My First SP - Java

Please start here if you want to connect your service to the SURFconext platform

Setting up Shibboleth

Set up Apache with Shibboleth as described here.

SURFconext Metadata

Take note that the metadata and the metadata locations used for the test and production environments of SURFconext differ. Use them accordingly:

- **Test:** https://metadata.test.surfconext.nl/idp-metadata.xml
- **Production:** https://metadata.surfconext.nl/idp-metadata.xml

Setting up the Servlet

We start with a basic Servlet:

Create a file WEB-INF/src/surfnet/tutorials/mfsp/MFSP.java with the following content:

```
package surfnet.tutorials.mfsp;

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

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MFSP extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    response.setContentType("text/html");

    PrintWriter out = response.getWriter();

    out.println("<html>");
    out.println("<head>");
    out.println("<title>My First SP</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>My First SP</h1>");
    out.println("<p>Hello World!</p>");
    out.println("</body>");
    out.println("</html>");
    out.close();
}

}
```

and create WEB-INF/web.xml:

```
<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">

        <servlet>
                <servlet-name>mfsp</servlet-name>
                <servlet-class>surfnet.tutorials.mfsp.MFSP</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>mfsp</servlet-name>
                <url-pattern>/</url-pattern>
        </servlet-mapping>
</web-app>
```

Create the WEB-INF/classes directory

Compile the servlet and create a `war`-file with:

> *javac -d WEB-INF/classes/ WEB-INF/src/surfnet/tutorials/mfsp/MFSP.java*
> *jar cvf mfsp-java.war **

Don't forget to include `servlet-api.jar` in your classpath. This library is included in the Tomcat lib directory.

After deploying the `war` file into your local Tomcat root, visiting http://localhost:8080/mfsp-java should show the servlet and tell you "Hello World!".

Setup Apache as a Proxy

Next, we are going to setup as a proxy for mfsp instance that is running on Tomcat.

First, make sure Tomcat is configured with an AJP connector. Typically, it should be listening on port 8009, and the following configuration statement should be present (and not commented out) in `/etc/tomcat7/server.xml`.

```
<Connector port="8009" protocol="AJP/1.3" tomcatAuthentication="false" redirectPort="8443" />
```

Then, make sure that the Apache module `proxy` and `proxy_ajp` are enabled:

```
a2enmod proxy
a2enmod proxy_ajp
```

Add the following directions to your Apache configuration:

```
ProxyPass /mfsp-proxied ajp://localhost:8009/mfsp-java/
```

to instruct Apache to map `/mfsp-proxied` to `/mfsp-java` at the Tomcat server. Then check if the Java applet can now be reached through the regular Apache web server at `https://mfsp.gadgets.surfconext.nl/mfsp-proxied` (substitute your own host name).

If that works, enable Shibboleth authentication for this url by adding the following snippet to you Apache configuration:

```
ProxyPass /mfsp-proxied ajp://localhost:8009/mfsp-java/
<Location /mfsp-proxied>
    AuthType shibboleth
    ShibRequestSetting requireSession 1
    require valid-user
</Location>
```

As a result, you should now be required to authenticate before accessing the servlet.

Accessing SAML attributes

Even though simple authentication is now active on your servlet, it is not yet possible to access SAML attributes (such as username, email address, etc) from Java.

In Apache, the SAML attributes are stored in server variables (see xxx), and the Apache module `mod_proxy_ajp` supports passing of such server variables to the Java servlet while proxying a page. However, only variables whose names start with "`AJP_`" are passed to the servlet. Therefore, configure the prefix in the Shibboleth configuration by adding a `attributeprefix` configuration option to the main `<ApplicationDefaults>` tag:

```
        <ApplicationDefaults entityID="https://mfsp.gadgets.surfconext.nl/shibboleth"
                             attributePrefix="AJP_"
                             REMOTE_USER="eppn persistent-id targeted-id">
```

and restart `shibd`.

