**Single Sign On (SSO) Example with JSON Web Token (JWT), Cookie, Spring Boot**

**Prerequisite**

* JDK 1.7+
* Maven 3+

**Stack**

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

**Projects**

Create the below services (projects) in maven

1. Authentication Service
2. Resource Service 1
3. Resource Service 2
4. **Authentication Service**

**Project Structure**

├── src

│   └── main

│   ├── java

│   │   └── com

│   │   └── hellokoding

│   │   └── sso

│   │   └── auth

│   │   ├── CookieUtil.java

│   │   ├── JwtUtil.java

│   │   ├── LoginController.java

│   │   └── WebApplication.java

│   ├── resources

│   │   └── application.properties

│   └── webapp

│   └── login.ftl

└── pom.xml

**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>*  *<artifactId>hello-sso-jwt-auth</artifactId>*  *<name>hello-sso-jwt-auth</name>*  *<description>hello-sso-jwt-auth</description>*  *<parent>*  *<groupId>org.springframework.boot</groupId>*  *<artifactId>spring-boot-starter-parent</artifactId>*  *<version>1.4.1.RELEASE</version>*  *</parent>*  *<properties>*  *<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>*  *<java.version>1.8</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>0.6.0</version>*  *</dependency>*    *</dependencies>*    *<build>*  *<plugins>*  *<plugin>*  *<groupId>org.springframework.boot</groupId>*  *<artifactId>spring-boot-maven-plugin</artifactId>*  *</plugin>*  *</plugins>*  *</build>*  *</project>* |

**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;  }  } |

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.

**CookieUtil.java**

We use JJWT to generate/parse JWT Token.

|  |
| --- |
| 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();  }  } |

**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;  }  } |

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

**Login.ftl ( View Template )**

|  |
| --- |
| <!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> |

**application.properties**

|  |
| --- |
| spring.freemarker.template-loader-path: /  spring.freemarker.suffix: .ftl  server.port=8082 |

**WebApplication.java**

|  |
| --- |
| package com.hellokoding.sso.auth;  import org.springframework.boot.SpringApplication;  import org.springframework.boot.autoconfigure.SpringBootApplication;  import org.springframework.boot.builder.SpringApplicationBuilder;  import org.springframework.boot.context.web.SpringBootServletInitializer;  @SpringBootApplication  public class WebApplication extends SpringBootServletInitializer {  @Override  protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {  return application.sources(WebApplication.class);  }  public static void main(String[] args) throws Exception {  SpringApplication.run(WebApplication.class, args);  }  } |

1. **Resource Service 1**

**Project Structure**

── src

│   └── main

│   ├── java

│   │   └── com

│   │   └── hellokoding

│   │   └── sso

│   │   └── resource

│   │   ├── CookieUtil.java

│   │   ├── JwtFilter.java

│   │   ├── JwtUtil.java

│   │   ├── ResourceController.java

│   │   └── WebApplication.java

│   ├── resources

│   │   └── application.properties

│   └── webapp

│   └── protected-resource.ftl

└── pom.xml

**Source**

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>  <artifactId>hello-sso-jwt-resource1</artifactId>  <name>hello-sso-jwt-resource1</name>  <description>hello-sso-jwt-resource1</description>  <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.4.1.RELEASE</version>  </parent>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <java.version>1.8</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>0.6.0</version>  </dependency>    </dependencies>    <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins>  </build>  </project> |

JwtFilter.java

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.

|  |
| --- |
| package com.hellokoding.sso.resource;  import org.springframework.stereotype.Component;  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;  @Component  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);  }  }  } |

JwtUtil.java

|  |
| --- |
| package com.hellokoding.sso.resource;  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();  }  } |

CookieUtil.java

|  |
| --- |
| package com.hellokoding.sso.resource;  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);  cookie.setDomain("localhost");  httpServletResponse.addCookie(cookie);  }  public static String getValue(HttpServletRequest httpServletRequest, String name) {  Cookie cookie = WebUtils.getCookie(httpServletRequest, name);  return cookie != null ? cookie.getValue() : null;  }  } |

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:/";  }  } |

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.builder.SpringApplicationBuilder;  import org.springframework.boot.context.embedded.FilterRegistrationBean;  import org.springframework.boot.context.web.SpringBootServletInitializer;  import org.springframework.context.annotation.Bean;  import java.util.Collections;  @SpringBootApplication  public class WebApplication extends SpringBootServletInitializer {  @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;  }  @Override  protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {  return application.sources(WebApplication.class);  }  public static void main(String[] args) throws Exception {  SpringApplication.run(WebApplication.class, args);  }  } |

Protected-resource.flt (ViewTemplate)

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <title>Protected Resource Service</title>  </head>  <body>  <h2>Hello, ${Request.username!}</h2>  <h2>Resource 11111111111</h2>  <a href="/logout">Logout</a>  </body>  </html> |

application.properties

|  |
| --- |
| spring.freemarker.template-loader-path: /  spring.freemarker.suffix: .ftl  server.port=8083  services.auth=http://localhost:8082/login |

1. **Resource Service 2**

Replicate the “resource service” project and change application.properties file.

application.properties

|  |
| --- |
| spring.freemarker.template-loader-path: /  spring.freemarker.suffix: .ftl  server.port=8084  services.auth=http://localhost:8082/login |

**Build and run the application**

Navigate to the project where pom file is located and use the below command

mvn clean spring-boot:run

D:\microservices-demo-ws\sso-jwt-spring-boot\hello-sso-jwt-auth>mvn clean spring-boot:run

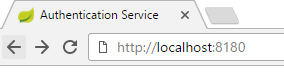
D:\microservices-demo-ws\sso-jwt-spring-boot\hello-sso-jwt-resource1>mvn clean spring-boot:run

D:\microservices-demo-ws\sso-jwt-spring-boot\hello-sso-jwt-resource2>mvn clean spring-boot:run

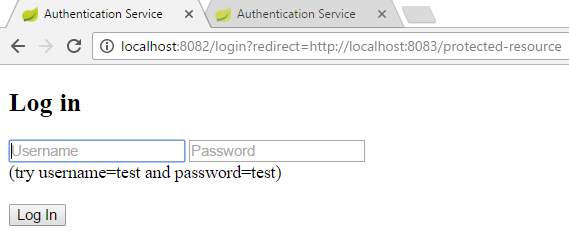
**Test and run the application**

<http://localhost:8083/>

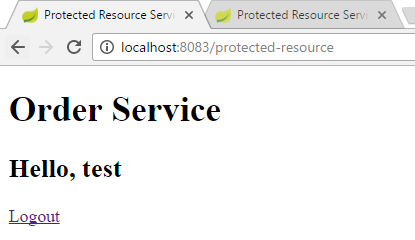
<http://localhost:8084/>



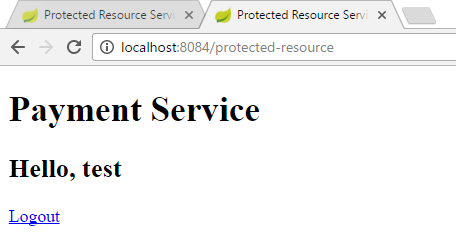
It will redirect to the login screen.



Enter the user id and password and click log in, It will redirect to the appropriate url



Now try second resource, It will login without asking credentials



Now click “logout” and test the other resource.

**References**

<https://hellokoding.com/hello-single-sign-on-sso-with-json-web-token-jwt-spring-boot/>

<https://github.com/jwtk/jjwt>

<https://jwt.io/>