



巨匠線上真人

Java Web OCE JWCD元件系統 開發認證

www.pcschoolonline.com.tw



巨匠線上真人

Java Web OCE JWCD元件系統開發認證

第十五堂

RESTful Web Service設計與應用

本堂教學重點

- REST/RESTful軟體風格架構應用
- JSON文件格式
- JAX-RS API設計RESTful API
- 抽象Response物件應用
- RESTful 整合資料存取應用
- 前端JavaScript介接應用
- Servlet 整合RESTful service應用

REST/RESTful軟體風格架構應用

- **Representational State Transfer**，縮寫：**REST**，是Roy Thomas Fielding博士於2000年在他的博士論文中提出來的一種全球資訊網軟體架構風格。
- 目的是便於不同軟體/程式在網路（例如網際網路）中互相傳遞資訊。
- 採用於HTTP通訊協定之上的一組約束和屬性，是一種設計提供全球資訊網絡服務的軟體構建風格。
- 符合或相容於這種架構風格(簡稱為 **REST** 或 **RESTful**)的網路服務，允許用戶端發出以URI存取和操作網路資源的請求，而與預先定義好的無狀態操作集一致化。

REST架構的限制條件

客戶端-伺服器 (Client-Server)

無狀態 (Stateless)

快取 (Cacheability)

統一介面 (Uniform Interface)

分層系統 (Layered System)

Code-On-Demand

REST Web Service

- ◆ 符合REST設計風格的Web API稱為RESTful API。
- ◆ 直觀精簡的資源位址：
 - ◇ URI，比如：`http://example.com/resources`。
- ◆ 傳輸的資源：Web服務接受與返回的網際網路媒體類型
 - ◇ 比如：JSON，XML，YAML等。
- ◆ 對資源的操作：Web服務在該資源上所支援的一系列請求方法
 - ◇ 比如：POST，GET，PUT或DELETE

RESTful API Request Method應用

資源	GET	PUT	POST	DELETE
一組資源的URI，比如 https://www.gjun.com.tw/resources	列出URI，以及該資源組中每個資源資訊。	使用給定的一組資源置換目前整組資源。	在本組資源中建立或者新增一個新的資源。	刪除整組資源。
單個資源的URI，比如 https://www.gjun.com.tw/resources/007	取得指定007的資源的資訊。格式可以自選一個合適的網路媒體類型（比如：XML、JSON等）	置換或者建立指定的007資源。。	把指定007的資源當做一個資源組，並在其下建立或者新增一個新的元素，使其隸屬於目前資源。	刪除指定的元素

JSON文件格式

- ◆ JSON (JavaScript Object Notation) , 由道格拉斯·克羅克福特構想和設計、輕量級的資料交換語言，該語言以易於讓人閱讀的文字為基礎，用來傳輸由屬性值或者序列性的值組成的資料物件。
- ◆ JSON是JavaScript的一個子集，但JSON是獨立於語言的文字格式，並且採用了類似於C語言家族的一些習慣。
- ◆ JSON 的官方 MIME 類型是 application/json，副檔名是 .json。

```
[
  {
    "id": "0001",
    "time": "2019-12-20T02:33:58",
    "department": "巨匠",
    "value": [
      {
        "empid": 1,
        "name": "張無忌",
        "address": "台北市"
      },
      {
        "empid": 2,
        "name": "張三豐",
        "address": "台北市"
      }
    ]
  },
  {
    "id": "0002",
    "time": "2019-12-20T02:33:58",
    "department": "巨匠北認",
    "value": [
      {
        "empid": 3,
        "name": "張泰山",
        "address": "台北市"
      },
      {
        "empid": 4,
        "name": "張翠珊",
        "address": "台北市"
      }
    ]
  }
]
```

```
▼ array [2]
  ▼ 0 {4}
    id : 0001
    time : 2019-12-20T02:33:58
    ▼ value [2]
      ▼ 0 {3}
        empid : 1
        name : 張無忌
        address : 台北市
      ▼ 1 {3}
        empid : 2
        name : 張三豐
        address : 台北市
    department : 巨匠
  ► 1 {4}
```

