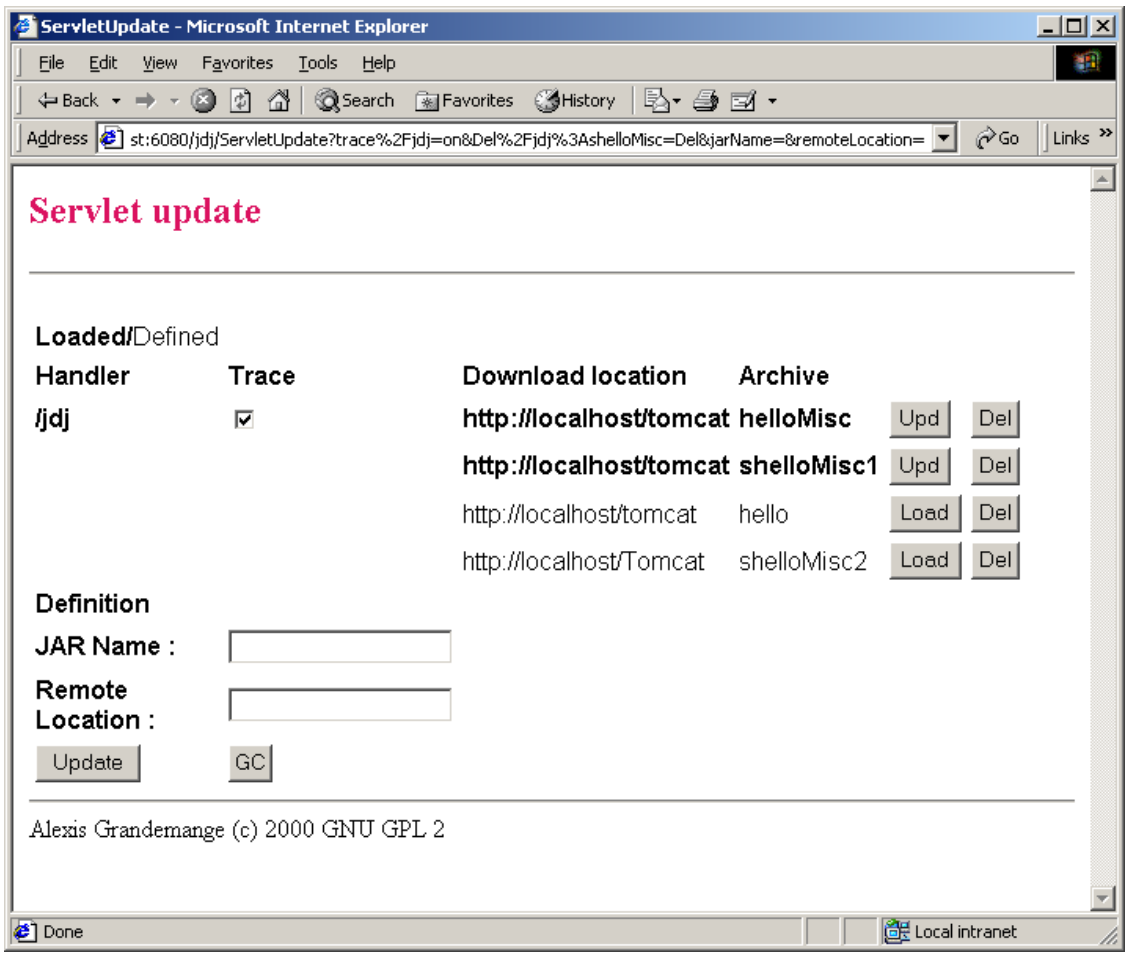


**Figure 4: ServletUpdate 1**

It supports a trace checkbox and five buttons:
?? If trace is checked, it means that trace is on.
?? Upd(ate) in the Loaded/define section means
- Download a new version of the archive in the cache
- Replace current version of classes and resources in mem ory

?? Del(ete) in the Loaded/define section means
- Remove the archive from the cache
- Remove classes and resources from memory

?? Update in the Definition section means the same thing as JSPupdate's *Submit Request*.
?? GC allows to invoke System.gc() in the Java serv er server context.

