

REST/RESTful



段維瀚 老師

REST/RESTful

- **R**epresentational **S**tate **T**ransfer，簡稱**REST**，它是一種網路架構風格，並不是一種標準。
- 而 **RE****S****T****f****u****l** 可以這樣子想像：美麗(**B**eauty)的事物可以稱為 **Beautif****u****l**；設計為 **RE****S****T** 的系統就可以稱為 **RE****S****T****f****u****l**



Web API 風格

- 傳統

- 取得所有使用者 GET /getAllUsers
- 取得單筆使用者 GET /getUser/1
- 新增使用者資料 POST /createUser
- 更新使用者資料 POST /updateUser/1
- 刪除使用者資料 POST /deleteUser/1

- REST

- 取得所有使用者 GET /rest/users
- 取得單筆使用者 GET /rest/user/1
- 新增使用者資料 POST /rest/user
- 更新使用者資料 PUT /rest/user/1
- 刪除使用者資料 DELETE /rest/user/1



JAX-RS

- Java API for RESTful Web Services (JAX-RS), is a set of APIs to develop REST service.
- JAX-RS is part of the Java EE6, and make developers to develop REST web application easily.
 - both Jersey and RESTEasy, popular JAX-RS implementation.
 - <https://www.mkyong.com/tutorials/jax-rs-tutorials/>



Jersey

- Jersey
 - 以前叫 Glassfish Jersey
 - 現在叫 Eclipse Jersey
 - 他是一個 REST 框架，它提供JAX-RS（JSR-370）實現以及更多其他功能
 - 核心服務器：用於基於註釋構建RESTful服務（jersey-core，jersey-server，jsr311-api）
 - 核心客戶端：幫助您與REST服務進行通信（jersey-client）
 - JAXB（Java映射XML）支持、JSON支持
 - Spring、Guice（輕量級 Spring）的整合模組



Jersey 資源部署

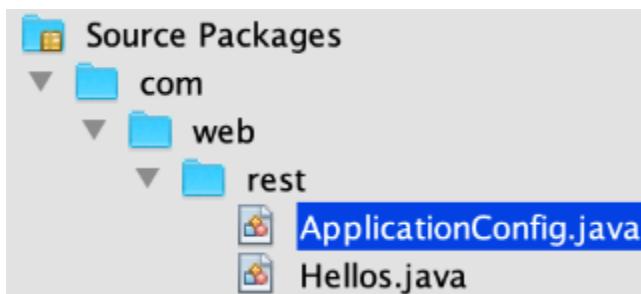
- pom.xml

```
<dependency>
    <groupId>org.glassfish.jersey.containers</groupId>
    <artifactId>jersey-container-servlet-core</artifactId>
    <version>2.6</version>
</dependency>

<dependency>
    <groupId>org.glassfish.jersey.core</groupId>
    <artifactId>jersey-client</artifactId>
    <version>2.6</version>
</dependency>
```



Jersey 路徑配置



```
package com.web.rest;

import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

@Path("rest")
public class JerseyApplication extends Application {

    public JerseyApplication() {
    }

}
```

REST 風格設計

- GET → /rest/hellos
- GET → /rest/hellos/1
- POST → /rest/hellos?value=HelloC
- PUT → /hellos/2?value=HelloD
- DELETE → /rest/hellos/2



```

@Path("/hellos")
public class Hellos {
    private static List<String> hellos = new ArrayList<>(Arrays.asList("HelloA", "HelloB"));

    @GET
    public String list() {
        return hellos.toString();
    }

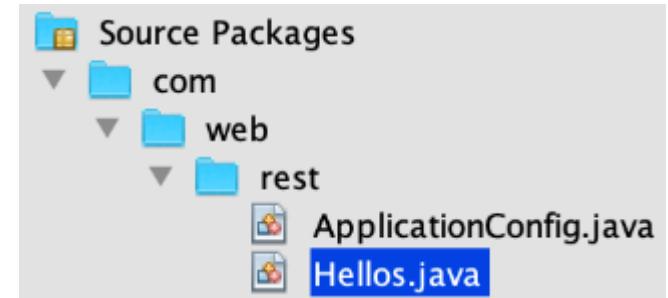
    @GET
    @Path("/{id}")
    public String show(@PathParam("id") int id) {
        return hellos.get(id);
    }

    @POST
    public String add(@QueryParam( "value" ) String value) { // @FormParam 表單資料
        hellos.add(value);
        return "add ok";
    }

    @PUT
    @Path("/{id}")
    public String update(@PathParam("id") int id, @QueryParam("value") String value) {
        hellos.set(id, value);
        return "update ok";
    }

    @DELETE
    @Path("/{id}")
    public String delete(@PathParam("id") int id) {
        hellos.remove(id);
        return "delete ok";
    }
}

