

# Java EE 7 Web ORM 與 JPA

鄭安翔

ansel\_cheng@hotmail.com

---

# 課程大綱

- 1) **Maven**
- 2) **ORM - Java Persistence API**

# Maven

## ■ Apache Maven

### □ 專案管理及自動構建的工具

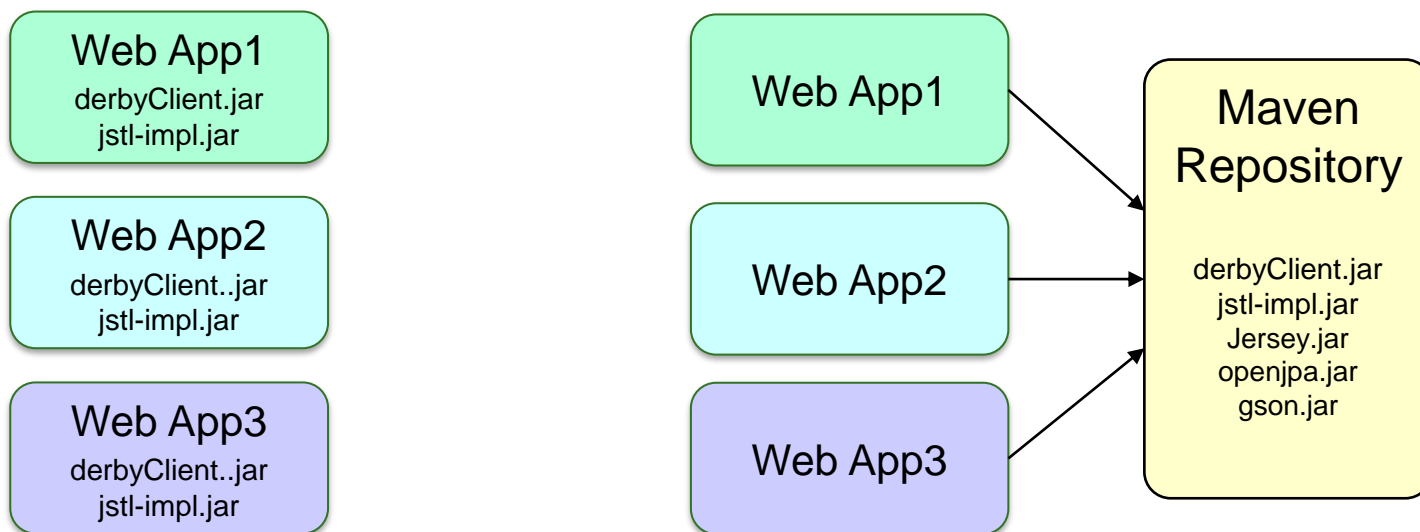
- Apache 基金會開發開放原始碼專案
- 專家、內行的意思
- 基於專案物件模型 Project Object Model (POM)

### □ 用途

- 專案自動建構
- 專案函式庫(JAR)共用
- 函式庫相依管理
- 專案測試及報告

# 專案函式庫(JAR)共用

- 減少專案所占空間大小
  - 傳統應用程式將用到的函式庫加入專案中
  - **Maven** 將用到的函式庫放在 **Maven Repository** 倉庫中
    - 可被多個應用程式共用