A fifth button is displayed if the application is not yet instantiated:

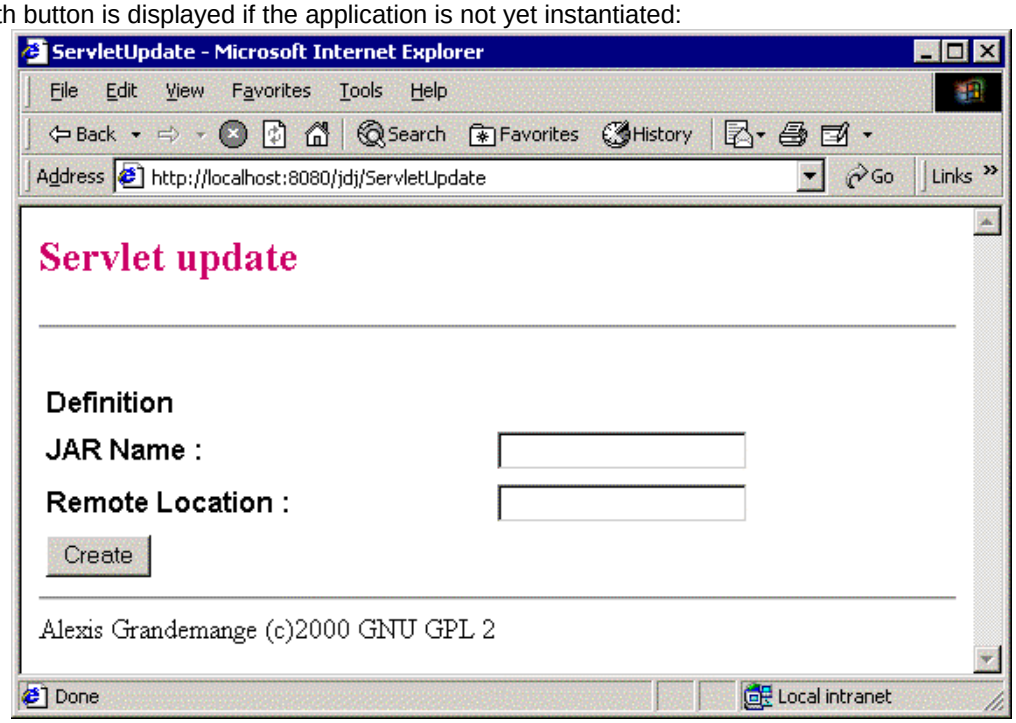


**Figure 5: ServletUpdate 2**

In this case ServletUpdate acts as JSPupdate.

Note that you can safely require an update when pages are access ed. Pending requests will hang up to the end of update for all archives controlled by a given JSPservletPkg Web Archive.

I chose a coarse grained synchronization to deliver the fastest response time in normal cases:
?? The tool is designed to run on small/me dium machines with 1or 2 processors and 128/256MB
?? Updates should be scheduled outside peak hours
?? When an update happens, it will use all available CPU and response time will be degraded anyway

It is implemented in JSPhandler.get and JSPhandler.update.

### *4.3 Non classes URL handling*

Typical case of that is image handling.
Suppose a JSP whose URL is http://www.mysite.com/jdj/myjar/myJSP references an image with a relative path images/myimage.gif. The Application Server looks for http://www.mysite.com/jdj/myjar/images/myimage.gif and therefore invokes JSPservlet.

In this case, JSPservlet looks first for images/myimage.gif in the archive. If it doesn't find it, it retrieves the download location in th e myjar's RemoteLocations property. If it finds its remote location is www.mydownloadsite.com, it tries downloading the image from www.mydownloadsite.com/images/myimage.gif. If it fails it relies on the Application Server to retrieve the image locally.

# 5 Special cases

## 5.1 RequestDispatcher

The only case where you must modify your code is when you need using a RequestDispatcher either to include or forward a request. You can use JSPservlet.getJAR() helper function.
RequestDispatcher rd = getServletContext().getRequestDispatcher(
JSPservlet.getJAR(request) + OtherServlet);
The reason is as a servlet developer you should not hardcode neither the Web Appli cation name as the standard enforces it, neither the jar name where your servlet will be deployed.