```



使用 Postman 測試



The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Postman' and 'My Workspace' (with a dropdown arrow) and an 'Invite' button. Below the navigation bar, there are five tabs for different HTTP methods: GET, POST, PUT, and DEL, each with a red dot indicating they have been used. A '+' button is also present. The main area shows a request configuration for a GET method to 'http://localhost:8080/javaWeb_Jersey/rest/hellos'. The 'Params' tab is selected, showing a table with one row:

KEY	VALUE	DESCRIPTION
Key	Value	Description

Below the table, there are tabs for 'Body', 'Cookies', 'Headers (4)', and 'Test Results'. The 'Body' tab is selected. At the bottom of the body section, there are buttons for 'Pretty', 'Raw', 'Preview', and 'Auto' (with a dropdown arrow), followed by a small orange square icon.

JQuery

```
<script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.4.1.min.js"></script>
<script>
    function update(id) {
        var url = './rest/hellos/' + id;
        var data = $('form').serialize();
        $.ajax({
            url: url,
            type: 'PUT',
            data: data,
            success: function (result) {
                console.log(result)
            }
        });
    }
</script>
```

```
<form>
    <input type="text" name="value">
    <input type="button" onclick="update(0)" value="update">
</form>
```

補充：簡易 *restful* 實作



```
public class RestRequest {
    private Pattern regExAllPattern = Pattern.compile("/user");
    private Pattern regExIdPattern = Pattern.compile("/user/([0-9]*)");

    private int id;

    public RestRequest(String pathInfo) throws ServletException {
        // regex parse pathInfo
        Matcher matcher;
        // Check for ID case first, since the All pattern would also match
        matcher = regExIdPattern.matcher(pathInfo);
        if (matcher.find()) {
            id = Integer.parseInt(matcher.group(1));
            return;
        }

        matcher = regExAllPattern.matcher(pathInfo);
        if (matcher.find())
            return;

        throw new ServletException("Invalid URI");
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
}
```

```
@WebServlet("/rest/*")
public class TestRestServlet extends HttpServlet {
    protected void doHandler(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        PrintWriter out = response.getWriter();
        out.println("method : " + request.getMethod());
        out.println("getPathInfo() : " + request.getPathInfo());
        out.println("getParameterMap() : " + request.getParameterMap());
        out.println("getParameterMap().size() : " + request.getParameterMap().size());
        try {
            RestRequest restRequest = new RestRequest(request.getPathInfo());
            out.println("restRequest.getId() : " + restRequest.getId());
        } catch (ServletException e) {
            e.printStackTrace();
            out.println(e.toString());
        }
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        doHandler(request, response);
    }

    //...續作 doPost()、doPut() 與 doDelete()
}
```

Lab

- /rest/user/1

終·身·學·習·好·伙·伴

Ajax

```
function getAllUsers() {  
    var action = "./rest/user";  
    var xhttp = new XMLHttpRequest();  
    xhttp.onload = callback;  
    xhttp.open("get", action);  
    xhttp.send();  
}  
  
function callback() {  
    var content = this.responseText;  
    console.log(content);  
    document.getElementById('result').innerText = content;  
}
```

Path

Method