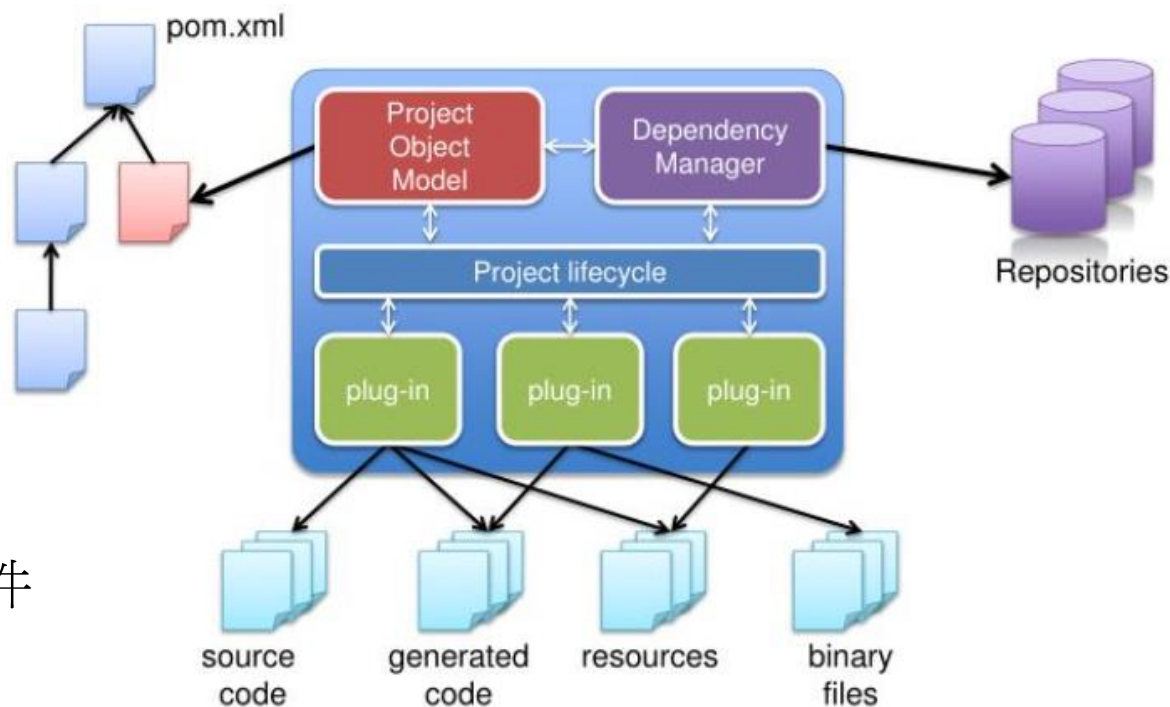
# 函式庫(JAR)相依管理

## ■ pom.xml

□ 描述專案如何建置及宣告專案模組依賴關係

□ 設定資訊

- 依賴函式庫
- 外掛
- 執行佈署目標
- 專案版本
- 專案建置 profile
- 專案開發者及郵件



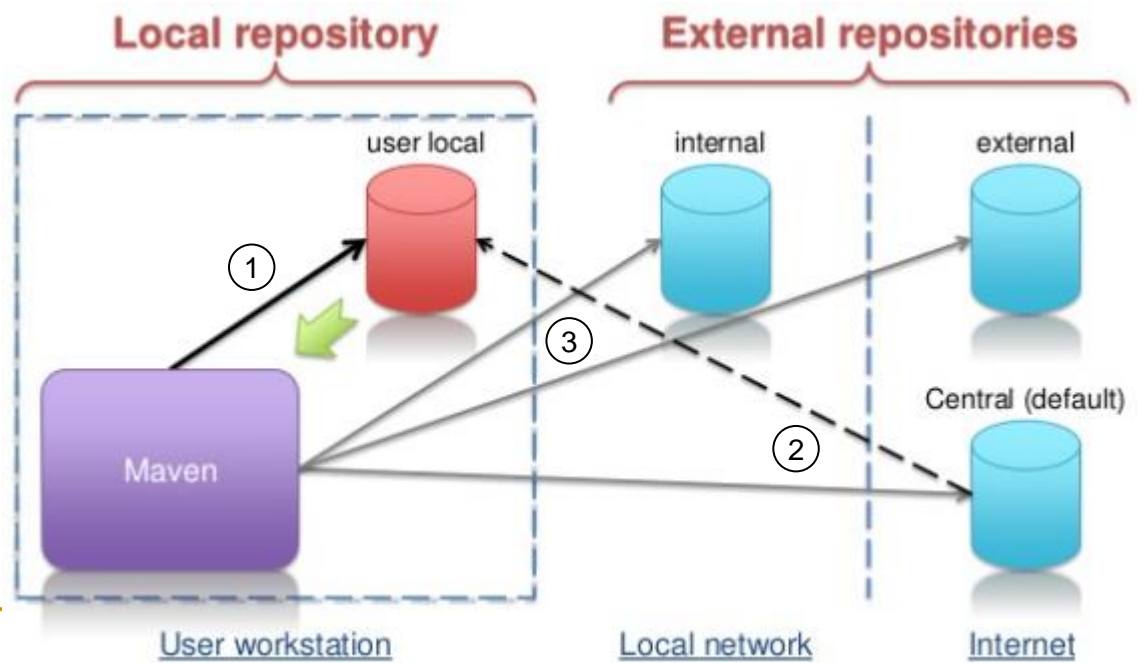
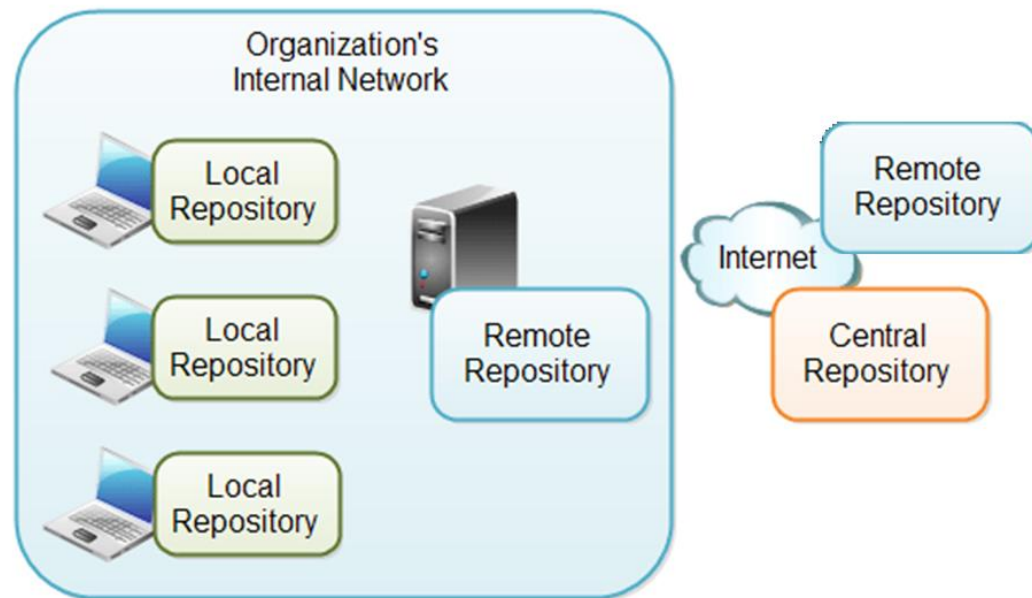
# POM 相依性設定

```
<project>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.6</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Result Scope	Compile classpath	Result package	Test classpath	Lookup repository
compile	+	+	+	+
provided	+	-	+	+
runtime	-	+	+	+
test	-	-	+	+
system	+	-	+	-

# Repository 倉庫類型

- 本地倉庫 local repository
  - 預設放在 `<user_home>/m2/repository/`
  - 函式庫優先在本地倉庫尋找
- 中央倉庫 central repository
  - Maven 社群提供，包含了大量常用的開源碼函式庫
  - 本地倉庫沒有的函式庫，嘗試從遠端倉庫下載至本地倉庫
  - 需有網路連線
- 遠端倉庫 remote repository
  - 自訂倉庫
  - 可在企業內部網路或外部雲端
- 本地 -> 中央 -> 遠端





# POM 倉庫及外掛設定

```
<project>
  <repositories>
    <repository>
      <id>Jboss.repository</id>
      <url>
        http://repository.jboss.org/maven2
      </url>
    </repository>
  </repositories>
</project>
```

```
<project>
  <build>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <configuration>
          <source>1.5</source>
          <target>1.5</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

# Maven 三個生命週期階段

## ■ Clean 生命週期

- 專案清理

## ■ Build 生命週期

- 編譯
- 測試
- 包裝
- 安裝
- 佈署

## ■ Site 生命週期

- 建立檔案、報告
- 佈署至網站

Clean Lifecycle
pre-clean
<b>clean</b>
post-clean

Default Lifecycle	
validate	test-compile
initialize	process-test-classes
generate-sources	test
process-sources	prepare-package
generate-resources	<b>package</b>
process-resources	pre-integration-test
<b>compile</b>	integration-test
process-classes	post-integration-test
generate-test-sources	verify
process-test-sources	<b>install</b>
generate-test-resources	<b>deploy</b>
process-test-resources	

Site Lifecycle
pre-site
<b>site</b>
post-site
site-deploy

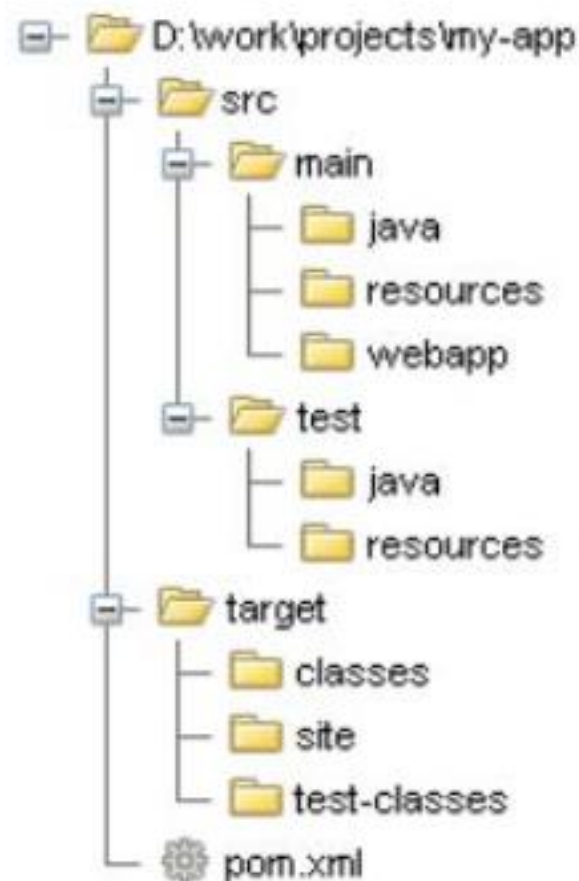
# 專案建構 (build project)

