

# 15-2 自訂登入頁面

## Principal機制



## 當事者Principal需要

- 一個實作個別身分UserDetailsService介面的類別
- 填入相對角色

## 實作一個被驗證的當事者類別

## 實作介面 AuthenticatedPrincipal

```
1 package com.khh.controller;
2
3 import org.springframework.security.core.userdetails.User;
4
5 @Service
6 public class DemoUserDetailsService implements UserDetailsService {
7     private String userName;
8     private String password;
9     private String[] roles;
10
11     //自訂建構子
12     public DemoUserDetailsService() {
13     }
14
15     public String getUsername() {
16         return userName;
17     }
18
19     public void setUsername(String userName) {
20         this.userName = userName;
21     }
22 }
23
24 package com.khh.controller;
25
26 import org.springframework.security.core.AuthenticatedPrincipal;
27
28 public class AppPrincipal implements AuthenticatedPrincipal {
29     private String name;
30     public AppPrincipal(String name) {
31         this.name = name;
32     }
33     @Override
34     public String getName() {
35         // TODO Auto-generated method stub
36         return this.name;
37     }
38 }
```

注入封裝使用者資訊的UserDetailsService物件

轉型為自訂的Service類型，重新封裝登入頁面輸入的使用者名稱與密碼等

設定角色群組

```
@Service
public class DemoUserDetailsService implements UserDetailsService {
    private String userName;
    private String password;
    private String[] roles;
```

```
@Controller
public class MemberController {
    //Attribute 注入相對使用者
    @Autowired
    private UserDetailsService userDetails;
```

```
//內部驗證使用者登入與授權設定
private boolean isValid(String username,String password)
{
    boolean isOk=false;
    //客製化附加資訊 注入使用者名稱與密碼至UserDetailsService中
    ((DemoUserDetailsService)this.userDetails).setUserName(username);
    ((DemoUserDetailsService)this.userDetails).setPassword(password);

    //設定角色(採用固定 一般配合到資料表存取)
    ((DemoUserDetailsService)this.userDetails).setRoles(new String[] {"IT","ADMIN"});
    System.out.println("相對使用者:"+userDetails.toString());
```

Token需要三個成員進行封裝

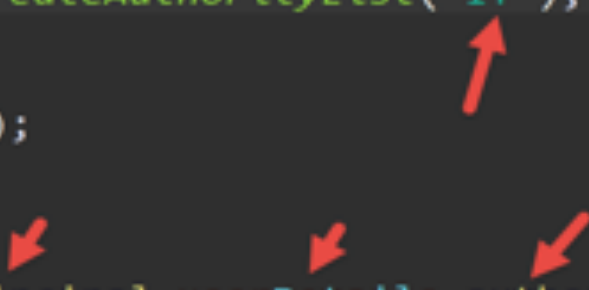
UserDetailsService

被授權的當事者AuthenticatedPrincipal

以及授權書 GrantedAuthority

```
//-----手動進行 Authority授權識別 使用工廠模式AuthorityUtils
List<GrantedAuthority> authorities=AuthorityUtils.createAuthorityList("IT");

//建構當事者
AppPrincipal myPrincipal=new AppPrincipal("AppUser");
//建立Authentiy Collection
//第一個參數是Principal 第二個身分 username/password
UsernamePasswordAuthenticationToken token
    = new UsernamePasswordAuthenticationToken(myPrincipal,userDetails,authorities);
```