JAX-RS API設計RESTful API

- ◆ Java API for RESTful Web Services是一個JEE API，用來支持 REST Web Service開發。
- ◆ JAX-RS使用了Annotation來簡化Web服務客戶端和服務端的開發和部署。

規劃一個REST Application，注入 Resource配置服務使用。

設計REST Resource類別及方法，提供服務對外的端點，與服務功能。

採用適當的Annotation進行 Resource類別與方法標註，設定服務特徵。

REST Application 設計

- ◆ 繼承 `javax.ws.rs.core.Application` 類別。可以產生一個 `Instance` 注入 `Resources` (REST 服務元件) 於應用系統中。
- ◆ 透過 `ApplicationPath` Annotation 配置 REST API 網站進入之後的端點起始位址 (URI)。
 - ◆ `http://hosted/web/v1`
 - ◆ `v1` 為服務進行端點起始位址

```
package com.gjun.service;  
  
import java.util.Set;  
  
import javax.ws.rs.ApplicationPath;  
import javax.ws.rs.core.Application;  
//配置服務入口位置  
@ApplicationPath("v1")  
public class MyApplication extends Application {  
  
    @Override  
    public Set<Class<?>> getClasses() {  
        // TODO Auto-generated method stub  
        return super.getClasses();  
    }  
}
```

REST Service Resource設計

HelloService.java

- ◆ 採用簡單的類別規劃，繼承Root Class(Object)，進行方法設計。
- ◆ HelloService簡易打招呼方法設計。
- ◆ 使用@Path設定Class或者Method URI目錄。
- ◆ 使用 @GET/@POST/@PUT/@DELETE設定傳送方法。
- ◆ 使用@Produces設定回應的資料 Media Type(Content-Type)

```
package com.gjun.service;

import javax.ws.rs.GET;
//打招呼服務
@Path("hello")
public class HelloService {
    //使用Annotation配置服務特徵
    @Path("helloworld")
    @GET
    @Produces("text/plain")
    public String helloWorld()
    {
        return "Hello World";
    }
}
```

使用Postman測試HelloService

◆ URL

◆ `http://hosted:8080/mod15/v1/hello/helloworld`

◆ 請求傳送方式GET

GET `http://localhost:8080/mod15/v1/hello/helloworld` Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (4) Test Results

Status: 200 OK Time: 233ms Size: 135 B Save

Pretty Raw Preview Visualize BETA Text

1 Hello World

Body Cookies (1) Headers (4) Test Results

Status: 200 OK

KEY	VALUE
Date	Sun, 05 Jan 2020 23:09:26 GMT
Content-Type	text/plain
Content-Length	11
Server	Apache TomEE

採用QueryString傳遞資訊

- ◆ Resource設計可採用QueryString方式傳遞多個參數。
 - ◇ http://hosted/web/xxx/xxx?參數=值&...
- ◆ 採用@QueryParam Annotation設定。綁定在Resource相對的方法參數中。

HelloService.java

```
import javax.ws.rs.GET;
//打招呼服務
@Path("hello")
public class HelloService {
    //使用Annotation配置服務特徵
    @Path("helloworld")
    @GET
    @Produces("text/plain")
    public String helloworld()
    {
        return "Hello World";
    }


    //採用QueryString方式傳遞參數值
    @Path("hihello")
    @GET
    @Produces("text/plain")
    public String helloworld(@QueryParam("w")String who) {
        return String.format("%s 您好",who);
    }
}
```

採用URL Path傳遞資訊

- ◆ REST Service可透過URI Path當作參數進行傳遞。
 - ◇ `http://hosted:8080/mod15/v1/hello/gjunhello/{message}/rawdata`
- ◆ 可以使用@PathParam Annotation進行與Method參數綁定。

//採用QueryString方式傳遞參數值|

```
@Path("gjunhello/{msg}/rawdata")
@GET
@Produces("text/plain")
public String gjunWorld(@PathParam("msg")String message) {
    return String.format("%s 您好",message);
}
```



