Getting started with Java, OpenAthens SP and the dashboard

This is a simple "Hello, world" guide to getting started as a Service Provider with OpenAthens using Java.

OpenAthens SP Java assumes you are serving your content as a JEE Web application and is implemented as two Filters:

- AtacamaBaseFilter - this filter is typically mapped to the root of your Web application (though it doesn't have to be) and handles incoming SAML responses from IdPs and adds attributes from those responses to the session context.
- AtacamaAuthFilter - this filter is mapped to your protected content path (which could be the root) and checks for the authenticated user in the session context. If it finds one the user is authenticated and the filter chain proceeds. If the user is not yet authenticated this filter will forward the user to the IdP so they can sign in.

Optionally a service provider may implement a third filter for *authorization* of the user to check the user, their organization or group is permitted to use the requested resource.

The filters should be in the chain in the following order: AtacamaBaseFilter -> AtacamaAuthFilter -> MyAuthZFilter.

Before you begin you will need:

- An OpenAthens customer domain and access to the OpenAthens admin area (https://admin.openathens.net)
- An OpenAthens personal account under that customer domain for testing
- The EntityID for your doman - this can be found in the OpenAthens admin area under ***Management > Connections***

What are we going to do?

For this example we'll first register our application with the Publisher Dashboard and then build a simple Spring Boot application that has a public facing home page and a protected content area served from /subscribers. We'll not go into the details of Spring Boot here, but provided you're familiar with Java and Maven you should be able to follow along with this guide.

Create the application in the OpenAthens publisher dashboard

Go to https://sp.openathens.net, sign in and follow these steps:

- Click the register new application button and choose OpenAthens SP in the dialogue box
- Name your application. Eventually this will be customer facing, but for now it can be anything
- Application URL: this is the root web address of the application - e.g: https://sp.yourdomain.com
- Leave 'users in my domain' ticked, but keep the other options unticked
- Click the create button. This creates the application record and a connection

At this point you're presented with a getting started guide in the dashboard that should provide you with enough to implement OpenAthens SP on your JEE Web application.

Make a note of the configuration URL and access key from the web.xml - we'll use these later.

Hello world

First up you'll need to create a Maven project in your IDE of choice and it'll need the following POM:

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.springframework</groupId>
    <artifactId>gs-serving-web-content</artifactId>
    <version>0.1.0</version>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.4.0.RELEASE</version>
    </parent>

    <repositories>
        <repository>
            <id>eduserv</id>
            <url>https://repo.openathens.net/maven</url>
        </repository>
    </repositories>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <optional>true</optional>
        </dependency>

        <!-- OpenAthens SP -->
        <dependency>
            <groupId>uk.org.eduserv.iam</groupId>
            <artifactId>openathens-sp</artifactId>
            <version>2.1.1</version>
            <type>pom</type>
        </dependency>
    </dependencies>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

This is the standard Spring Boot starter POM with the OpenAthens SP repository and dependency added.

Spring Boot uses MVC so we'll create the controller next:

Sample Controller

```
@Controller
public class PageController {

    @Autowired
    private HttpSession session;

    @RequestMapping("/")
    public String homePage() {
        return "homepage";
    }

    @RequestMapping("/subscribers")
    public String subscriptionPage(Model model) {
        Map<String, String> oaAttrs = new HashMap<>();
        Enumeration<String> names = session.getAttributeNames();
        while (names.hasMoreElements()) {
            String name = names.nextElement();
            oaAttrs.put(name, session.getAttribute(name).toString());
        }
        model.addAttribute("username", session.getAttribute("oa_username"));
        model.addAttribute("oa_attrs", oaAttrs);
        return "subscriptionContent";
    }
}
```

And next the views which here are Thymeleaf templates and should be put in src/main/resources/templates:

Public home page template

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Simple SP</title>
</head>
<body>
    <h1>Simple SP</h1>
    <p>
        This is a public page. You might want to visit the <a
            href="/subscribers">subscription content</a>.
    </p>
    <hr />
</body>
</html>
```

and:

Sample protected page that echos attributes

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<title>Simple SP - Subscription Content</title>
</head>
<body>
    <h1>Simple SP - Subscription Content</h1>
    <p th:text="'You are signed in as ' + ${username}" />
    <hr />
    <ul>
        <li th:each="attr : ${oa_attrs}"><b><span th:text="${attr.key}">
                key </span></b> -&gt; <span th:text="${attr.value}"> value </span></li>
    </ul>
</body>
</html>
```

