Tutorial

Introduction

There's a good getting started guide (http://www.hueniverse.com/hueniverse/2007/10/beginners-gui-1.html) that illustrates OAuth 1.0 by describing two different (but related) services. One is a photo-sharing application. The other is a photo-printing application. In OAuth terms, the photo sharing application is the OAuth *provider* and the photo printing application is the OAuth *consumer* or *client*.

For this tutorial, we will see OAuth for Spring Security in action by deploying a photo-sharing application and a photo-printing application on our local machine. We'll name the photo-sharing application "Sparklr" and the photo-printing application "Tonr". A user named "Marissa" (who has an account at both Sparkr and Tonr) will use Tonr to access her photos on Sparklr without ever giving Tonr her credentials to Sparklr.

There is a Sparklr application for both OAuth 1.0 and for OAuth 2.0, likewise Tonr. The best way to run them is to clone or download the repo on github (https://github.com/spring-projects/spring-security-oauth) and run from source code See the samples/README.md (https://github.com/spring-projects/spring-security-oauth/tree/master/samples) for detailed instructions.

OAuth 1.0	OAuth 2.0
Sparklr 1	Sparklr 2
Tonr 1	Tonr 2

Each application is a standard Maven (http://maven.apache.org/) project, so you will need Maven installed. Each application is also a Spring MVC application with Spring Security integrated. If you are familiar with Spring and Spring Security, the configuration files will look familiar to you (the OAuth2 samples use a single application context whereas many MVC applications use a root context and a child for the DispatcherServlet).

Setup

Checkout the Sparklr and Tonr applications, and take a look around. Note especially the Spring configuration files in `src/main/webapp/WEB-INF`.

For Sparklr, you'll notice the definition of the OAuth provider mechanism and the consumer/client details along with the standard spring security configuration (http://docs.spring.io/spring-security/site/docs/4.0.x/reference/html/ns-config.html) elements. For Tonr, you'll notice the definition of the OAuth consumer/client mechanism and the resource details. For more information about the necessary components of an OAuth provider and consumer, see the developers guide (devguide.html).

You'll also notice the Spring Security filter chain in `applicationContext.xml` and how it's configured for OAuth support.

Deploy Sparklr

```
mvn install
cd samples/oauth(2)/sparklr
mvn tomcat7:run
```

Sparklr should be started on port 8080. Go ahead and browse to http://localhost:8080/sparklr (http://localhost:8080/sparklr). Note the basic login page and the page that can be used to browse Marissa's photos. Logout to ensure Marissa's session is no longer valid. (Of course, the logout isn't mandatory; an active Sparklr session will simply bypass the step that prompts for Marissa's credentials before confirming authorization for Marissa's protected resources.)

Start Tonr.

Shutdown sparklr (it will be launched in the same container when tonr runs), then

```
mvn install
cd samples/oauth(2)/tonr
mvn tomcat7:run
```

Tonr should be started on port 8080. Browse to http://localhost:8080/tonr(2) (http://localhost:8080/tonr). Note Tonr's home page has a '2' on the end if it is the oauth2 version.

Observe…

Now that you've got both applications deployed, you're ready to observe OAuth in action.

1. Login to Tonr.

   Marissa's credentials are already hardcoded into the login form.

2. Click to view Marissa's Sparklr photos.

   You will be redirected to the Sparklr site where you will be prompted for Marissa's credentials.

3. Login to Sparklr.

   Upon successful login, you will be prompted with a confirmation screen to authorize access to Tonr for Marissa's pictures.

4. Click "authorize".

   Upon authorization, you should be redirected back to Tonr where Marissa's Sparklr photos are displayed (presumably to be printed).