JSON內容回應應用

- ◆ 直接在Resource Method回應JavaBean，交由JEE Container直接序列化JSON。
- ◆ JavaBean設計。
- ◆ @Produces Media Type application/json

```
package com.gjun.domain;
//JavaBean設計
public class Customers implements java.io.Serializable {
    private String customerId;
    private String companyName;
    private String address;
    private String phone;
    public String getCustomerId() {
        return customerId;
    }
    public void setCustomerId(String customerId) {
        this.customerId = customerId;
    }
    public String getCompanyName() {
        return companyName;
    }
    public void setCompanyName(String companyName) {
        this.companyName = companyName;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public String getPhone() {
        return phone;
    }
    public void setPhone(String phone) {
```

Customers.java

CustomersService.java

```
@Path("customers")
public class CustomersService {

    @Path("getcustomers")
    @GET
    @Produces("application/json")
    public Customers getCustomers() {
        Customers customers=new Customers();
        customers.setCustomerId("0001");
        customers.setCompanyName("巨匠電腦");
        customers.setAddress("台北市");
        customers.setPhone("02-12345678");
        return customers;
    }
}
```

REST Resource回應JSON示範

- ◆ 採用PostMan進行測試
- ◆ `http://localhost:8080/mod15/v1/customers/getcustomers`

The screenshot displays the Postman interface for a REST client. The request method is GET, and the URL is `http://localhost:8080/mod15/v1/customers/getcustomers`. The response status is 200 OK. The response body is a JSON object, and the Content-Type header is `application/json`.

Request:

- Method: GET
- URL: `http://localhost:8080/mod15/v1/customers/getcustomers`

Response:

Status: 200 OK

Body:

```
1 {
2   "address": "台北市",
3   "companyName": "巨匠電腦",
4   "customerId": "0001",
5   "phone": "02-12345678"
6 }
```

Headers:

KEY	VALUE
Date	Mon, 06 Jan 2020 00:30:28 GMT
Content-Type	application/json
Transfer-Encoding	chunked
Server	Apache TomEE

POST JSON to REST Service

- ◆ 採用@Consumes Annotation描述傳送進來的資料格式(Media Type)
- ◆ 直接設定JavaBean類別類型參數，透過JEE Container直接反序列化物件。

Message.java

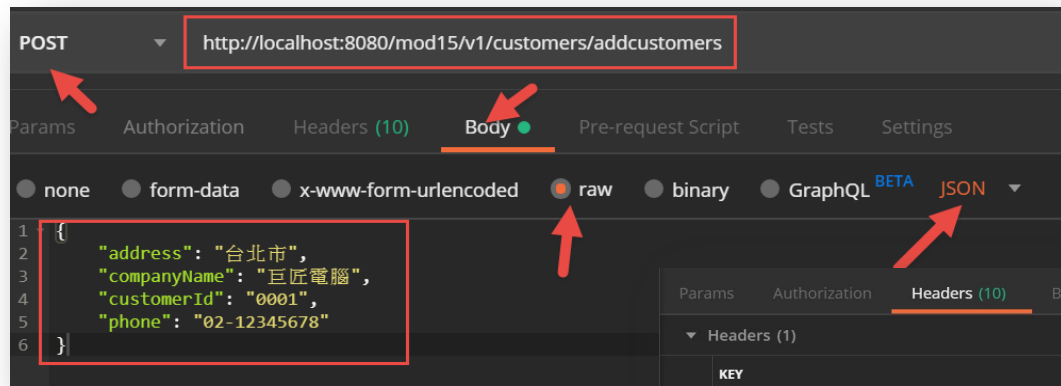
```
package com.gjun.domain;
//JavaBean設計
public class Message {
    //attribute
    private String message;
    private int code;
    public String getMessage() {
        return message;
    }
    public void setMessage(String message) {
        this.message = message;
    }
    public int getCode() {
        return code;
    }
    public void setCode(int code) {
        this.code = code;
    }
}
```

