7-3 傳遞參數到頁面運算

Spring MVC Request狀態持續架構

MVC設計模式，Action程序產生的狀態

- 如前端傳遞進來的參數
- 控制程序中產生的狀態，如查詢資結果等

調用頁面當中的Http Request與Response生命週期的延伸

狀態隱含與明確參照持續應用

表單欄位參數定義與傳遞

表單欄位需要使用name attribute定義識別名稱

與Controller-Action參數名稱對應

採用自訂Model(JavaBean)注入與封裝表單欄位

- 表單欄位需與JavaBean Property(setter/getter)名稱應對
- 自動封裝表單欄位至JavaBean物件屬性中

```
<body>
    <fieldset>
        <legend>訂單查詢</legend>
        <form>
            <div>訂單編號</div>
            <input type="text" th:placeholder="${message}" name="orderId"/>
            <input type="submit" value="查詢"/>
        </form>
    </fieldset>
    <div>輸入的訂單編號:<label th:text="${orderId}"></label></div>
</body>
```

```
//訂單查詢作業-依照訂單編號
@RequestMapping(path="/orders/qry",
        method= {RequestMethod.GET,RequestMethod.POST})
public String ordersQry(String orderId,Model model) {
    if(orderId!=null) {
        //TODO 進行查詢作業
        System.out.println("輸入的訂單編號:"+orderId);
        model.addAttribute("orderId",orderId);
    }
    //進行狀態資料模組持續
    model.addAttribute("message","請輸入訂單編號");
    return "ordersqry";
}
```

表單欄位傳遞透過注入的JavaBean自動封裝屬性

```
//前端傳遞進來的註冊資訊
//表單欄位form field name=xxxx 對應注入進來的JavaBean 透過屬性Property自動封裝進來
@PostMapping(path= {"/member/register"})
public String memberSave2(Model model,MemberEntity entity) {
    System.out.println(mydatasource);
    System.out.println("新增資料:"+entity.getUserName());
```

```
<form method="post" action="save">
    <div>使用者帳號</div>
    <input type="text" name="userName"/>
    <div>密碼</div>
    <input type="password" name="password"/>
    <div>EMAIL</div>
    <input type="text" name="email"/>
    <br/>
    <input type="submit" value="註冊"/>
    <input type="reset" value="取消"/>
</form>
```

```
public class MemberEntity implements java.io.Serializable{
    //Attribute
    private String userName;
    private String password;
    private String email;
    public String getUserName() {
        return userName;
    }
    //表單欄位 name=userName 透過Property setXxx() getXxx() 去頭 xxx=value
    public void setUserName(String userName) {
        this.userName = userName;
    }
    public String getPassword() {
        return password;
    }
```

透過 Model物件模組進行參考，傳遞至頁面進行渲染

透過 org.springframework.ui.Model進行狀態管理

- 持續狀態進入類似Map集合架構

預設Scope為Request Scope

Thymeleaf expression 渲染model狀態

```
//訂單作業控制器
@Controller
public class OrdersController {
    //訂單查詢作業-依照訂單編號
    @RequestMapping(path="/orders/qry",
            method= {RequestMethod.GET,RequestMethod.POST})
    public String ordersQry(String orderId,Model model) {
        if(orderId!=null) {
            //TODO 進行查詢作業
            System.out.println("輸入的訂單編號:"+orderId);
            model.addAttribute("orderId",orderId);
        }
        //進行狀態資料模組持續
        model.addAttribute("message","請輸入訂單編號");
        return "ordersqry";
    }

}
```

```
<body>
    <fieldset>
        <legend>訂單查詢</legend>
        <form>
            <div>訂單編號</div>
            <input type="text" th:placeholder="${message}" name="orderId"/>
            <input type="submit" value="查詢"/>
        </form>
    </fieldset>
    <div>輸入的訂單編號:<label th:text="${orderId}"></label></div>
</body>
```

透過 Model物件模組進行參考，傳遞至頁面進行渲染-Demo

總結：7-3 傳遞參數到頁面運算

了解spring mvc如何透過model進行狀態持續到View，與前端如何傳遞表單欄位架構之後，我們進一步了解如何在Spring Data Model如何在畫面中進行渲染