1、课程名称：SpringSecurity 安全访问

2、具体内容

所有的 Rest 服务最终都是暴露在公网上的，也就是说如果你的 Rest 服务属于一些你自己公司的私人业务，这样的结果会直接导致你信息的泄漏，所以对于 Rest 访问，安全性是首要的因素。

2.1、配置安全验证

如果要想进行安全的验证处理，那么首先一定要先在服务的提供方上进行处理。

1、 【microcloud-provider-dept-8001】修改 pom.xml 配置文件，追加 SpringSecurity 相关依赖包引入：

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
```

如果你现在配置了安全框架，则在启动时会出现有如下的一个提示信息：

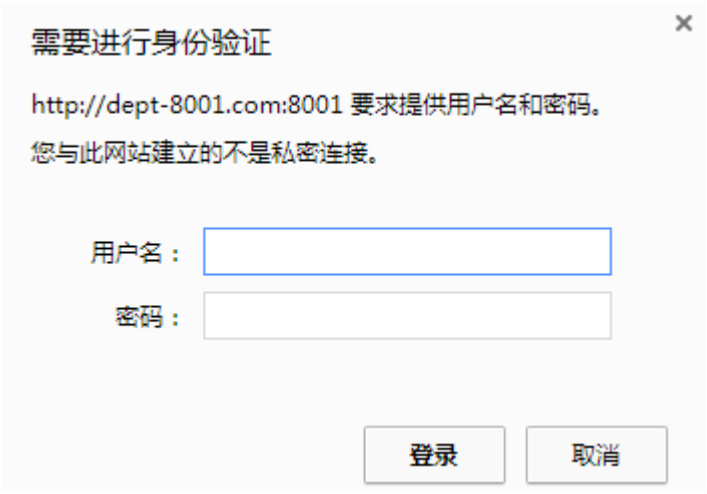
```
Using default security password: 73f5d975-0cfc-46e7-b7cc-65fbb0b67447
```

2、 【microcloud-provider-dept-8001】修改 application.yml 配置文件，进行安全的用户名配置：

```
security:
```

```
basic:
  enabled: true # 启用SpringSecurity的安全配置项
user:
  name: mldnjava  # 认证用户名
  password: hello # 认证密码
  role:       # 授权角色
  - USER
```

随后在项目之中访问 Rest 服务接口：http://dept-8001.com:8001/dept/list，此时在访问的时候会直接询问用户要求用户输入用户名以及密码。



这个时候有一种更简化的方法进行内容的输入：http://**mldnjava:hello@**dept-8001.com:8001/dept/list；

2.2、服务消费端处理

在实际的开发之中，对于 Rest 服务提供者是不可能被用户直接进行访问的，于是肯定需要有一个 Rest 客户端（WEB 端、SpringBoot）进行调用，可是现在 Rest 提供者的服务上有了认证信息，那么该如何访问呢？

```
public static final String DEPT_GET_URL = "http://mldnjava:hello@dept-8001.com:8001/dept/get/";
```

如果这个时候在 Rest 客户端上直接使用用户名和密码做加密处理，那么根本就无法进行访问，此时会出现有 401 的错误代码，因为认证出现了错误。之所以无法访问，是因为所有的认证的处理操作，应该以头信息的模式来进行处理。而后要使用 Base64 进行加密处理后才可以得到一个正确的访问路径。

1、 【microcloud-consumer-80】修改 RestConfig 配置类，在这个配置类上追加有新的 Bean 配置项：

```
package cn.mldn.microcloud.config;
import java.nio.charset.Charset;
import java.util.Base64;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpHeaders;
import org.springframework.web.client.RestTemplate;
@Configuration
public class RestConfig {
    @Bean
    public HttpHeaders getHeaders() { // 要进行一个Http头信息配置
```

```
        HttpHeaders headers = new HttpHeaders(); // 定义一个HTTP的头信息
        String auth = "mldnjava:hello"; // 认证的原始信息
        byte[] encodedAuth = Base64.getEncoder()
                .encode(auth.getBytes(Charset.forName("US-ASCII"))); // 进行一个加密的处理
        // 在进行授权的头信息内容配置的时候加密的信息一定要与“Basic”之间有一个空格
        String authHeader = "Basic " + new String(encodedAuth);
        headers.set("Authorization", authHeader);
        return headers;
    }
    @Bean
    public RestTemplate getRestTemplate() {
        return new RestTemplate();
    }
}
```

2、 【microcloud-consumer-80】修改 ConsumerDeptController 配置类，在进行 Rest 访问的时候设置好这个头部的信息

```
package cn.mldn.microcloud.controller;
import java.util.List;
import javax.annotation.Resource;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;
import cn.mldn.vo.Dept;
@RestController
public class ConsumerDeptController {
    public static final String DEPT_GET_URL = "http://dept-8001.com:8001/dept/get/";
    public static final String DEPT_LIST_URL = "http://dept-8001.com:8001/dept/list/";
    public static final String DEPT_ADD_URL = "http://dept-8001.com:8001/dept/add?dname=";
    @Resource
    private RestTemplate restTemplate;
    @Resource
    private HttpHeaders headers;
    @RequestMapping(value = "/consumer/dept/get")
    public Object getDept(long id) {
        Dept dept = this.restTemplate
                .exchange(DEPT_GET_URL + id, HttpMethod.GET,
                        new HttpEntity<Object>(this.headers), Dept.class)
                .getBody();
        return dept;
    }
    @SuppressWarnings("unchecked")
    @RequestMapping(value = "/consumer/dept/list")
```