CustomersService.java

```
//新增客戶資料
@Path("addcustomers")
@POST
@Produces("application/json")
@Consumes("application/json")
public Message addCustomers(Customers customers) {
    //處理
    Message msg=new Message();
    msg.setMessage(String.format("客戶編號:%s 公司行號:%s",
        customers.getCustomerId(),customers.getCompanyName()));
    msg.setCode(200);
    return msg;
}
```

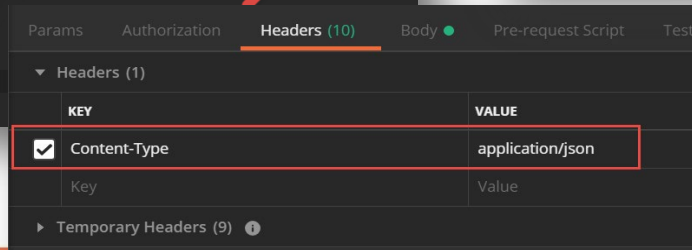

上傳JSON文件應用

- ◆ 採用PostMan前端工具進行測試
- ◆ 設定Body為Raw Data
- ◆ 並且設定Request Header Content-Type:application/json
- ◆ 採用Request Method為POST



```
POST /mod15/v1/customers/addcustomers HTTP/1.1
Host: localhost:8080
Content-Type: application/json

{
  "address": "台北市",
  "companyName": "巨匠電腦",
  "customerId": "0001",
  "phone": "02-12345678"
}
```



測試結果

The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8080/mod15/v1/customers/addcustomers
- Body Tab:** Selected, showing a JSON payload:

```
{  "address": "台北市",  "companyName": "巨匠電腦",  "customerId": "0001",  "phone": "02-12345678"}
```
- Response Tab:** Selected, showing a JSON response with a 200 status code and a message:

```
{  "code": 200,  "message": "客戶編號:0001 公司行號:巨匠電腦"}
```

The response is highlighted with a red rectangle. The interface also includes tabs for Params, Authorization, Headers (10), Pre-request Script, Tests, and Settings. The response is also viewable in Pretty, Raw, Preview, and Visualize (BETA) formats, with JSON selected.

抽象Response物件應用-1

- ◆ 透過Response物件回應資訊，優勢可以確定Http Statuscode等資訊。
- ◆ 可以因應REST Service例外處理時，回應自訂符合標準的StatusCode與訊息到前端。
- ◆ 透過Response回應,可以調用不同的回應JSON格式
- ◆ 透過Response回應可以針對HTTP協定設定前置作業的設定，如Header/Cookie等。
- ◆ 情境題
 - ◇ 如果查詢客戶沒有傳遞Key進來，則回應400狀態碼,且回應Message物件序列化JSON文件格式。
 - ◇ 如果傳遞Key之後，則產生Customers物件序列化回應Status code -200訊息。

CustomersService.java

```
@Path("saycustomers")
@GET
@Produces("application/json")
public Response sayCustomers(@QueryParam("key")String key) {
    if(key!=null) {
        Customers customers=new Customers();
        customers.setCustomerId("0001");
        customers.setCompanyName("巨匠電腦");
        customers.setAddress("台北市");
        customers.setPhone("02-12345678");
        //回應statuscode 200正確訊息
        return Response.ok()
            .header("status", "ok")
            .encoding("UTF-8")
            .entity(customers)
            .build();
    }else
```

抽象Response物件應用-2

```
}else
{
    Message msg=new Message();
    msg.setMessage("查詢鍵值尚未輸入");|
    msg.setCode(400);
    //回應statuscode 400-bad request訊息
    return Response.status(400)
        .entity(msg)
        .build();
}
```

GET <http://localhost:8080/mod15/v1/customers/saycustomers?key=100>

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> key	100	
Key	Value	Description