Its use is however optional:

- If the servlet or resource is in the same directory as the including servlet, I recommend using relative paths with ServletRequ est.getRequestDispatcher(). As we are using the ServletRequest, the path is relative to the current request.
- If you need to include or forward a request to another jar servlet, you have no choice but RequestDispatcher rd = getServletContext().getRequestDis patcher(OtherJarName + OtherServlet);

## 5.2 Use of SSL

JSPservlet allows using SSL to download archives using the Sun Java ™ Secure Socket Extension (JSSE).
You can download it from http://java.sun.com/products/jss e. The 1.0.2 version is exportable with strong encryption.

The explanation below is not JSPservlet related but it still can help you to configure the thing. As it can fail at your first attempt, I recommend you reading the JSSE API user guide. It explain s you how to trace the SSL connection with javax.net.debug=all and gives you a lot of information.

You need first to SSL enable the Web Server where you download the archive from. It implies generating a key pair. The server keeps the private key internal ly and asks you to submit a certificate request to the Certificate Authority of your choice.

During the SSL handshaking, jsse will check it knows the certificate chain the Web Server presents it using
<java-home>/lib/security/jssecacerts or <java -home>/lib/security/cacerts if jssecacerts doesn't exist. If your CA authority certificate is in the embedded list you have nothing to do.

### 5.2.1 cacert analysis

cacerts format is jks, which means you can use keytool to display or update its content.
To check if it is enl isted, list <jsse-home>/samples/samplecacerts – it contains the same data as cacert but with a known password *changeit* – with the command:
keytool -list -keystore *keystorepath* -storepass changeit.

You only get a summary:

```
thawtepersonalfreemailca, Fri Feb 12 21:12:16 CET 1999, trustedCertEntry,
Certificate fingerprint (MD5): 1E:74:C3:86:3C:0C:35:C5:3E:C2:7F:EF:3C:AA:3C:D9
thawtepersonalbasicca, Fri Feb 12 21:11:01 CET 1999, trustedCertEntry,
Certificate fingerprint (MD5): E6:0B:D2:C9:CA:2D:88:DB:1A:71:0E:4 B:78:EB:02:41
verisignclass3ca, Mon Jun 29 19:05:51 CEST 1998, trustedCertEntry,
Certificate fingerprint (MD5): 78:2A:02:DF:DB:2E:14:D5:A7:5F:0A:DF:B6:8E:9C:5D
thawtepersonalpremiumca, Fri Feb 12 21:13:21 CET 1999, trustedCertEntry,
Certificate fingerprint (MD5): 3A:B2:DE:22:9A:20:93:49:F9:ED:C8:D2:8A:E7:68:0D
thawteserverca, Fri Feb 12 21:14:33 CET 1999, trustedCertEntry,
```

```
Certificate fingerprint (MD5): C5:70:C4:A2:ED:53:78:0C:C8:10:53:81:64:CB:D0:1D
verisignclass4ca, Mon Jun 29 19:06:57 CEST 1998, trustedC  ertEntry,
Certificate fingerprint (MD5): 1B:D1:AD:17:8B:7F:22:13:24:F5:26:E2:5D:4E:B9:10
duke, Thu Apr 27 01:36:49 CEST 2000, trustedCertEntry,
Certificate fingerprint (MD5): 4C:CF:5E:5C:CE:69:5B:52:67:53:B7:8A:79:D3:A2:5B
verisignserverca, Mon Jun 29 19:0 7:34 CEST 1998, trustedCertEntry,
Certificate fingerprint (MD5): 74:7B:82:03:43:F0:00:9E:6B:B3:EC:47:BF:85:A5:93
verisignclass1ca, Mon Jun 29 19:06:17 CEST 1998, trustedCertEntry,
Certificate fingerprint (MD5): 51:86:E8:1F:BC:B1:C3:71:B5:18:10:DB:5F:DC:F6:  20
thawtepremiumserverca, Fri Feb 12 21:15:26 CET 1999, trustedCertEntry,
Certificate fingerprint (MD5): 06:9F:69:79:16:66:90:02:1B:8C:8C:A2:C3:07:6F:3A
verisignclass2ca, Mon Jun 29 19:06:39 CEST 1998, trustedCertEntry,
Certificate fingerprint (MD5): EC:40 :7D:2B:76:52:67:05:2C:EA:F2:3A:4F:65:F0:D8
```