使用SecurityContextHolder.getContext方法問出相對的SecurityContext

注入Token

```
//第一個參數是Principal 第二個身分 username/password
UsernamePasswordAuthenticationToken token
    = new UsernamePasswordAuthenticationToken(myPrincipal,userDetails,authorities);

SecurityContext sc = SecurityContextHolder.getContext();
//角色群取設定
//貫穿你個人Context 設定憑證
sc.setAuthentication(token);

//是否被驗證
if(token.isAuthenticated()) {
    System.out.println("驗證狀態:被驗證!!");
    isOk=true;
}

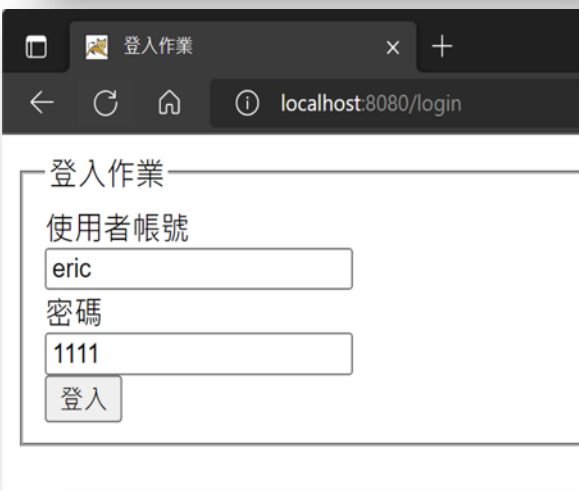
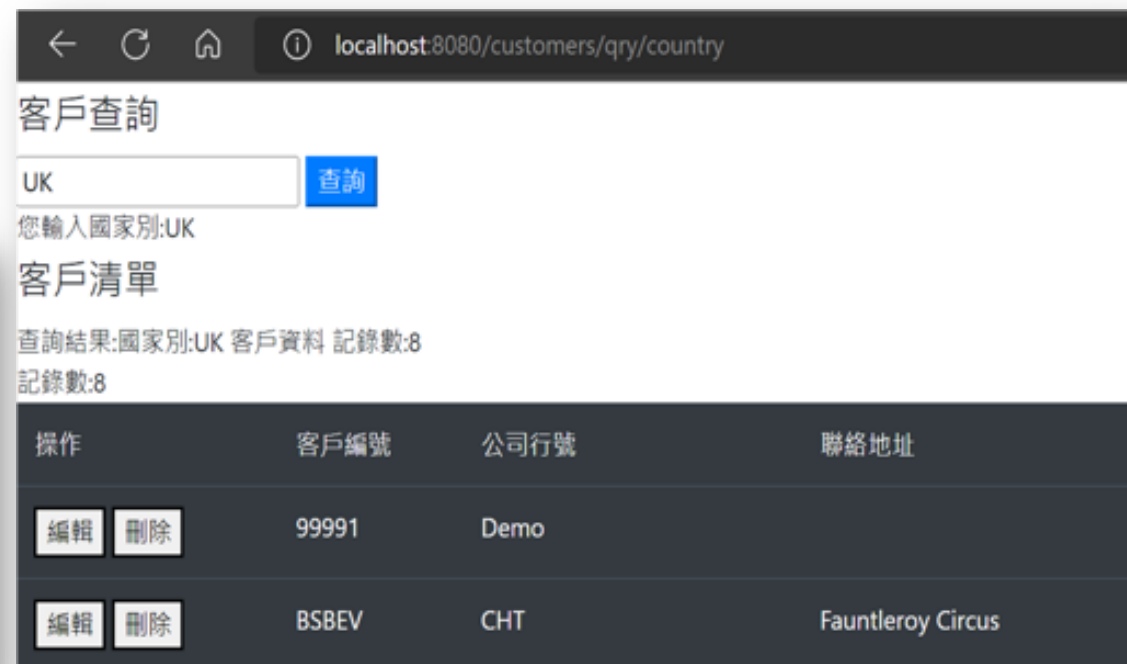
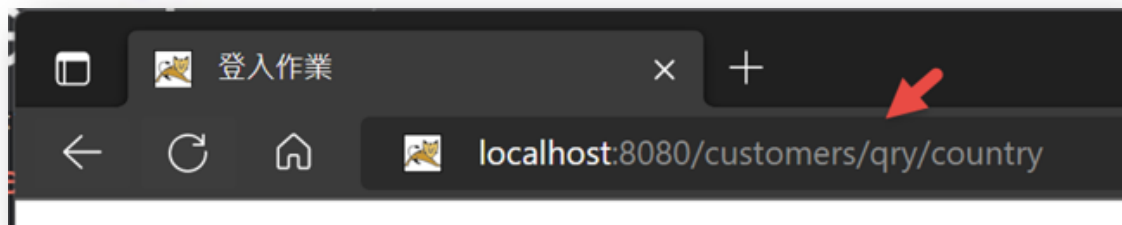
return isOk;
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>登入作業</title>
</head>
<body>
  <fieldset>
    <legend>登入作業</legend>
    <form method="post" th:action="@{/login}">
      <div>使用者帳號</div>
      <input name="username" type="text"/>
      <div>密碼</div>
      <input name="password" type="text"/>
      <div>
        <input type="submit" value="登入"/>
      </div>
    </form>
  </fieldset>
  <h3 th:text="${message}"></h3>
</body>
</html>
```

```
//登入頁面
@RequestMapping(path="/login",method= {RequestMethod.GET,RequestMethod.POST})
public String loginForm(String username,String password,Model model,
    HttpServletRequest request) {
    //判斷是否為第一次請求 或者是傳送驗證
    String message=null;

    if(request.getMethod().equals("POST")) {
        //進行驗證
        boolean result=memberRep.existsById(username);
        if(result==false) {
            //非法帳號
            message="登入驗證失敗!!!";
            model.addAttribute("message",message);
        }else {
            //進行憑證處理
            boolean tokenValid=userValid(username,password);
            message="登入驗證成功!!!";
            model.addAttribute("message",message);
        }
    }
}
```





```
2022-09-17 07:17:15.307 INFO 16476 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
Hibernate: select count(*) as col_0_0_ from member memberenti0_ where memberenti0_.username=?
相對使用者:com.khh.controller.DemoUserDetailsService@8091d80
驗證狀態:被驗證!!
```



## 總結：15-2 自訂登入頁面Principal機制

學習到如何自訂Login頁面驗證程序之後，接下來我們來看看設定Spring Filter機制。