Body Cookies (1) Headers (6) Test Results Status: 200 OK Time: 11

Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "address": "台北市",
3   "companyName": "巨匠電腦",
4   "customerId": "0001",
5   "phone": "02-12345678"
6 }
```

GET <http://localhost:8080/mod15/v1/customers/saycustomers>

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

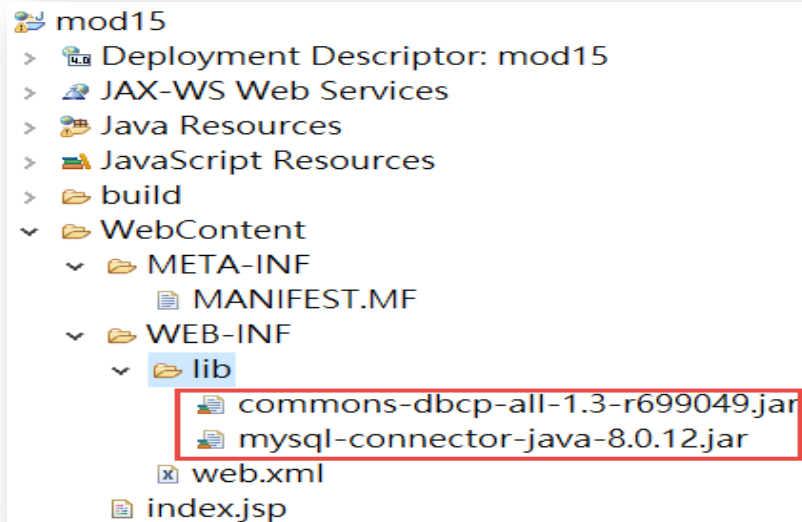
Body Cookies (1) Headers (5) Test Results Status: 400 Bad Request Time:

Pretty Raw Preview Visualize BETA JSON

```
1 {
2   "code": 400,
3   "message": "查詢鍵值尚未輸入"
4 }
```

RESTful 整合資料存取應用

- ◆ 採用DAO設計模式自訂資料存取元件。
 - ◆ DAO介面設計
 - ◆ DAO Implements
- ◆ 應用DI(Dependency Injection)軟體工程，設計DAO資料存取元件與DataSource注入依賴互動關係。
- ◆ 使用Apache DBCP DataSource API



Entity Class規劃

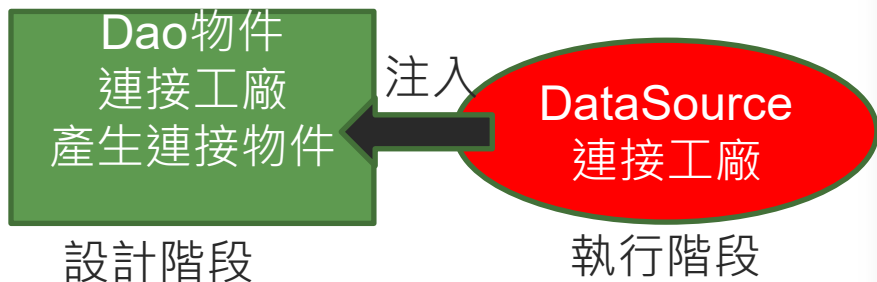
GjunCustomers.java

- ◆ 查詢MySQL sakila範例資料庫中的customer Table。
- ◆ 規劃Entity Class-GjunCustomers

```
public class GjunCustomers implements java.io.Serializable{  
    private Short customer_id;  
    private String first_name;  
    private String last_name;  
    private String emai;  
    private Byte active;  
    public Short getCustomer_id() {  
        return customer_id;  
    }  
    public void setCustomer_id(Short customer_id) {  
        this.customer_id = customer_id;  
    }  
    public String getFirst_name() {  
        return first_name;  
    }  
    public void setFirst_name(String first_name) {  
        this.first_name = first_name;  
    }  
    public String getLast_name() {  
        return last_name;  
    }  
    public void setLast_name(String last_name) {  
        this.last_name = last_name;  
    }  
    public String getEmai() {  
        return emai;  
    }  
}
```

DAO 設計模式

- ◆ 設計Idao介面，規範查詢方法架構。
- ◆ 設定屬性注入DataSource-setter，形成依賴注入架構。



IDao.java

```
package com.gjun.domain;

import java.sql.SQLException;

