Spring Security

- Create a dynamic web project **'SecurityApp'** in STS. When you create the project select the create web.xml deployment descriptor option.
- Add the necessary Spring MVC jars in WebContent/WEB-INF/lib folder.
- Configure the web.xml with the DispatcherServlet, ContextLoaderListener in the web.xml like this. We'call DispatcherServlet as **ds**.
- Create two spring bean configuration files; WEB-INF/applicationContext.xml and WEB-INF/ds-servlet.xml.
- Create a Controller 'SimpleController' like this.

```
@Controller
class SimpleController{
        @RequestMapping("/")
        @ResponseBody
        public String index(){
          return "Welcome";
        }
}
```

- Configure <mvc:annotation-driven> and <context:component-scan> elements in ds-servlet.xml
- Execute the SimpleController, index method in the browser

- Extract Spring Security jars.
- Add the following Spring security jar files in WEB-INF/lib folder.

spring-security-config-3.1.4.RELEASE.jar
spring-security-core-3.1.4.RELEASE.jar
spring-security-web-3.1.4.RELEASE.jar

- Configure a filter **DelegatingFilterProxy** provided by Spring Security filter in web.xml as given below. This filter will intercept the requests to the application and delegate it to the security filter beans that you'll configure in applicationContext.xml.

```
        <filter>
                <filter-name>springSecurityFilterChain</filter-name>
                <filter-
class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
        </filter>
        <filter-mapping>
                <filter-name>springSecurityFilterChain</filter-name>
                <url-pattern>/*</url-pattern>
        </filter-mapping>
```

- In application-context.xml add the security namespace
- Add the first basic Http configuration in applicationContext.xml

```
<security:http auto-config="true">
```

```
            <security:intercept-url pattern="/**" access="ROLE_USER,ROLE_ADMIN"/>
        </security:http>
        <security:authentication-manager>
            <security:authentication-provider>
                <security:user-service>
                    <security:user name="sam" password="sam123"
authorities="ROLE_ADMIN"/>
                    <security:user name="kim" password="kim123"
authorities="ROLE_USER"/>
                </security:user-service>
            </security:authentication-provider>
        </security:authentication-manager>
```

- Restart your server and access the SimpleController. You should see a login form. Provide the necessary credentials as specified in **<security:user-service>** element in applicationContext.xml.
- Modify the <security:http> element and add the **`<security:http-basic/>`** element inside it.
- Restart your server and access the SimpleController. You should see a login form as a windows authentication box.
- You can comment the **`<security:http-basic/>`** `element now.`
- Modify the index() method of the SimpleController as shown here.

```
@RequestMapping("/")
public String index(Principal principal,Model model){
      model.addAttribute("message","Hello " + principal.getName() +
"!!!");
            return "index";
}
```

- Create index.jsp in /WEB-INF/pages folder. If you don't have pages folder create it. Configure the InternalResourceViewResolver bean in ds-servlet.xml
- The contents of index.jsp are shown here.

```
<html>
      <body>
            <h1>${message}</h1>
      </body>
</html>
```

- Let's configure our application to read the credentials from a database.
- Create a database 'SecurityDB' in mysql. Execute the following query to create the users table.

```
create table users(
  id int(5) not null primary key auto_increment,
username varchar(40),
password varchar(50),
authority varchar(50)
);

insert into users(username,password,authority) values("sam","sam123","ROLE_ADMIN");
```

insert into users(username,password,authority) values("kim","kim123","ROLE_ADMIN");

- Modify applicationContext.xml. Comment **<security:user-service>** element and add the following in its place.

```
<security:jdbc-user-service
        data-source-ref="dataSource"
        users-by-username-query="select username,password,true from users
                                    where username=?"
        authorities-by-username-query="select username,authority from users
                                    where username=?"
/>
```

- Configure the dataSource in applicationContext.xml.

```
<bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
                <property name="driverClassName" value="com.mysql.jdbc.Driver"></property>
                <property name="url" value="jdbc:mysql://localhost:3306/securitydb"></property>
                <property name="username" value=" DB USERNAME"></property>
                <property name="password" value="DB PASSWORD "></property>
</bean>
```

- Execute the application. Make sure the login form works fine by using the database now.
- Let's construct a custom login form for the application. Modify the SimpleController by adding a login() method as shown here.

```
        @RequestMapping("/login")
        public String login(){
              return "login";
        }
```

- Create login.jsp in /WEB-INF/pages folder. Add the following <form> element in the page.

```
<form method="POST" action="j_spring_security_check">
        User name <input type="text" name="j_username"><br/>
        Password <input type="password" name="j_password"><br/>
        <input type="submit" value="Login">
</form>
```

- Modify the applicationContext.xml file. Replace the existing **<security:http auto-config="true"> …</security:http>** element with the following.

```
<security:http auto-config="true">
            <security:intercept-url pattern="/**"
                  access="ROLE_USER,ROLE_ADMIN"/>
            <security:form-login login-page="/login" />
</security:http>
```

- Modify the applicationContext.xml file. Add the following element at the top of the file.

```
<security:http pattern="/login" security="none"></security:http>
```

- Execute the application to see the custom login page.
- Modify index.jsp to have a logout link. Add the following <form> element anywhere in the page that will generate a logout link.

```
<form action="j_spring_security_logout">
                 <input type="submit" value="Logout">
</form>
```

- Execute the application to see the logout link in the index.jsp page. Clicking on logout should take you back to the login page.