## ■ Maven專案建構生命週期

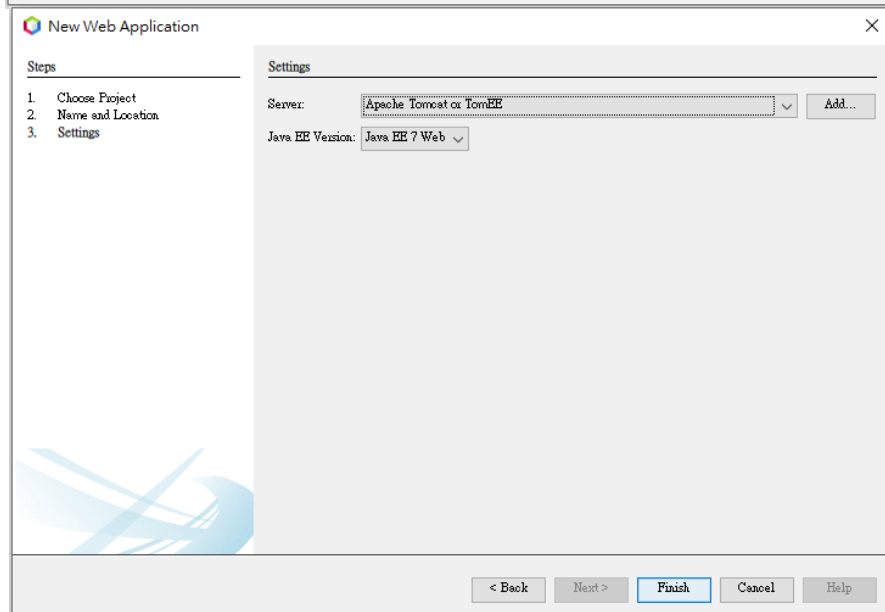
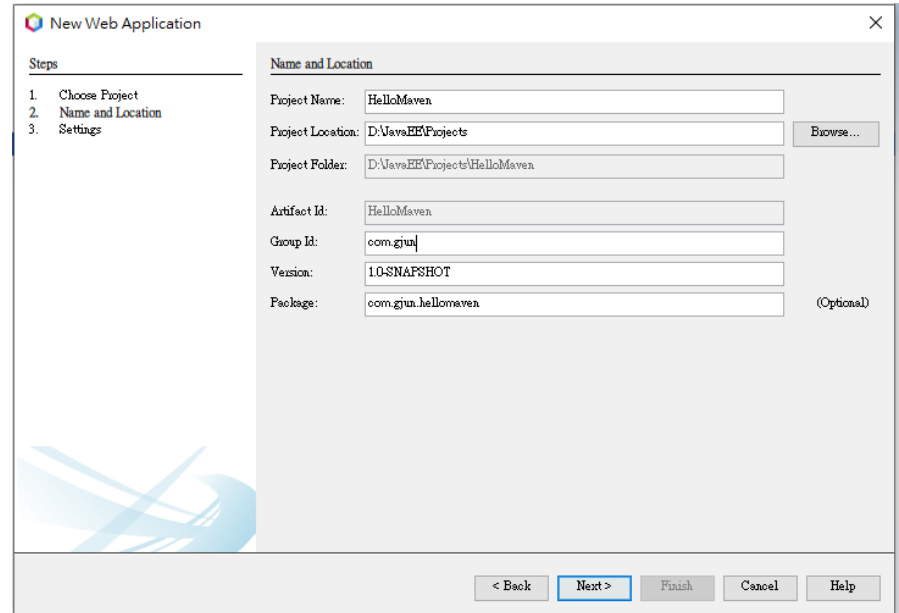
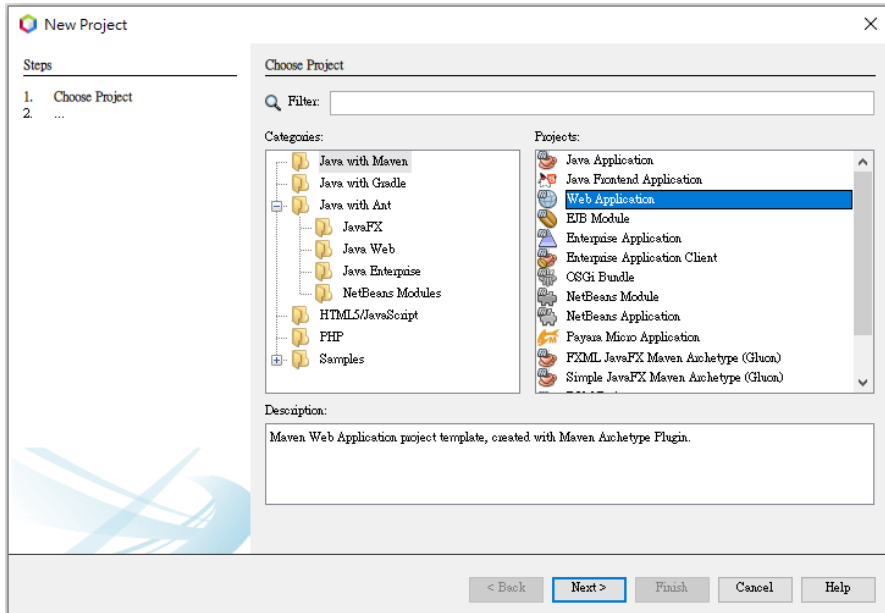
- 驗證 validate
- 編譯 compile
- 測試 test
- 包裝 package
- 安裝 install
- 佈署 deploy

## ■ Maven 自動化建構

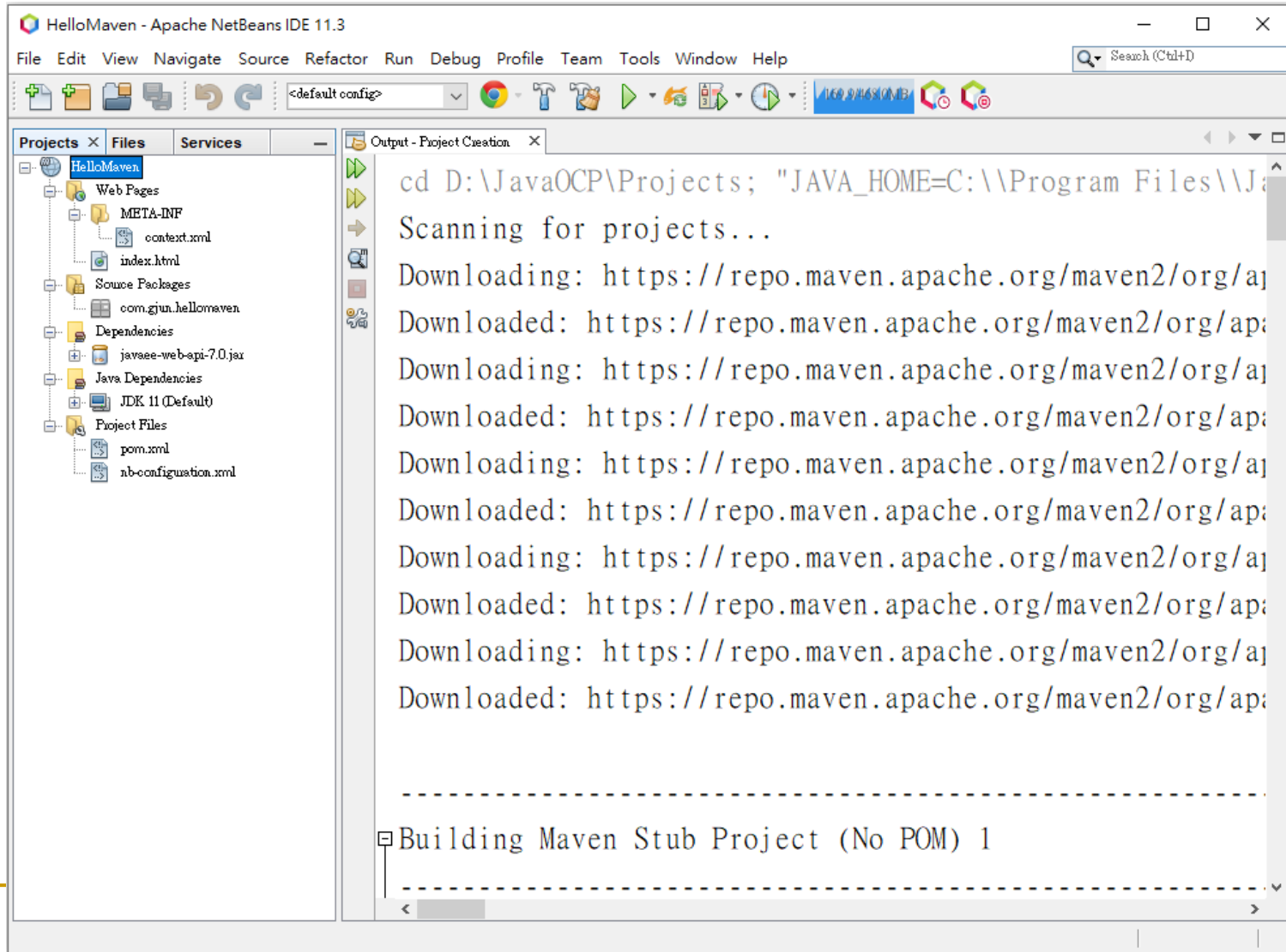
- 大型專案團隊開發時，簡化建構工作



# 使用 NetBeans 建立 Maven 專案



# HelloMaven專案



# 新增JSTL 相依性

The screenshot shows the Apache NetBeans IDE 11.3 interface. The main window displays the `pom.xml` file for a project named "HelloMaven". The XML content is as follows:

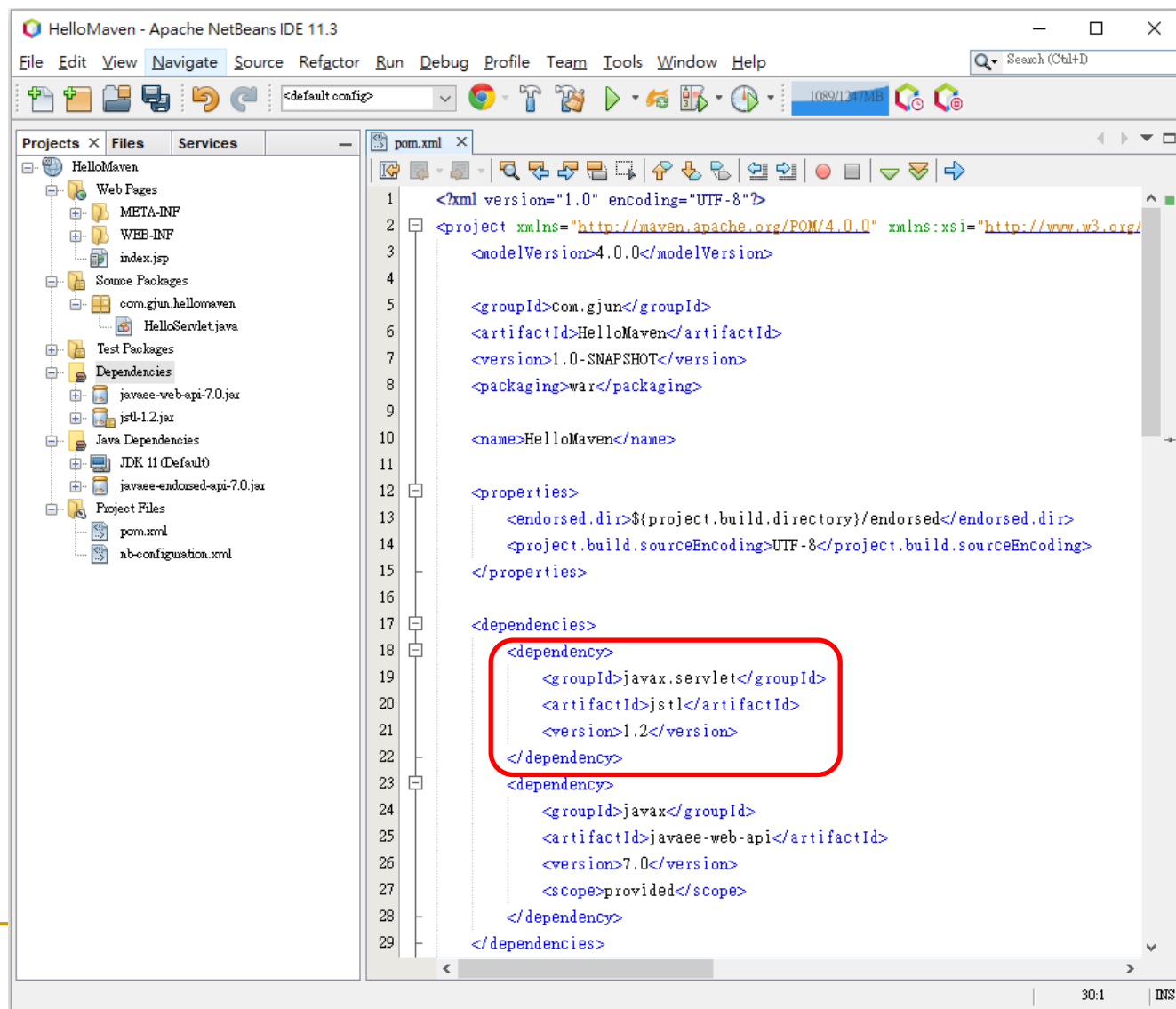
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.gjun</groupId>
  <artifactId>HelloMaven</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>HelloMaven</name>
  <properties>
    <endorsed.dir>${project.build.directory}/endorsed</endorsed.dir>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>javax</groupId>
      <artifactId>javaxee-web-api</artifactId>
      <version>7.0</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
</project>
```

A context menu is open over the "Dependencies" folder in the left sidebar, with the "Add Dependency..." option selected. An "Add Dependency" dialog box is also open, showing the following details:

- Group ID: javax.servlet
- Artifact ID: jstl
- Version: 1.2
- Scope: compile

The dialog box includes a "Search" tab and a "Query" field. The "Add" button is highlighted.

# 新增JSTL 相依性



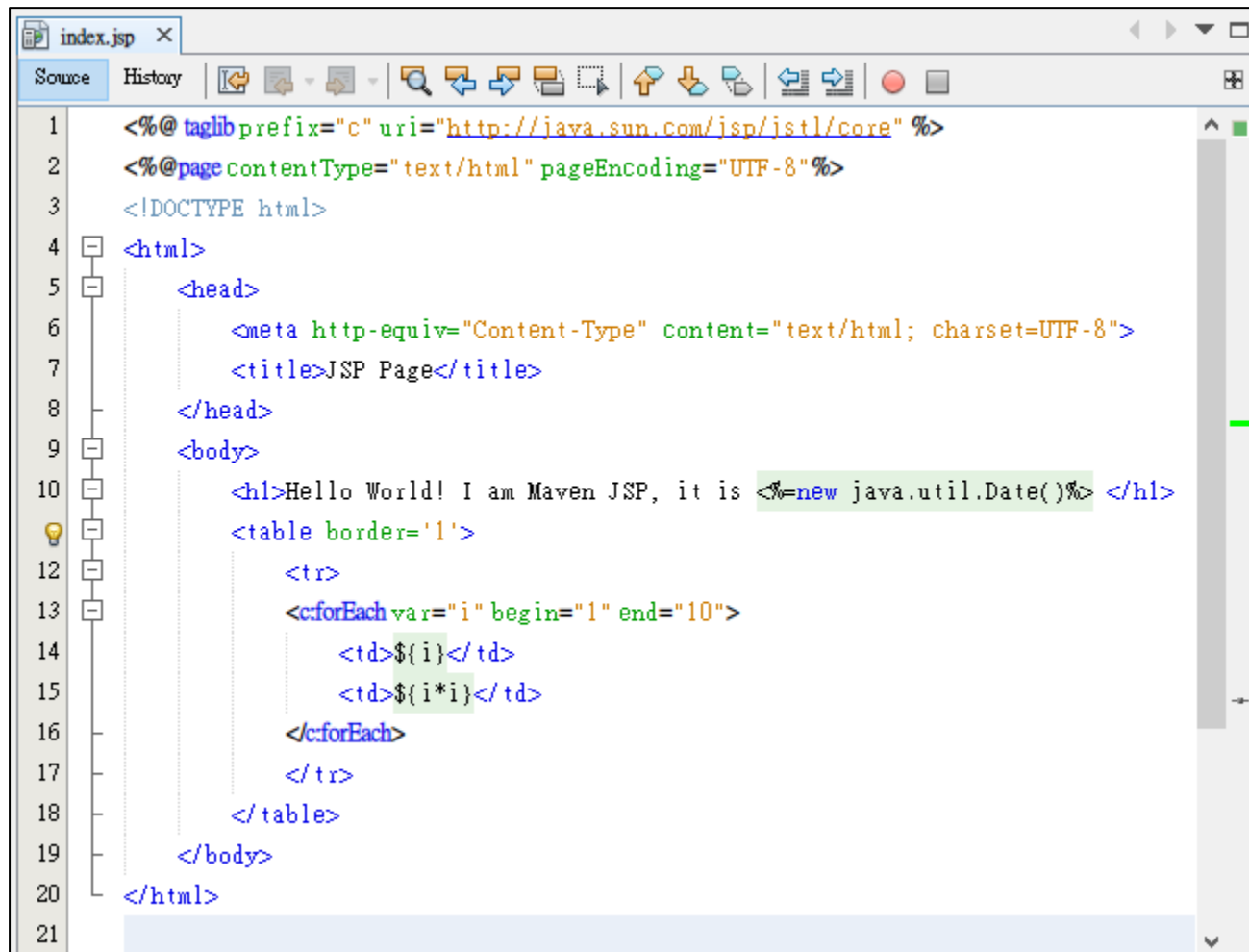
# 修改 plug-in

- maven-war-plugin
  - 3.1.0
- maven-dependency-plugin
  - 3.1.0