Tie it all together with the self-contained application that also includes the configuration (which mirrors the web.xml example provided when you create the application). You'll need to set your access key and config URI that you saved earlier (find them on the application configuration if you mislaid that info)

Application and Configuration

```
@Configuration
@SpringBootApplication
public class Application {
    public static final String ACCESS_KEY_PARAM = "OA_ACCESS_KEY";
    public static final String ACCESS_KEY = ""; // put your access key here
    public static final String CONFIG_URI_PARAM = "OA_CONFIG_URI";
    public static final String CONFIG_URI = ""; // put your configuration URL here

    private static final int FIRST = 1;
    private static final int SECOND = 2;

    public static void main(String[] args) {
        // You might need to set these...
        // System.setProperty("atacama.keystore.password", "atacama");
        // System.setProperty("atacama.keystore.alias", "1:atacama");

        SpringApplication.run(Application.class, args);
    }

    @Bean
    public ServletContextInitializer initializer() {
        return new ServletContextInitializer() {
            @Override
            public void onStartup(ServletContext servletContext) throws ServletException {
                servletContext.setInitParameter(ACCESS_KEY_PARAM, ACCESS_KEY);
                servletContext.setInitParameter(CONFIG_URI_PARAM, CONFIG_URI);
            }
        };
    }

    @Bean
    public FilterRegistrationBean registerBaseFilter() {
        FilterRegistrationBean frb = new FilterRegistrationBean();
        frb.setFilter(new AtacamaBaseFilter());
        frb.addUrlPatterns("/*");
        frb.setOrder(FIRST);
        return frb;
    }

    @Bean
    public FilterRegistrationBean registerAtacamaAuthFilter() {
        FilterRegistrationBean frb = new FilterRegistrationBean();
        frb.setFilter(new AtacamaAuthFilter());
        frb.addUrlPatterns("/subscribers/*");
        frb.setOrder(SECOND);
        return frb;
    }
}
```

Who goes there?

Next we associate the new Application with an identity provider. We will use your OpenAthens customer domain as identity provider for testing and you'll need the Entity ID of the Identity Provider. This can be found in the admin area under Management -> Connections.

Armed with your entity ID open the OpenAthens publisher dashboard and do the following:

- Click Applications
- Select your newly created application
- Open the Configuration tab
- Under "Discovery Method" select "Use default identity provider" and in the text box below put the Entity ID obtained from the Admin Area
- Click "Save Changes"
- Finally restart your webserver to get it to reload the configuration

Almost there

There is one more thing we need to do and that is add the metadata signing certificate to the classpath. The "Getting Started Guide" on the dashboard outlines how to create the key pair. Keystore Explorer is a useful tool to create a keystore pair with an empty keystore password and entry password.

This is quite simple:

- Download and open Keystore Explorer
- Select "Create a new KeyStore"
- Select JKS and then click OK
- From the Tools menu select Generate Key Pair (or press CTRL+G)
- Choose RSA with a keysize of 2048 and click OK
- Click the book icon next to Name field and fill in Common Name (CN) with your application name and Organization Name (O) with your application host name
- Click OK and then OK
- Set the alias to "1" without the quotes
- When prompted for a new password just click OK
- Hopefully you'll see a message saying creation successful
- Select File -> Save
- When prompted for password just click OK
- Save as `atacamaKeystore.jks`
- Double click on the newly created key pair
- Click the PEM button bottom right
- Copy the PEM including the BEGIN/END CERTIFICATE lines

Back on the Publisher Dashboard:

- Open the Connection associated with your new application
- Under "SAML Connector" there is a section for certificates
- Click "Add a certificate"
- Paste the PEM into the box and click "Save Changes"

The keystore must be called `atacamaKeystore.jks` and be on the classpath for your application. For this sample application copy `atacamaKeystore.jks` to `src/main/resources`.

The final set of files should look like this:

```
.
 pom.xml
 src
     main
         java
            uk
                org
                    eduserv
                        iam
                            simplesp
                                Application.java
                                PageController.java
                                SubscriberPageController.java
         resources
             atacamaKeystore.jks
             templates
                 homepage.html
                 subscriptionContent.html
```

With all that done, run your application class and you'll hopefully have a working OpenAthens SP install.

After installing the certificate, you may need to restart your webserver again.

What next?

At the moment your service provider is only connected to your customer domain. For very small applications that may be all you need, but it is more likely you'll want to join the federation as soon as possible and for that you will need to get production ready.