Auth Security SSO JWT Cookie Java Spring Boot FreeMarker

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

Aug 01, 2016. By Giau Ngo @giaunv

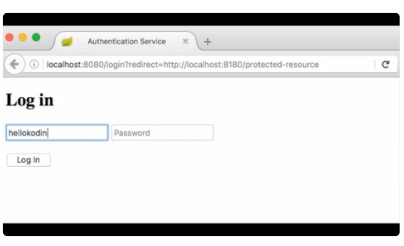
This post walks you through the process of creating the Single Sign On (SSO) Example with JSON Web Token (JWT), Spring Boot.

```
Hello Spring Boot Series:
- Spring Boot Hello World Example with FreeMarker
- Spring Boot Hello World Example with Thymeleaf
- Spring Boot Hello World Example with JSP
```

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

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
```

Project dependencies

[redacted]

CookieUtil

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

[redacted]

`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 to generate/parse JWT Token.

[redacted]

LoginController

[redacted]

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

View Template

[redacted]

Application Configuration

[redacted]

[redacted]

Run

```
mvn clean spring-boot:run
```

Resource Service

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
```

Project dependencies

[redacted]

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.

[redacted]

ResourceController

[redacted]

View Template

[redacted]

Application Configuration

[redacted]

[redacted]

Run

Resource Service 1

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

Resource Service 2

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

Source code

`git@github.com:hellokoding/hello-sso-jwt.git`
https://github.com/hellokoding/hello-sso-jwt

If you have found the post useful, please consider making a donation on paypal, following hellokoding.com on github / facebook / twitter / google+, and sharing this post.

share

Latest on hellokoding.com

Hello Single Sign On (SSO) Example with JSON Web Token (JWT), Cookie, Spring Boot
JPA Many-To-Many Extra Columns Relationship Mapping Example with Spring Boot
JPA Many-To-Many Relationship Mapping Example with Spring Boot
JPA One-To-Many Relationship Mapping Example with Spring Boot
JPA One-To-One Shared Primary Key Relationship Mapping Example with Spring Boot