```
    public Object listDept() {
        List<Dept> allDepts = this.restTemplate
                .exchange(DEPT_LIST_URL, HttpMethod.GET,
                        new HttpEntity<Object>(this.headers), List.class)
                .getBody();
        return allDepts;
    }
    @RequestMapping(value = "/consumer/dept/add")
    public Object addDept(Dept dept) throws Exception {
        Boolean flag = this.restTemplate.exchange(DEPT_ADD_URL, HttpMethod.POST,
                new HttpEntity<Object>(dept, this.headers), Boolean.class)
                .getBody();
        return flag;
    }
}
```

对于 Rest 而言，实际上在 Spring 课程之中就讲解过，里面如果要进行参数的传递有各种方式，例如：各种页面的路径信息组成。如果要是传递复杂内容，建议你在整个处理的时候就去使用那些页面的参数传递的模式。

3、无状态 Session 配置

通过之前一系列的演示可以发现整个 Rest 项目中的一个问题所在，所有的 Rest 都是基于 HTTP 协议的一种应用，而在这种应用上，所有的 WEB 容器一般都会提供有一个 Session 的机制，也就是说每一个用户访问之后如果该用户一直连接，则认为该用户应该一直被服务器保存状态，但是微服务有可能同时并发访问几十万人，那么如果所有的 Session 状态都被维护着？

1、 【microcloud-provider-member-8001】现在修改 Rest 程序类，追加一个取得 session id 的方法：

```
    @RequestMapping("/dept/sessionId")
    public Object id(HttpServletRequest request) {
        return request.getSession().getId() ;
    }
```

随后进行提供者的 Rest 连接访问：http://mldnjava:hello@dept-8001.com:8001/dept/sessionId；

2、 在有一些的 SpringCloud 的配置之中，默认是会保存有 Session 状态的，而后如果用户有需要则可以根据“SessionCreationPolicy”枚举类进行不同的 session 状态设置，但是从整体的操作来说，session 最好设置为无状态。

- 以下为保持 Session 状态（服务器内存有可能被占满）：

```
security:
  sessions: always
```

- 以下为无状态的 Session 设置（服务器不保存 Session 状态，每一次连接都是一个新的用户）：

```
security:
  sessions: stateless
```

不管你以后的项目或者支持类中是否有设置无状态的问题，你最好都进行一下设置，否则你的 Rest 服务将受到严重的内存困扰，最严重的问题就是内存溢出。

4、定义安全配置程序类

在进行 Rest 服务开发的时候，为了保证安全所有的程序里面都需要进行 Spring-Security 安全认证处理，可是之前所进行的认证处理都是在 application.yml 配置文件完成的，这样的配置明显是非常不合乎逻辑的，因为如果此时你要开发的微服务很多，并且这些微服务都要求使用统一的用户名和密码的时候就非常不方便了。所以现在最简单的做法是进行统一的设置。

1、 创建一个 microcloud-security 的 Maven 模块；

2、 【microcloud-security】修改 pom.xml 配置文件：

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>springloaded</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
        </dependency>
    </dependencies>
```

3、 【microcloud-security】建立一个统一的安全配置类：

```
package cn.mldn.microcloud.config;
import javax.annotation.Resource;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.config.http.SessionCreationPolicy;
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    @Resource
    public void configGlobal(AuthenticationManagerBuilder auth)
            throws Exception {
        auth.inMemoryAuthentication().withUser("mldnjava").password("hello")
                .roles("USER").and().withUser("admin").password("hello")
```

```
                .roles("USER", "ADMIN");
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        // 表示所有的访问都必须进行认证处理后才可以正常进行
        http.httpBasic().and().authorizeRequests().anyRequest()
                .fullyAuthenticated();
        // 所有的Rest服务一定要设置为无状态，以提升操作性能
        http.sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS);
    }
}
```

4、 【microcloud-provider-member-8001】修改 pom.xml 配置文件，引入安全配置模块：

```
        <dependency>
            <groupId>cn.mldn</groupId>
            <artifactId>microcloud-security</artifactId>
        </dependency>
```

5、 【microcloud-provider-member-8001】删除掉 application.yml 中与安全有关的配置项（以下内容删除）：

```
security:
  sessions: stateless
  basic:
    enabled: true # 启用SpringSecurity的安全配置项
  user:
    name: mldnjava  # 认证用户名
    password: hello # 认证密码
    role:       # 授权角色
    - USER
```

由于现在所写的安全处理类是在程序启动类的子包之中，应该可以自动扫描到。

6、 访问地址：http://mldnjava:hello@dept-8001.com:8001/dept/sessionId

3、总结

Rest 服务的重要配置：

1、 安全访问；

2、 Session 控制。