```
31 <build>
32   <plugins>
33     <plugin>
34       <groupId>org.apache.maven.plugins</groupId>
35       <artifactId>maven-compiler-plugin</artifactId>
36       <version>3.1</version>
37       <configuration>
38         <source>1.7</source>
39         <target>1.7</target>
40         <compilerArguments>
41           <endorseddirs>${endorsed.dir}</endorseddirs>
42         </compilerArguments>
43       </configuration>
44     </plugin>
45     <plugin>
46       <groupId>org.apache.maven.plugins</groupId>
47       <artifactId>maven-war-plugin</artifactId>
48       <version>3.1.0</version>
49       <configuration>
50         <failOnMissingWebXml>>false</failOnMissingWebXml>
51       </configuration>
52     </plugin>
53     <plugin>
54       <groupId>org.apache.maven.plugins</groupId>
55       <artifactId>maven-dependency-plugin</artifactId>
56       <version>3.1.0</version>
57       <executions>
58         <execution>
59           <id>default</id>
60           <goals>
61             <goal>copy</goal>
62           </goals>
63         </execution>
64       </executions>
65     </plugin>
66   </plugins>
67 </build>
```

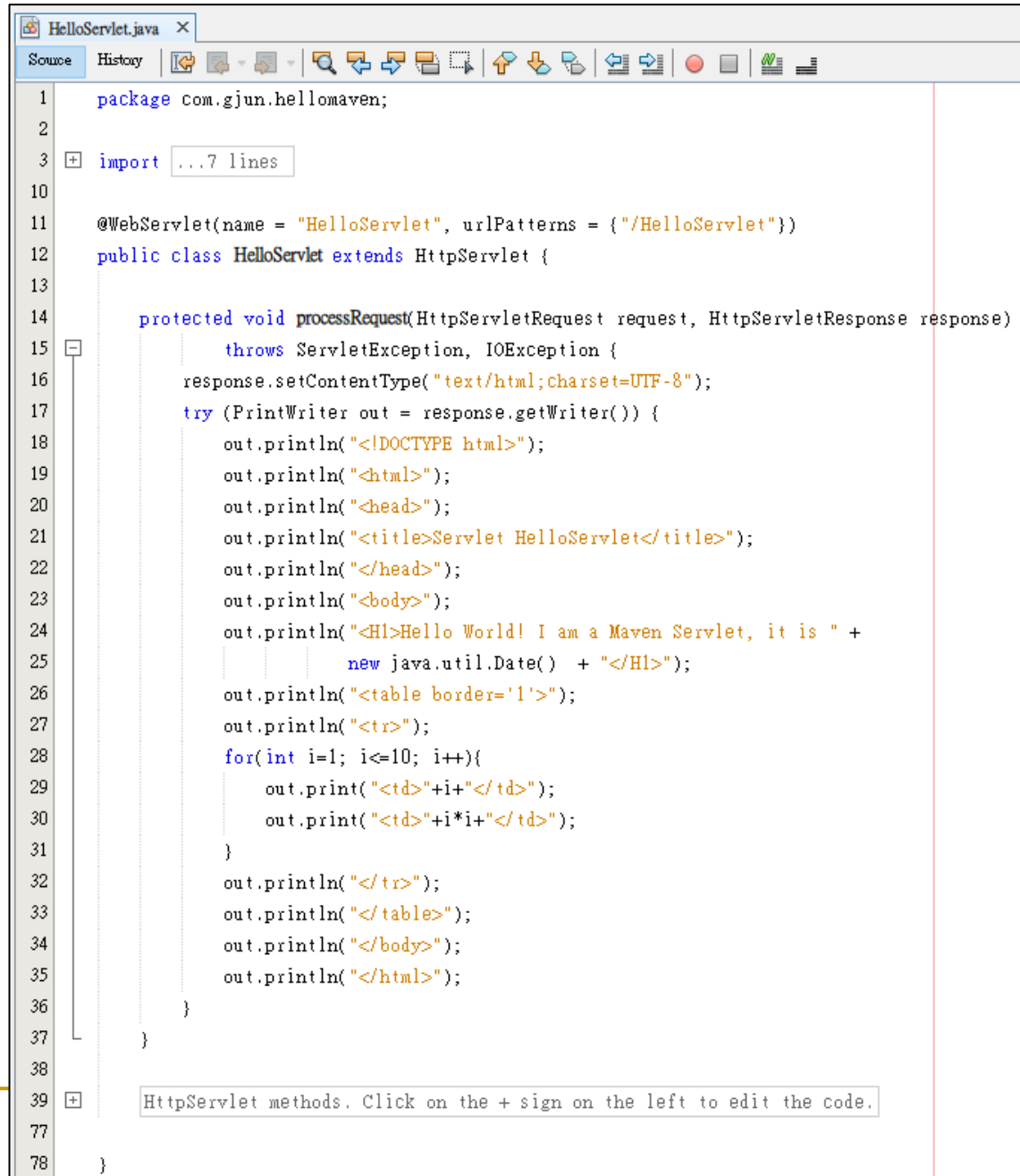


# index.jsp



```
1 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
2 <%@ page contentType="text/html" pageEncoding="UTF-8" %>
3 <!DOCTYPE html>
4 <html>
5   <head>
6     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7     <title>JSP Page</title>
8   </head>
9   <body>
10    <h1>Hello World! I am Maven JSP, it is <%=new java.util.Date()%> </h1>
11    <table border='1'>
12      <tr>
13        <c:forEach var="i" begin="1" end="10">
14          <td>${i}</td>
15          <td>${i*i}</td>
16        </c:forEach>
17      </tr>
18    </table>
19  </body>
20 </html>
21
```

# HelloServlet



```
1 package com.gjun.hellomaven;
2
3 import ...7 lines
4
10
11 @WebServlet(name = "HelloServlet", urlPatterns = {"/HelloServlet"})
12 public class HelloServlet extends HttpServlet {
13
14     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
15         throws ServletException, IOException {
16         response.setContentType("text/html;charset=UTF-8");
17         try (PrintWriter out = response.getWriter()) {
18             out.println("<!DOCTYPE html>");
19             out.println("<html>");
20             out.println("<head>");
21             out.println("<title>Servlet HelloServlet</title>");
22             out.println("</head>");
23             out.println("<body>");
24             out.println("<H1>Hello World! I am a Maven Servlet, it is " +
25                 new java.util.Date() + "</H1>");
26             out.println("<table border='1'>");
27             out.println("<tr>");
28             for(int i=1; i<=10; i++){
29                 out.print("<td>"+i+"</td>");
30                 out.print("<td>"+i*i+"</td>");
31             }
32             out.println("</tr>");
33             out.println("</table>");
34             out.println("</body>");
35             out.println("</html>");
36         }
37     }
38
39     HttpServlet methods. Click on the + sign on the left to edit the code.
40
77
78 }
```

# Maven 清理及建構

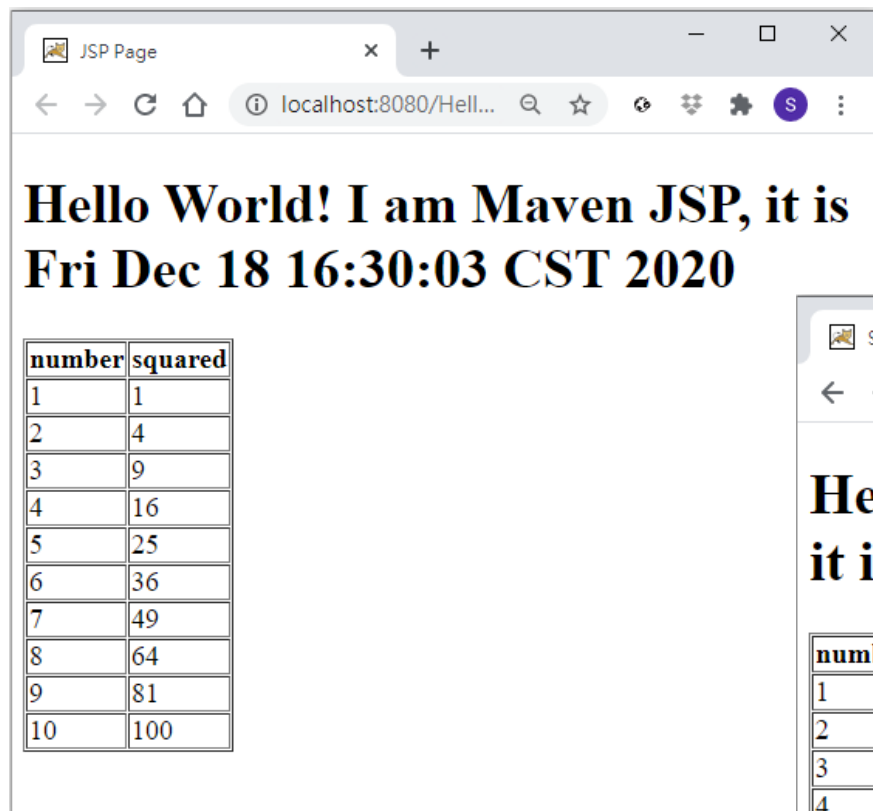
The screenshot displays the Apache NetBeans IDE 11.3 interface. The left-hand 'Projects' pane shows the 'HelloMaven' project structure, including 'Web Pages', 'META-INF', 'WEB-INF', 'Source Packages', 'Test Packages', 'Dependencies', 'Java Dependencies', 'JDK 11 (Default)', 'javaee-endorsed-api-7.0.jar', and 'Project Files'. The middle pane shows the 'pom.xml' file with the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <groupId>com.gun.hellomaven</groupId>
  <artifactId>hellomaven</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>HelloMaven</name>
  <url>http://java.sun.com/jsp/jstl/core</url>
  <dependencies>
    <dependency>
      <groupId>jstl</groupId>
      <artifactId>jstl</artifactId>
      <version>1.2</version>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-war-plugin</artifactId>
        <version>2.4</version>
      </plugin>
    </plugins>
  </build>
</project>
```