//DAOe功能規範設計
//採用泛型 決定查詢回應值類型
public interface IDao<T> {
    //單筆查詢
    public T selectForObject(String sqlString,Object[] args) throws SQLException;
    //注入DataSource
    public void setDataSource(DataSource datasource);
    //多筆資料查詢
    public List<T> selectForList(String sqlString,Object[] args) throws SQLException;
}
```

DAO介面實作-Customer DAO

◆ 實作DAO介面查詢方法。

```
public class CustomerDao implements IDao<GjunCustomers> {  
    //attribute  
    private DataSource datasource;  
  
    @Override  
    public GjunCustomers selectForObject(String sqlString, Object[] args) throws SQLException {  
        GjunCustomers customers=null;  
        if(this.datasource==null) {  
            throw new SQLException("DataSource物件尚未注入");  
        }  
        //1.透過DataSource取得連接物件  
        Connection connection=datasource.getConnection();  
        //2.產生命令物件  
        PreparedStatement st=connection.prepareStatement(sqlString);  
        //走訪參數值 設定設定內容  
        int start=1;  
        for(Object value:args) {  
            st.setObject(start, value);  
            start++;  
        }  
        //執行查詢  
        ResultSet rs=st.executeQuery();  
        if (rs.next()) {  
            customers=new GjunCustomers();  
            customers.setCustomer_id(rs.getShort("customer_id"));  
            customers.setFirst_name(rs.getString("first_name"));  
            customers.setLast_name(rs.getString("last_name"));  
            customers.setEmai(rs.getString("email"));  
        }  
        connection.close();  
    }  
}
```

CustomerDao.java

```
@Override  
public void setDataSource(DataSource datasource) {  
    this.datasource=datasource;  
}
```

```
@Override  
public List<GjunCustomers> selectForList(String sqlString, Object[] args) throws SQLException {  
    List<GjunCustomers> customers=new ArrayList<>();  
    if(this.datasource==null) {  
        throw new SQLException("DataSource物件尚未注入");  
    }  
    //1.透過DataSource取得連接物件  
    Connection connection=datasource.getConnection();  
    //2.產生命令物件  
    PreparedStatement st=connection.prepareStatement(sqlString);  
    //走訪參數值 設定設定內容  
    int start=1;  
    for(Object value:args) {  
        st.setObject(start, value);  
        start++;  
    }  
    //執行查詢  
    ResultSet rs=st.executeQuery();  
    while(rs.next()) {  
        GjunCustomers customer=new GjunCustomers();  
        customer=new GjunCustomers();  
        customer.setCustomer_id(rs.getShort("customer_id"));  
        customer.setFirst_name(rs.getString("first_name"));  
        customer.setLast_name(rs.getString("last_name"));  
        customer.setEmai(rs.getString("email"));  
        //集合參考  
        customers.add(customer);  
    }  
    connection.close();  
    return customers;  
}
```


REST Service資料存取服務

設計方法

定義Path/Produces/GET等
Annotation

建構DataSource物件與DAO物件

執行查詢

```
//客戶資料查詢 By 客戶編號
@Path("customersqry/id/{customerid}/rawdata")
@GET
@Produces("application/json")
public GjunCustomers custqryById(@PathParam("customerid") short customerid) {
    GjunCustomers result=null;
    //建構DataSource物件
    BasicDataSource datasource=new BasicDataSource();
    datasource.setDriverClassName("com.mysql.cj.jdbc.Driver");
    datasource.setUrl("jdbc:mysql://localhost:3306/sakila?serverTimezone=UTC"
        + "&useUnicode=true&characterEncoding=utf8&useSSL=false");
    datasource.setUsername("root");
    datasource.setPassword("1111");
    //建構DAO
    IDao<GjunCustomers> dao=new CustomerDao();
    dao.setDataSource(datasource);
    try {
        result=dao.selectForObject("select customer_id,first_name,last_name,email "
            + "from customer where customer_id=?",
            new Object[] {customerid});
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return result;
}
```

Postman單元測試

GET http://localhost:8080/mod15/v1/customers/customersqry/id/2/rawdata Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (4) Test Results Status: 200 OK Time: 26ms Size: 246 B Save

Pretty Raw Preview Visualize BETA JSON ↕

