互联网应用开发技术

Web Application Development

第12课
WEB后端-SPRING SECURITY SAMPLES

Episode Twelve

Spring Security Samples

陈昊鹏

chen-hp@sjtu.edu.cn

Login Sample

- MvcConfig.java

```
@Configuration
public class MvcConfig implements WebMvcConfigurer {

  public void addViewControllers(ViewControllerRegistry registry) {
    registry.addViewController("/home").setViewName("home");
    registry.addViewController("/").setViewName("home");
    registry.addViewController("/hello").setViewName("hello");
    registry.addViewController("/login").setViewName("login");
  }

}
```

Login Sample

- WebSecurityConfigureAdapter.java

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
  @Override
  protected void configure(HttpSecurity http) throws Exception {
    http
      .authorizeRequests()
        .antMatchers("/", "/home").permitAll()
        .anyRequest().authenticated()
        .and()
      .formLogin()
        .loginPage("/login")
        .permitAll()
        .and()
      .logout()
        .permitAll();
  }
```

Login Sample

- WebSecurityConfigureAdapter.java

```
@Bean
@Override
public UserDetailsService userDetailsService() {
  UserDetails user =
     User.withDefaultPasswordEncoder()
      .username("user")
      .password("password")
      .roles("USER")
      .build();

  return new InMemoryUserDetailsManager(user);
}
}
```

Login Sample

- home.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
  <head>
    <title>Spring Security Example</title>
  </head>
  <body>
    <h1>Welcome!</h1>

    <p>Click <a th:href="@{/hello}">here</a> to see a greeting.</p>
  </body>
</html>
```

Login Sample

- hello.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="https://www.thymeleaf.org"
      xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
    <head>
        <title>Hello World!</title>
    </head>
    <body>
        <h1 th:inline="text">Hello [[${#httpServletRequest.remoteUser}]]!</h1>
        <form th:action="@{/logout}" method="post">
            <input type="submit" value="Sign Out"/>
        </form>
    </body>
</html>
```

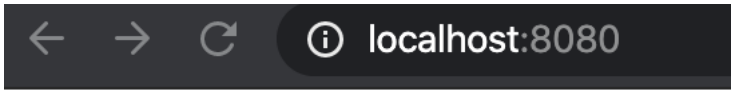
Login Sample

- login.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="https://www.thymeleaf.org"
    xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity3">
  <head>
    <title>Spring Security Example </title>
  </head>
  <body>
    <div th:if="${param.error}">
      Invalid username and password.
    </div>
    <div th:if="${param.logout}">
      You have been logged out.
    </div>
    <form th:action="@{/login}" method="post">
      <div><label> User Name : <input type="text" name="username"/> </label></div>
      <div><label> Password: <input type="password" name="password"/> </label></div>
      <div><input type="submit" value="Sign In"/></div>
    </form>
  </body>
</html>
```

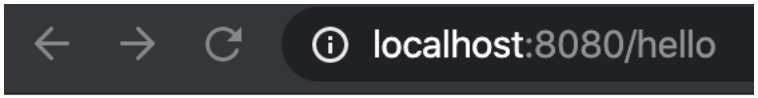
Login Sample

REliable, INtelligent & Scalable Systems



Welcome!

Click here to see a greeting.



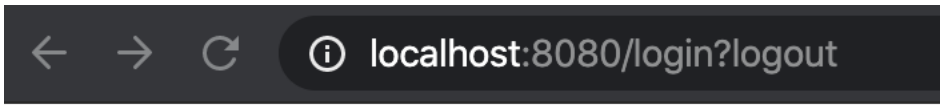
Hello user!

Sign Out

User Name :

Password:

Sign In



You have been logged out.

User Name :

Password:

Sign In

Invalid username and password.

User Name :

Password:

Sign In

React + Spring Security: Front-end

- App.js

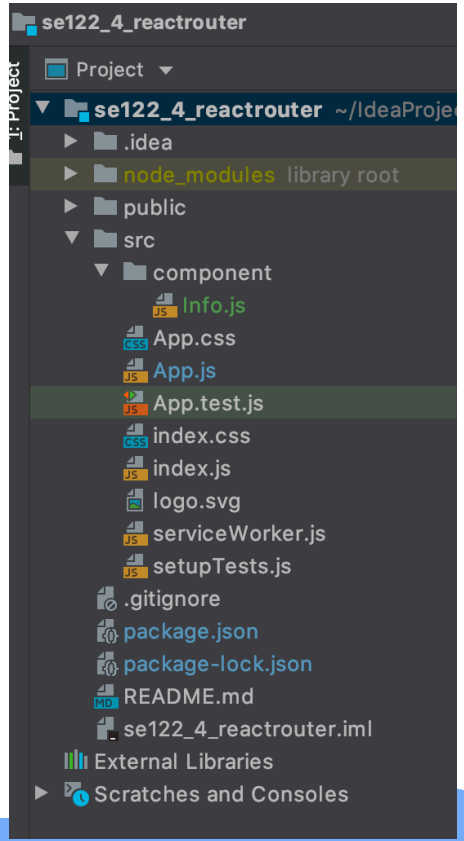
```
import React from 'react';
import {BrowserRouter as Router, Switch, Route, Link} from "react-router-dom";
import Info from './component/Info';
function App() {
  return (
    <Router>
      <div>
        <nav>
          <ul>
            <li><Link to="/">Home</Link></li>
            <li><Link to="/about">About</Link></li>
            <li><Link to="/users">Users</Link></li>
          </ul>
        </nav>
        <Switch>
          <Route path="/about"><Info menu="about"/></Route>
          <Route path="/users"><Info menu="users"/></Route>
          <Route path="/"><Info menu=""/></Route>
        </Switch>
      </div>
    </Router>
  );
}
export default App;
```

React + Spring Security: Front-end

- Info.is

```
import React from 'react';
function Info(props) {
   let url = 'http://localhost:8080/' + props.menu;
   let username = 'root';
   let password = '123';
   let headers = new Headers();
   headers.set('Authorization', 'Basic ' + Buffer.from(username + ":" + password).toString('base64'));

   fetch(url, {
      method: 'GET',
      headers: headers,
      credentials: 'include'
   }).then(response => response.text())
      .then(data => {
         document.getElementById("info").innerText = data
      }).catch(function (ex) {
      console.log('parsing failed', ex)
   })
   return (
      <div>
         <h1 id="info">Welcome</h1>
      </div>
   );
}
export default Info;
```



React + Spring Security: Back-end

- SpringSecurityApplication.java

```
@SpringBootApplication(exclude= {DataSourceAutoConfiguration.class})
public class SpringSecurityApplication {
    public static void main(String[] args) {
        SpringApplication.run(SpringSecurityApplication.class, args);
    }
}
```

- application.properties

```
spring.security.user.name=root
spring.security.user.password=123
```

React + Spring Security: Back-end

- GreetingController.java

```
@CrossOrigin(maxAge = 3600)
@RestController
public class GreetingController {
  @GetMapping("/about")
  public String getAbout() {
    return "This is a Spring security sample";
  }

  @GetMapping("/users")
  public String getUser() {
    return "I am a user";
  }

  @GetMapping("/")
  public String getHome() {
    return "Let' start!";
  }
}
```

React + Spring Security: Back-end

- SecurityConfig.java

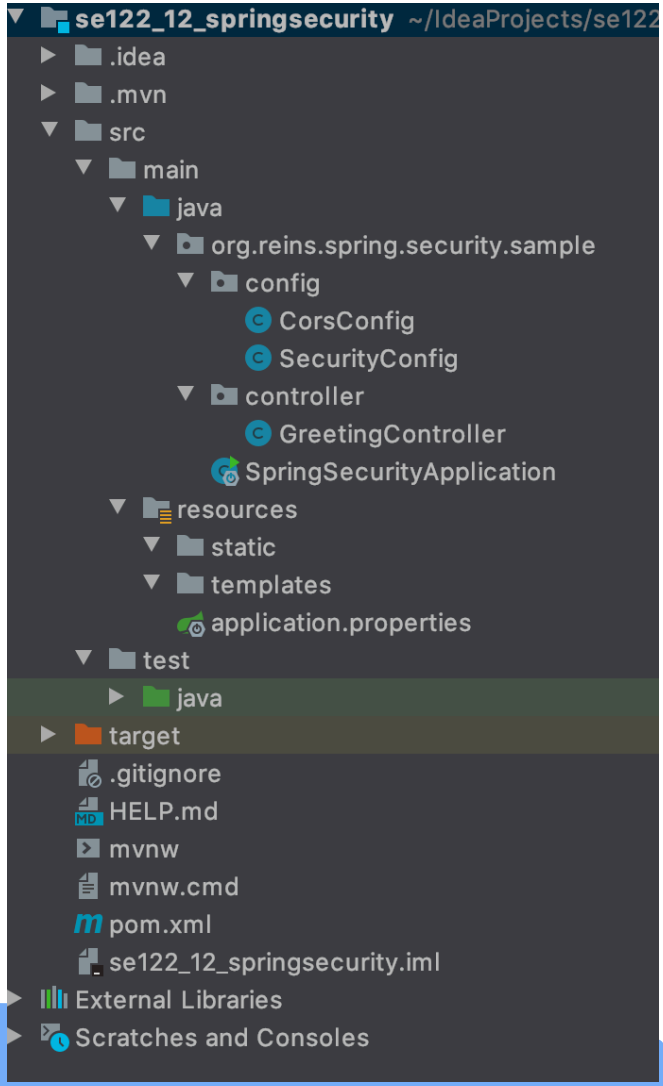
```
@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {
   @Override
   protected void configure(HttpSecurity http) throws Exception {
      http.cors();
   }
   @Bean
   public CorsConfigurationSource corsConfigurationSource() {
      final CorsConfiguration configuration = new CorsConfiguration();
      configuration.setAllowedOrigins(ImmutableList.of("*"));
      configuration.setAllowedMethods(ImmutableList.of("HEAD", "GET", "POST", "PUT", "DELETE", "PATCH"));
      configuration.setAllowCredentials(true);
      configuration.setAllowedHeaders(ImmutableList.of("Authorization", "Cache-Control", "Content-Type"));
      final UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
      source.registerCorsConfiguration("/**", configuration);
      return source;
   }
}
```