To show the attributes, we adjust the example servlet:

```
package surfnet.tutorials.mfsp;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;


import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MFSP extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    /*
     * Get the value of form parameter
     */
    //String name = request.getParameter("name");
    //String welcomeMessage = "Welcome "+name;
    /*
     * Set the content type(MIME Type) of the response.
     */
    response.setContentType("text/html");

    PrintWriter out = response.getWriter();
    /*
     * Write the HTML to the response
     */
    out.println("<html>");
    out.println("<head>");
    out.println("<title>My First SP</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>My First SP</h1>");
    out.println("<p>Hello World!</p>");


    /* names of the SAML attributes to display */
    String[] shib_attributes = {
        "persistent-id","Shib-user","Shib-displayName","Shib-surName",
        "Shib-commonName","Shib-givenName","Shib-eduPersonPN",
        "Shib-email","Shib-HomeOrg","Shib-uid",
        "Shib-userStatus","Shib-voName","Shib-memberOf"
    };

    out.println("<h2>SAML attributes</h2>");
    out.println("<table>");
    for (int i=0; i<shib_attributes.length; i++)
    {
        out.print("<tr><td>"+shib_attributes[i]+"</td>");
        out.print("<td>"+request.getAttribute(shib_attributes[i])+"</td></tr>\n");
    }
    out.println("</table>");

    /* also print generic attributes
     * NOTE: this will nog display the SAML attributes, because for some
     * reason these are not included in request.getAttributeNames()
```

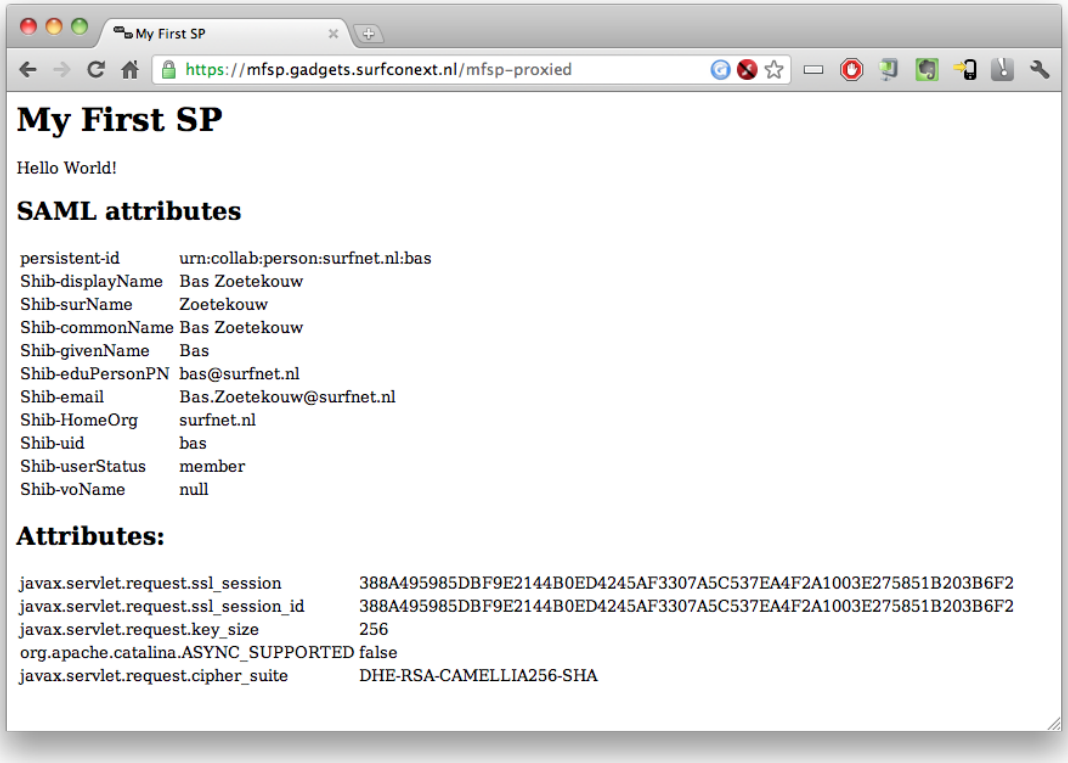
```
     */
    out.println("<h2>Attributes:</h2>");
    out.println("<table>");
    Enumeration attributes = request.getAttributeNames();
    while ( attributes.hasMoreElements() )
    {
        String attr_name = (String) attributes.nextElement();
        Object attr_val  = request.getAttribute(attr_name);

        out.print("<tr><td>"+attr_name+"</td>");
        out.print("<td>"+attr_val+"</td></tr>\n");
    }
    out.println("</table>");

}

}
```

Compile and deploy again, and your servlet should show something like this:



That's all folks!

You now have a functional example of a Java servlet connected to SURFconext. Enjoy!

Any questions and comments are welcome at support@surfconext.nl.