```
1 {
2   "customer_id": 2,
3   "emai": "PATRICIA.JOHNSON@sakilacustomer.org",
4   "first_name": "PATRICIA",
5   "last_name": "JOHNSON"
6 }
```

Body Cookies (1) Headers (4) Test Results Status: 200 OK Time: 26ms

KEY	VALUE
Date	Thu, 09 Jan 2020 17:04:55 GMT
Content-Type	application/json
Transfer-Encoding	chunked
Server	Apache TomEE

前端JavaScript介接應用

網頁設計查詢介面

採用jQuery Framework

使用JavaScript進行Ajax程序設計

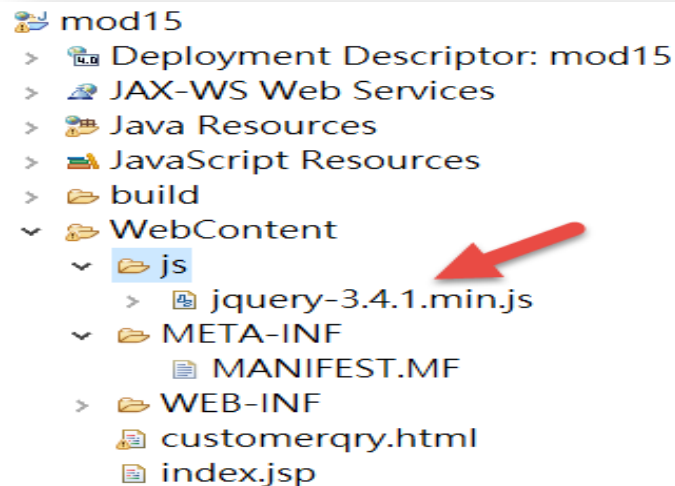
呼喚REST Service進行資料查詢

結果回呼程序，更新畫面

- ◆ 使用JavaScript撰寫AJAX非同步處理，呼喚REST Service。
- ◆ 使用jQuery Framework ajax()實現AJAX。
- ◆ 下載jQuery Core與網頁引用。
 - ◇ <https://jquery.com/download/>

Download the compressed, production jQuery 3.4.1

Download the uncompressed, development jQuery 3.4.1



網頁設計查詢介面/引用jQuery Framework

customerqry.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>客戶資料查詢</title>
<script src="js/jquery-3.4.1.min.js"></script>
<script>
    //使用jquery Selector-選擇文件下載瀏覽起引發事件
    $(document).ready(
        function(){
            alert('hi jquery');
        }
    );
</script>
</head>
<body>
<fieldset>
    <legend>客戶資料查詢</legend>
    <div>客戶編號</div>
    <input type="text" id="customerid"/>
    <input type="button" value="查詢"/>
</fieldset>
</body>
</html>
```

客戶資料查詢

客戶編號

查詢

網站訊息...

hi jquery

☐ 不要讓此網頁建立更多訊息

確定

使用JavaScript進行Ajax程序設計

```
<script>
//使用jquery Selector-選擇文件下載瀏覽起引發事件
$(document).ready(
function(){
    //參照button物件
    $('#btnQuery').click(
function(){
    //參照文字輸入方塊
    var key=$('#customerid').val();
    var urlString="http://localhost:8080/mod15/v1/customers/customersqry/id/"+key+"/rawdata";
    //設定物件(settings)
    var params={
        url:urlString,
        type:'GET',
        success:function(result,status,xhr){
            alert(result.first_name);
        },
        error:function(xhr,status,error){
            alert(error);
        }
    }
    $.ajax(params);
});
});
</script>
```

客戶資料查詢

客戶編號

網站訊息...