The right-hand 'Output' window shows the build log for 'Build (HelloMaven)'. The log indicates that the war is being built at 'D:\JavaEE\Projects\HelloMaven\target\hellomaven-1.0-SNAPSHOT.war'. The build process includes installing the Maven plugin and the project dependencies. The final output is:

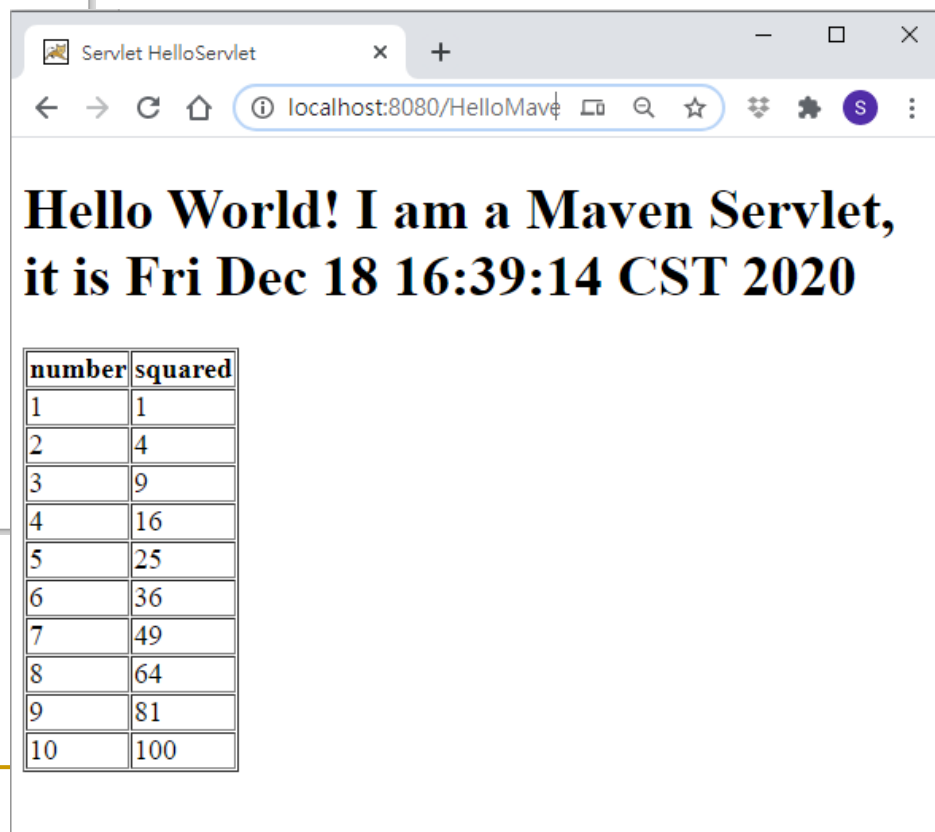
```
Building war: D:\JavaEE\Projects\HelloMaven\target\hellomaven-1.0-SNAPSHOT.war
--- maven-install-plugin:2.4:install (default-install) ---
Installing D:\JavaEE\Projects\HelloMaven\target\hellomaven-1.0-SNAPSHOT.war to D:\JavaEE\Projects\HelloMaven\target\hellomaven-1.0-SNAPSHOT.war
Installing D:\JavaEE\Projects\HelloMaven\pom.xml to D:\JavaEE\Projects\HelloMaven\target\hellomaven-1.0-SNAPSHOT.war
---
BUILD SUCCESS
---
Total time: 2.472 s
Finished at: 2020-12-18T16:32:51+08:00
Final Memory: 17M/70M
```

# 測試執行



**Hello World! I am Maven JSP, it is  
Fri Dec 18 16:30:03 CST 2020**

number	squared
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100



**Hello World! I am a Maven Servlet,  
it is Fri Dec 18 16:39:14 CST 2020**

number	squared
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	100

# Lab 1 Maven DVD Library

- 建立 DVDMaven專案
- 複製JSP檔案至Web Pages
  - index.jsp / list\_library.jsp / add\_dvd.jsp / success.jsp / error.jsp / set\_pref.jsp
- 複製Java檔案至Source Packages
  - controller 套件
    - AddDVDServlet.java / SetPrefsServlet.java
  - model類別
    - DVDItem.java
  - web套件
    - InitializeLibrary.java

# Lab 1 Maven DVD Library

- 新增web.xml
  - 設定libraryFile初始參數為/WEB-INF/library.txt
  - 設定genre-list初始參數為Animation, Action, Sci-Fi
- 複製library.txt至 /WEB-INF/
- 修改pom.xml
  - 新增JSTL 相依性
- Maven 清理及建構
- 測試、執行