React + Spring Security: Back-end

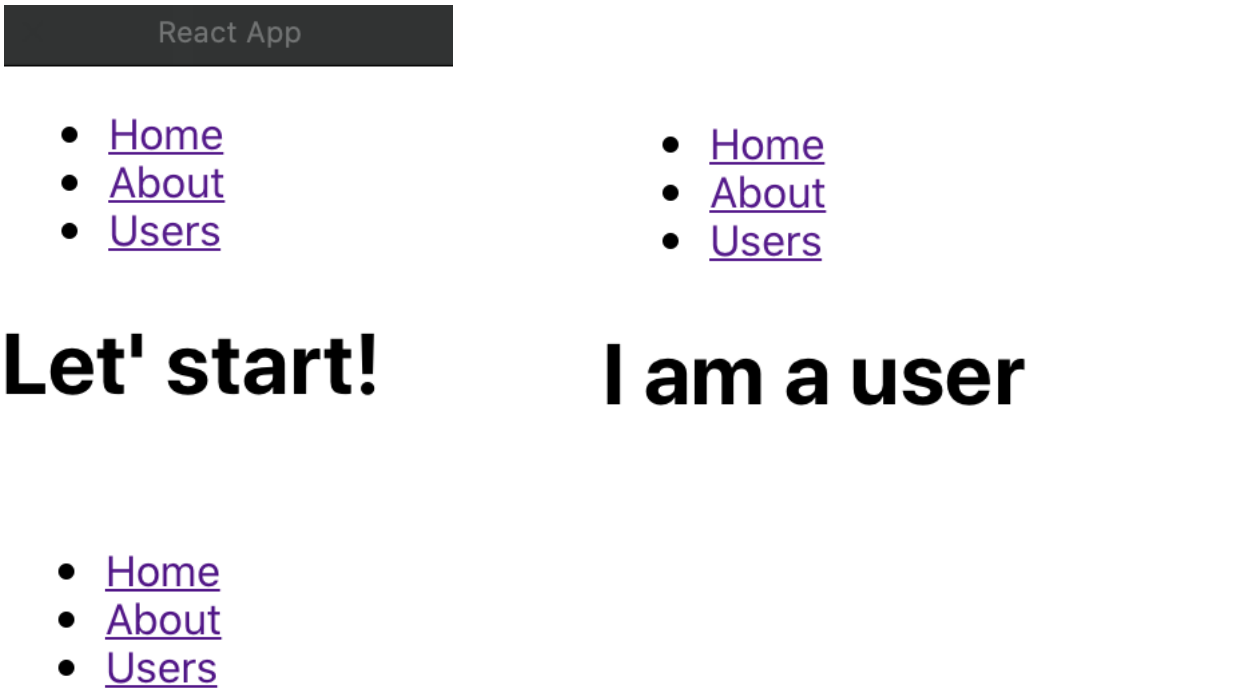
- CorsConfig.java

```
@Configuration
public class CorsConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
                .allowedOrigins("*")
                .allowedMethods("*")
                .allowedHeaders("*")
                .exposedHeaders(HttpHeaders.SET_COOKIE)
                .allowCredentials(true).maxAge(1800);
    }
}
```



Run the application



References

- 如何在Spring启动时在Spring Security级别启用CORS(How to enable CORS at Spring Security level in Spring boot)
  - http://www.it1352.com/978249.html
- Securing a Web Application
  - https://spring.io/guides/gs/securing-web/

- *Web*开发技术
- *Web Application Development*

Thank You!