MARY

確定

結果回呼程序，更新畫面

```
<fieldset id="result">
  <legend>查詢結果</legend>
  <table border="1">
    <tr>
      <td>客戶編號</td>
      <td id="cid"></td>
    </tr>
    <tr>
      <td>First Name</td>
      <td id="firstname"></td>
    </tr>
    <tr>
      <td>Last Name</td>
      <td id="lastname"></td>
    </tr>
  </table>
</fieldset>
```

```
//設定物件(settings)
var params={
  url:urlString,
  type:'GET',
  success:function(result,status,xhr){
    $('#cid').text(result.customer_id);
    $('#firstname').text(result.first_name);
    $('#lastname').text(result.last_name);
    $('#result').show();
  },
  error:function(xhr,status,error){
    alert(error);
  }
}
$.ajax(params);
```

Servlet 整合RESTful service應用

- ◆ 使用java.net package進行遠端服務呼喚設計。
- ◆ 使用java.io進行Response回應資料讀取設計。

CustomesQryServlet.java

```
@WebServlet("/CustomersQryServlet")
public class CustomersQryServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

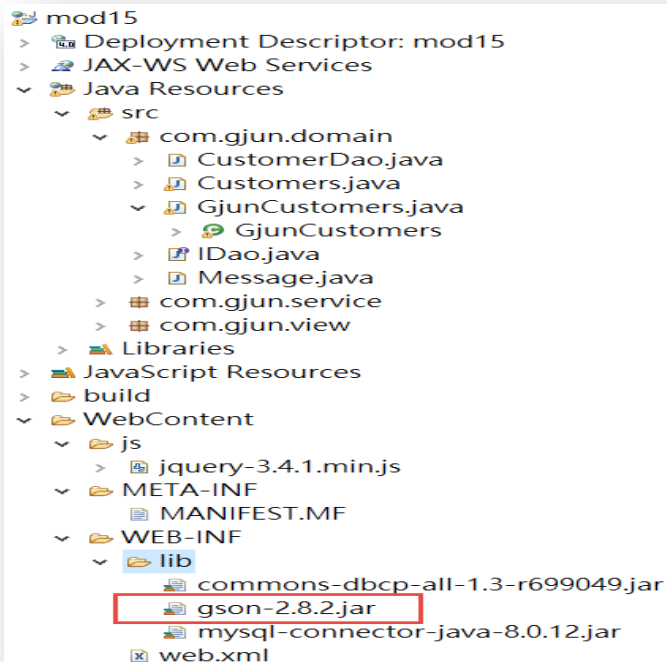
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        //擷取表單欄位參數
        String customerid=request.getParameter("customerid");
        //服務URL設定
        String urlString=String.format("http://localhost:8080/mod15/"
            + "v1/customers/customersqry/id/%s/rawdata",customerid);
        //建構URL物件
        URL url=new URL(urlString);
        //開啟連接(Request需求)
        HttpURLConnection connection=(HttpURLConnection)url.openConnection();
        //設定傳送方式為GET(配合後端服務設定)
        connection.setRequestMethod("GET");
        //取得回應資訊的InputStream
        InputStream is=connection.getInputStream();
    }
}
```

讀取回應的串流資訊

```
//讀取文字串
InputStreamReader reader=new InputStreamReader(is);
BufferedReader buffer=new BufferedReader(reader);
StringBuilder builder=new StringBuilder();
String line=null;
while((line=buffer.readLine())!=null) {
    builder.append(line);
}
String content=builder.toString();
```


反序列化JsonString為物件

◆ 使用Google Gson API



```
String content=builder.toString();
```

```
//反序列化物件
```

```
Gson gson=new Gson();
```

```
GjunCustomers customers=gson.fromJson(content, GjunCustomers.class);
```

```
reader.close();
```

```
//回應物件資訊
```

```
PrintWriter out=response.getWriter();
```

```
out.println(String.format("<h1>FristName: %s</h1>",customers.getFirst_name()));
```

Lab

- ◆ 何謂REST/RESTful軟體風格
- ◆ 採用JEE JAX-RS API如何建立起一個REST Service，如何啟動Register Resource
- ◆ 設計一個採用POST Method與採用Path傳遞Key的Service，進行客戶資料查詢
- ◆ JAX-RS如何客製化Service回應訊息的Status Code，與Body 中Json資訊