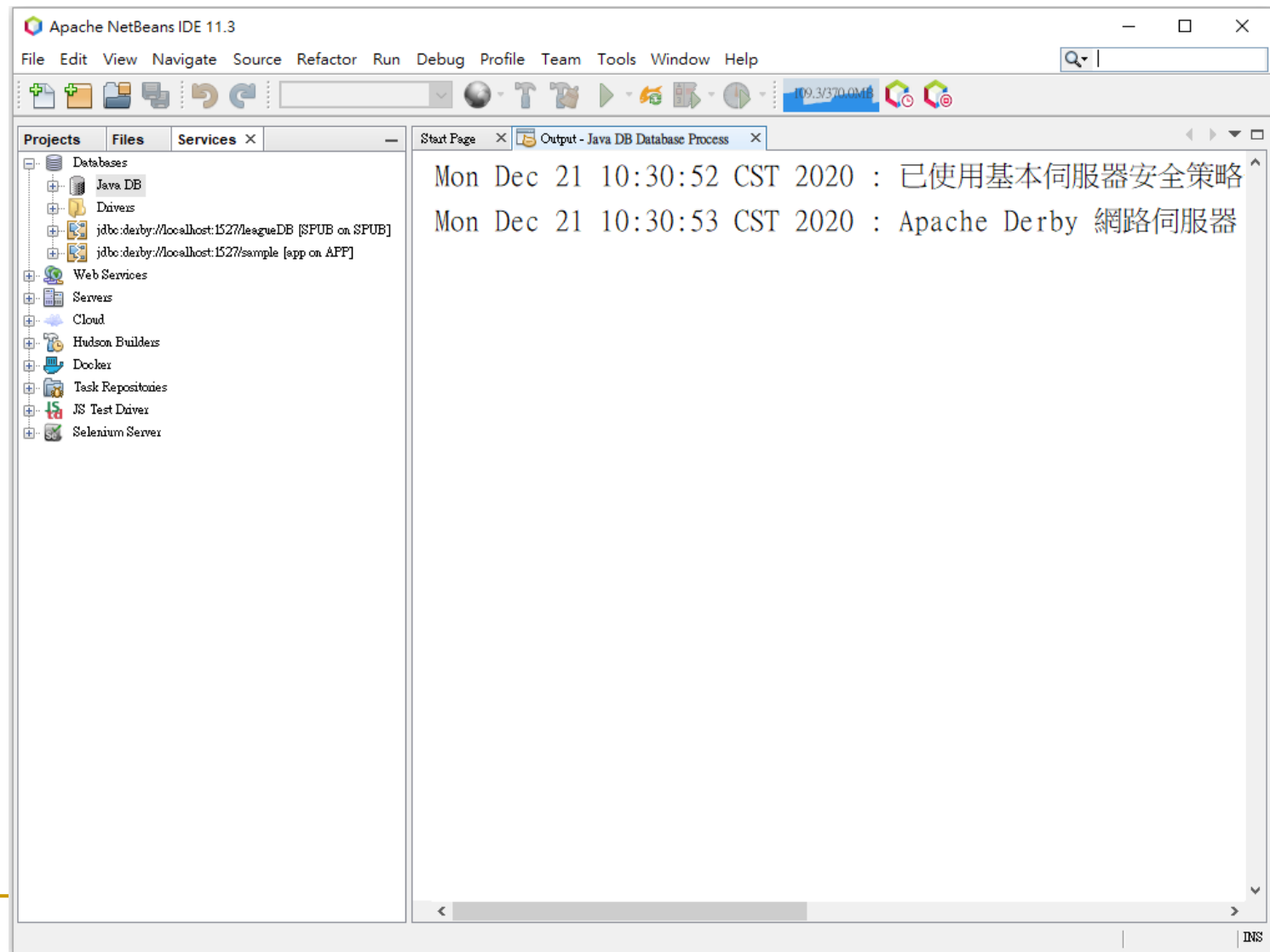
# 課程大綱

1) Maven

2) **Java Persistence API**

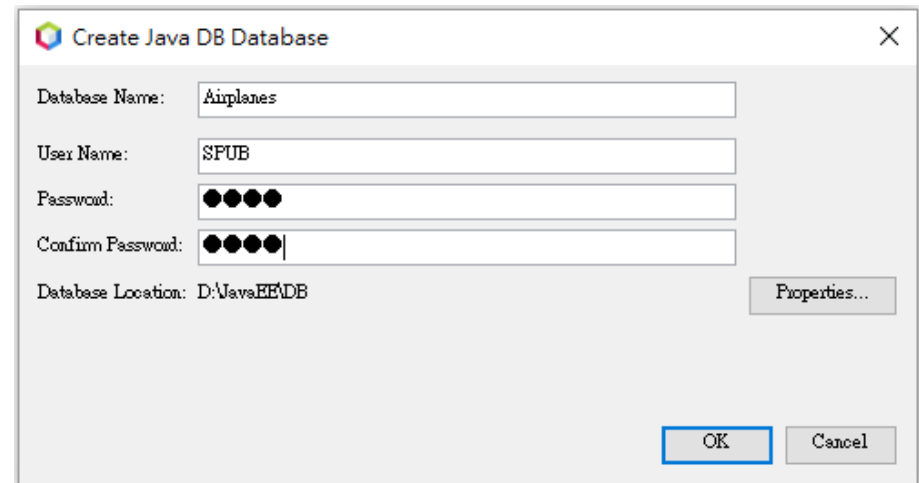
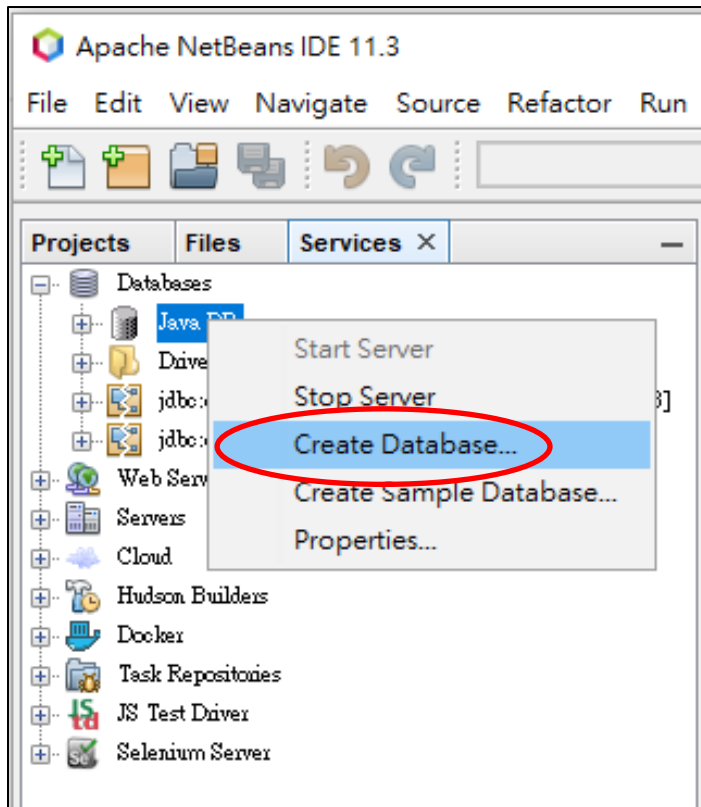
- ❑ **derby 資料庫設定**
- ❑ **ORM 設定**
- ❑ **Java Persistence API 開發**