If you want to know more on a given certificate you need to export it with:
keytool -export -file C:\TEMP\certfile.cer -alias verisignserverca -keystore *keystorepath* -storepass changeit

Then you can list it with:
keytool -printcert -file C:\TEMP\ certfile.cer

### 5.2.2 cacerts

If the server Certificate Authority certificate is not in the list, you have to add it.
Retrieving it depends on the browser and on the Certificate Authority. You can ask for a DER encoded or for a base 64 format.

You also need to copy <jsse-home>/samples/samplecacerts into <java-home>/lib/security/jssecacerts.

Then you can add your server certificate with:
keytool -import -file C:\TEMP \serverCA.cer -alias myCA –keystore <java-home>/lib/security/jssecacerts -storepass changeit

# 6 Tests

Tested environments:
?? Resin 1.1.3/Resin 1.2
?? Tomcat

## *6.1 Load test*

### 6.1.1 Performances

With up to 6000 servlets and JSP in a single jar file.
The performance impact of tracing is below 10%.
The sandbox performance impact is not measurable.

### 6.1.2 Loitering

No memory increase found between 6000 and 18000 invocations among 6000 different JSPs.
Test performed with and without sandbox, with signed or not archives.

### *6.2 Functional tests*

#### 6.2.1 Update

JSPupdate
ServletUpdate
Multiple archives, multiple web applications
Invalid archive name/location

Note:
servletUpdate can or not display multiple Web Application depending on the JSPhandler class loader.
If the same class loader instance is used for all JSPhandler – which is the case if you put the package on the CLASSPATH, then you display all Web applications.
Otherwise generally you are displayed only the Web application, servletUpdate belongs to.

#### 6.2.2 Security

Non-signed archive
?? without sandbox
?? with sandbox and no definition in policy file
?? with sandbox and permissions granted in policy file.

Signed archive with and without sandbox. With sandbox:
?? Valid certificate
?? Certificate not found in CA
?? Revoked certificate
?? Uncheckable archive
- CAURL not set or not accessible
- CRLURL not set or not accessible

?? Revocation when the server is running

1 policy file per web application or 1 per archive.
No permission granted to a signed archive.
Policy/certificate download

#### 6.2.3 Misc

?? Image handling
?? Beans handling
?? Servlet inheritance
?? Servlet/html include and forward
?? Caching
?? JSPservletPkg defined in CLASSPATH
?? https for download

## 7 Limitations

I checked the tool supports:
?? Servlet inheritance, case where you define a servlet as extending a base servlet.
?? Tag libs
?? JSP beans. Note however a bean is created with Beans.instantiate(). This method tries to restore the bean from a bean.ser resource and if it doesn't find it, it creates it using newInstance(). The tool searches the resource first in the archive, next in the same remote directory as the archive and finally asks the resource to the application server.

Generally speaking, it requires no change of servlets/JSP code and compilation.

# 8 Miscellaneous

## *8.1 License*

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the F ree Software Foundation; version 2 of the License. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Publ ic License for more details.

## *8.2 Deliveries*

Package name: JSPLoaderPkg.

Source files:

| | |
|---|---|
| CRLchecker.java | JNDI/LDAP code |
| JSPloader.java | class loader |
| ResourcePrivilegedAction.java | resource handling designed to be invoked by AccessController.doPrivileged (implem ents PrivilegedAction) |
| JSPloaderException.java | package exception |
| JSPhandler.java | Web application handler |
| JSPservlet.java | service servlet |
| PageBoxAPI.java | API to get the PageBox ID and log user messages |
| ServletLog.java | servlet to clear and display log |
| ServletStat.java | servlet to display stats |
| ServletUpdate.java | update servlet |
| SoapUpdate.java | update class designed to be invoked by SOAP (tested with Apache SOAP version 2) |
| JSPupdate.jsp | update JSP |

Documentation:

?? This document

?? javadoc