Single Sign On (SSO) Example with JSON Web Token (JWT) and Spring Boot

Last modified @ 25 February 2020

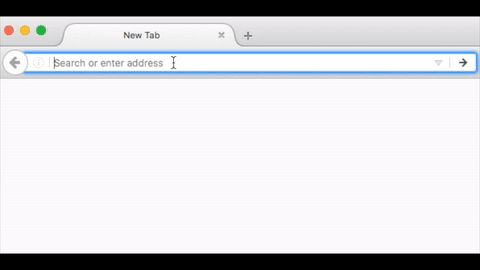
[Spring Boot](https://hellokoding.com/tag/spring-boot/)[Security with Spring](https://hellokoding.com/tag/security/)

This tutorial will walk you through the steps of creating a Single Sign On (SSO) Example with JSON Web Token (JWT) and Spring Boot

What you'll build

You'll build 3 separated services:

* 1 Authentication Service: will be deployed at localhost:8080.
* 2 Resource Services (to simplify, we use the same code base): will be deployed at localhost:8180 and localhost:8280.



What you'll need

* JDK 1.7+
* Maven 3+

Stack

* Java
* Single Sign On
* JSON Web Token
* Spring Boot
* Freemarker

Build Authentication Service

Project structure

├── src

│ └── main

│ ├── java

│ │ └── com

│ │ └── hellokoding

│ │ └── sso

│ │ └── auth

│ │ ├── CookieUtil.java

│ │ ├── JwtUtil.java

│ │ ├── LoginController.java

│ │ └── WebApplication.java

│ └── resources

│ └── templates

│ └── login.ftl

└── pom.xml

Copy

Project dependencies

[pom.xml](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/auth-service/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>com.hellokoding.security</groupId>

<artifactId>single-sign-on.auth-service</artifactId>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.1.4.RELEASE</version>

</parent>

<properties>

<java.version>1.7</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-freemarker</artifactId>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Copy

CookieUtil

JWT Token'll be saved and extracted from browser cookies.

[CookieUtil.java](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/auth-service/src/main/java/com/hellokoding/sso/auth/CookieUtil.java)

package com.hellokoding.sso.auth;

import org.springframework.web.util.WebUtils;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class CookieUtil {

public static void create(HttpServletResponse httpServletResponse, String name, String value, Boolean secure, Integer maxAge, String domain) {

Cookie cookie = new Cookie(name, value);

cookie.setSecure(secure);

cookie.setHttpOnly(true);

cookie.setMaxAge(maxAge);

cookie.setDomain(domain);

cookie.setPath("/");

httpServletResponse.addCookie(cookie);

}

public static void clear(HttpServletResponse httpServletResponse, String name) {

Cookie cookie = new Cookie(name, null);

cookie.setPath("/");

cookie.setHttpOnly(true);

cookie.setMaxAge(0);

httpServletResponse.addCookie(cookie);

}

public static String getValue(HttpServletRequest httpServletRequest, String name) {

Cookie cookie = WebUtils.getCookie(httpServletRequest, name);

return cookie != null ? cookie.getValue() : null;

}

}

Copy

cookie.setSecure(secure): secure=true => work on HTTPS only.

cookie.setHttpOnly(true): invisible to JavaScript.

cookie.setMaxAge(maxAge): maxAge=0: expire cookie now, maxAge<0: expire cookiie on browser exit.

cookie.setDomain(domain): visible to domain only.

cookie.setPath("/"): visible to all paths.

JwtUtil

We use [JJWT](https://github.com/jwtk/jjwt) to generate/parse JWT Token.

[JwtUtil.java](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/auth-service/src/main/java/com/hellokoding/sso/auth/JwtUtil.java)

package com.hellokoding.sso.auth;

import io.jsonwebtoken.JwtBuilder;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import javax.servlet.http.HttpServletRequest;

import java.util.Date;

public class JwtUtil {

public static String generateToken(String signingKey, String subject) {

long nowMillis = System.currentTimeMillis();

Date now = new Date(nowMillis);

JwtBuilder builder = Jwts.builder()

.setSubject(subject)

.setIssuedAt(now)

.signWith(SignatureAlgorithm.HS256, signingKey);

return builder.compact();

}

public static String getSubject(HttpServletRequest httpServletRequest, String jwtTokenCookieName, String signingKey){

String token = CookieUtil.getValue(httpServletRequest, jwtTokenCookieName);

if(token == null) return null;

return Jwts.parser().setSigningKey(signingKey).parseClaimsJws(token).getBody().getSubject();

}

}

Copy

LoginController

[LoginController.java](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/auth-service/src/main/java/com/hellokoding/sso/auth/LoginController.java)

package com.hellokoding.sso.auth;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import javax.servlet.http.HttpServletResponse;

import java.util.HashMap;

import java.util.Map;

@Controller

public class LoginController {

private static final String jwtTokenCookieName = "JWT-TOKEN";

private static final String signingKey = "signingKey";

private static final Map<String, String> credentials = new HashMap<>();

public LoginController() {

credentials.put("hellokoding", "hellokoding");

credentials.put("hellosso", "hellosso");

}

@RequestMapping("/")

public String home(){

return "redirect:/login";

}

@RequestMapping("/login")

public String login(){

return "login";

}

@RequestMapping(value = "login", method = RequestMethod.POST)

public String login(HttpServletResponse httpServletResponse, String username, String password, String redirect, Model model){

if (username == null || !credentials.containsKey(username) || !credentials.get(username).equals(password)){

model.addAttribute("error", "Invalid username or password!");

return "login";

}

String token = JwtUtil.generateToken(signingKey, username);

CookieUtil.create(httpServletResponse, jwtTokenCookieName, token, false, -1, "localhost");

return "redirect:" + redirect;

}

}

Copy

To simplify, we use a HashMap (credentials) as user database.

View Template

[login.ftl](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/auth-service/src/main/resources/templates/login.ftl)

<!DOCTYPE html>

<html lang="en">

<head>

<title>Authentication Service</title>

</head>

<body>

<form method="POST" action="/login?redirect=${RequestParameters.redirect!}">

<h2>Log in</h2>

<input name="username" type="text" placeholder="Username"

autofocus="true"/>

<input name="password" type="password" placeholder="Password"/>

<div>(try username=hellokoding and password=hellokoding)</div>

<div style="color: red">${error!}</div>

<br/>

<button type="submit">Log In</button>

</form>

</body>

</html>

Copy

Application Configuration

[WebApplication.java](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/auth-service/src/main/java/com/hellokoding/sso/auth/WebApplication.java)

package com.hellokoding.sso.auth;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class WebApplication{

public static void main(String[] args) throws Exception {

SpringApplication.run(WebApplication.class, args);

}

}

Copy

Run

mvn clean spring-boot:run

Build Resource Service

Project structure

├── src

│ └── main

│ ├── java

│ │ └── com

│ │ └── hellokoding

│ │ └── sso

│ │ └── resource

│ │ ├── CookieUtil.java

│ │ ├── JwtFilter.java

│ │ ├── JwtUtil.java

│ │ ├── ResourceController.java

│ │ └── WebApplication.java

│ └── resources

│ ├── templates

│ │ └── protected-resource.ftl

│ └── application.properties

└── pom.xml

Copy

Project dependencies

[pom.xml](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/resources-service/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>com.hellokoding.security</groupId>

<artifactId>single-sign-on.resources-service</artifactId>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.1.4.RELEASE</version>

</parent>

<properties>

<java.version>1.7</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-freemarker</artifactId>

</dependency>

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Copy

JwtFilter

JwtFilter enforces SSO. If JWT Token's not existed (unauthenticated), redirects to Authentication Service. If JWT Token's existed (authenticated), extracts user identity and forwards the request.

[JwtFilter.java](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/resources-service/src/main/java/com/hellokoding/sso/resource/JwtFilter.java)

package com.hellokoding.sso.resource;

import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

public class JwtFilter extends OncePerRequestFilter {

private static final String jwtTokenCookieName = "JWT-TOKEN";

private static final String signingKey = "signingKey";

@Override

protected void doFilterInternal(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, FilterChain filterChain) throws ServletException, IOException {

String username = JwtUtil.getSubject(httpServletRequest, jwtTokenCookieName, signingKey);

if(username == null){

String authService = this.getFilterConfig().getInitParameter("services.auth");

httpServletResponse.sendRedirect(authService + "?redirect=" + httpServletRequest.getRequestURL());

} else{

httpServletRequest.setAttribute("username", username);

filterChain.doFilter(httpServletRequest, httpServletResponse);

}

}

}

Copy

ResourceController

[ResourceController.java](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/resources-service/src/main/java/com/hellokoding/sso/resource/ResourceController.java)

package com.hellokoding.sso.resource;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import javax.servlet.http.HttpServletResponse;

@Controller

public class ResourceController {

private static final String jwtTokenCookieName = "JWT-TOKEN";

@RequestMapping("/")

public String home() {

return "redirect:/protected-resource";

}

@RequestMapping("/protected-resource")

public String protectedResource() {

return "protected-resource";

}

@RequestMapping("/logout")

public String logout(HttpServletResponse httpServletResponse) {

CookieUtil.clear(httpServletResponse, jwtTokenCookieName);

return "redirect:/";

}

}

Copy

View Template

[protected-resource.ftl](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/resources-service/src/main/resources/templates/protected-resource.ftl)

<!DOCTYPE html>

<html lang="en">

<head>

<title>Protected Resource Service</title>

</head>

<body>

<h2>Hello, ${Request.username!}</h2>

<a href="/logout">Logout</a>

</body>

</html>

Copy

Application Configuration

[application.properties](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/resources-service/src/main/resources/application.properties)

services.auth=http://localhost:8080/login

Copy

[WebApplication.java](https://github.com/hellokoding/hellokoding-courses/blob/master/springboot-examples/springboot-single-sign-on-jwt/resources-service/src/main/java/com/hellokoding/sso/resource/WebApplication.java)

package com.hellokoding.sso.resource;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.web.servlet.FilterRegistrationBean;

import org.springframework.context.annotation.Bean;

import java.util.Collections;

@SpringBootApplication

public class WebApplication {

@Value("${services.auth}")

private String authService;

@Bean

public FilterRegistrationBean jwtFilter() {

final FilterRegistrationBean registrationBean = new FilterRegistrationBean();

registrationBean.setFilter(new JwtFilter());

registrationBean.setInitParameters(Collections.singletonMap("services.auth", authService));

registrationBean.addUrlPatterns("/protected-resource");

return registrationBean;

}

public static void main(String[] args) throws Exception {

SpringApplication.run(WebApplication.class, args);

}

}

Copy

Run

Resource Service 1

mvn clean spring-boot:run -Dserver.port=8180

Resource Service 2

mvn clean spring-boot:run -Dserver.port=8280