# 設定 JavaDB Derby 資料庫



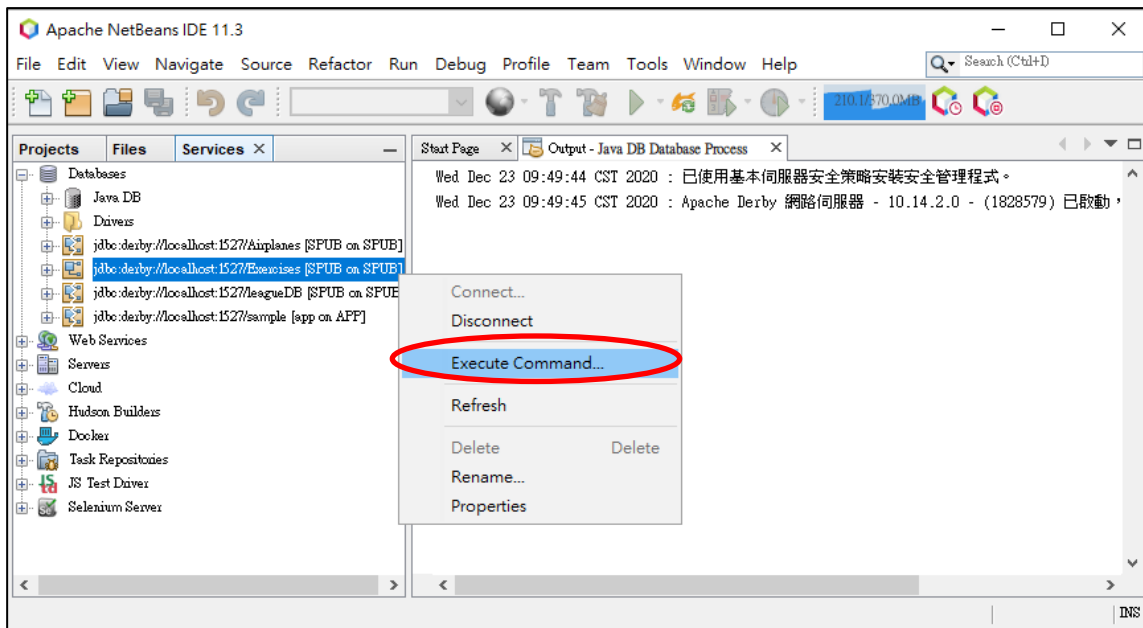


# 建立Airplanes資料庫



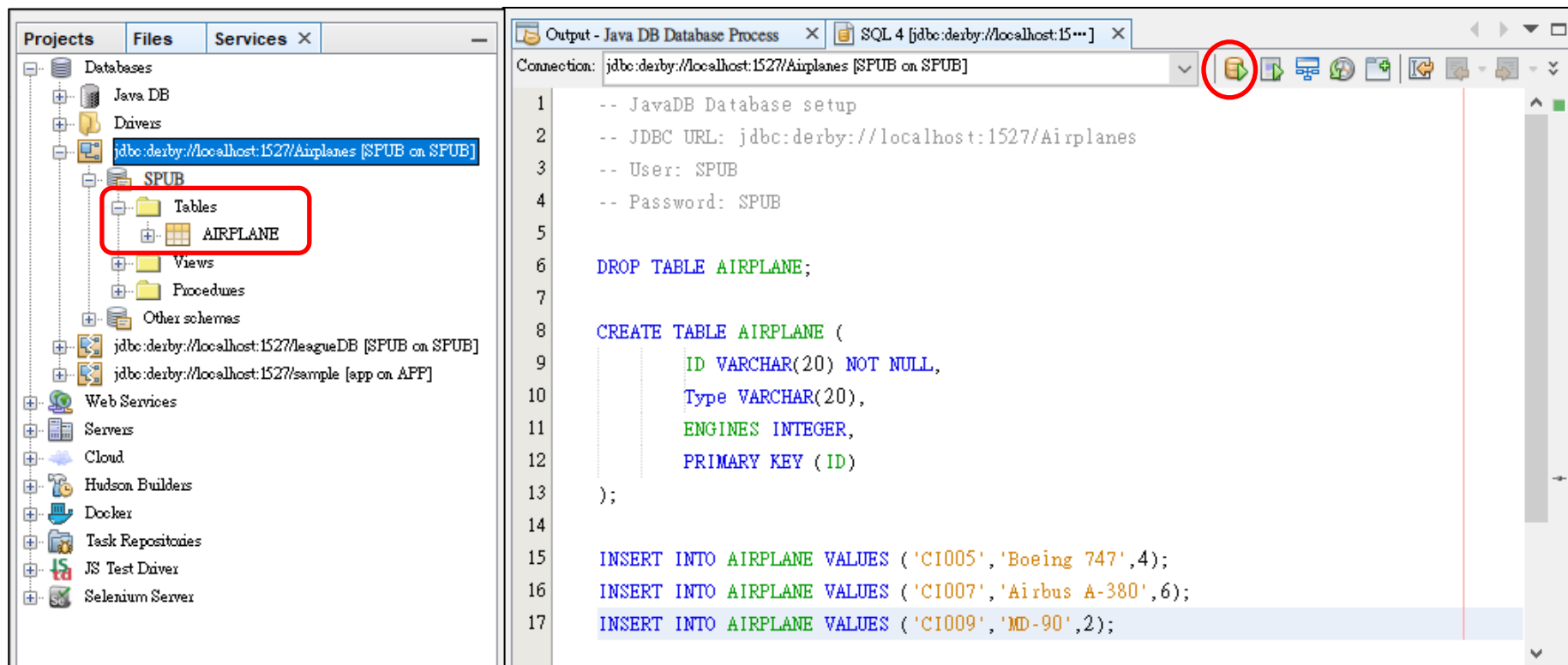
User Name : SPUB  
Password: SPUB

# 建立資料表



# 建立AIRPLANE資料表

- 複製airplanes.sql
  - 建立AIRPLANE資料表



# 檢視AIRPLANE資料表

The screenshot displays a database management interface. On the left, a tree view shows the project structure under 'Services'. The 'AIRPLANE' table is selected, and a context menu is open with options like 'View Data...', 'Execute Command...', 'Add Column...', 'Refresh', 'Delete', 'Grab Structure...', 'Recreate Table...', and 'Properties'. The 'View Data...' option is highlighted.

The right pane shows the SQL editor with the query: `SELECT * FROM SPUB.AIRPLANE FETCH FIRST 100 ROWS ONLY;`. Below the editor, the results are displayed in a table format. The table has columns: #, ID, TYPE, and ENGINES. The data shows three rows: 1 (CI005, Boeing 747, 4), 2 (CI007, Airbus A-380, 6), and 3 (CI009, MD-90, 2).

#	ID	TYPE	ENGINES
1	CI005	Boeing 747	4
2	CI007	Airbus A-380	6
3	CI009	MD-90	2

# Object Relational Mapping

## ■ Problem

- ❑ Java程式運算是以物件模型運作
- ❑ 資料永續儲存是以關聯式資料庫表格處理
- ❑ 需要撰寫相當多程式碼來做兩者的更新及同步

## ■ Solution

- ❑ 物件與關連式資料映射(Object Relational Mapping)
  - 將 **Java** 中物件資訊，直接映射至關聯式資料庫中的表格
  - 使用宣告式的映射指令，取代手寫將資料儲存至關連式資料庫的程式碼及**SQL** 命令
  - 簡化應用程式在不同資料庫廠商之間移植作業

# Java Persistence API

- Object Relational Mapping Software解決方案
  - JBoss Hibernate
  - Oracle TopLink(Eclipse Link)
- Java Persistence API (JPA)
  - Java根據前述方案的經驗，制訂官方永續儲存標準
  - javax.persistence 套件
    - 物件的資料儲存至關聯式資料庫的標準介面
  - 底層可以使用不同廠商的ORMS實作
    - NetBeans：預設的底層實作為Eclipse Link
    - JBoss的工具：預設的其底層實作則為Hibernate

# 新增ORM相依性

**Add Dependency**

Group ID:

Artifact ID:

Version:  Scope:

Type:  Classifier:

Search:

Query:

(coordinate, class name, project name...)

Search Results:

**Add Dependency**

Group ID:

Artifact ID:

Version:  Scope:

Type:  Classifier:

Search:

Query:

(coordinate, class name, project name...)

Search Results:

Airplanes - Apache NetBeans IDE 11.3

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help Search (Ctrl+D)

<default config>

438.4/627.0MB

Projects Files Services

Airplanes

- Web Pages
  - META-INF
    - context.xml
  - WEB-INF
    - ExceptionHandler.jsp
    - Listview.jsp
    - index.jsp
  - RESTful Web Services
  - Source Packages
    - entity
    - web
  - Dependencies
    - derbyclient-10.14.2.0.jar
    - eclipselink-2.7.7.jar
    - javaee-web-api-7.0.jar
    - commonj.sdo-2.1.1.jar
    - jakarta.persistence-2.2.3.jar
  - Java Dependencies
    - JDK 11 (Default)
    - javaee-endowed-api-7.0.jar
  - Project Files
    - pom.xml
    - nb-configuration.xml

Output pom.xml

```
16
17
18 <dependencies>
19     <dependency>
20         <groupId>org.apache.derby</groupId>
21         <artifactId>derbyclient</artifactId>
22         <version>10.14.2.0</version>
23     </dependency>
24     <dependency>
25         <groupId>org.eclipse.persistence</groupId>
26         <artifactId>eclipselink</artifactId>
27         <version>2.7.7</version>
28     </dependency>
29     <dependency>
30         <groupId>javax</groupId>
31         <artifactId>javaee-web-api</artifactId>
32         <version>7.0</version>
33         <scope>provided</scope>
34     </dependency>
35 </dependencies>
```

37.21 INS

# JSTL 相依性

Add Dependency

Group ID:

Artifact ID:

Version:  Scope:

Type:  Classifier:

Search Open Projects Dependency Management

Query:

(coordinate, class name, project name...)

Search Results:

```
17 <dependencies>
18   <dependency>
19     <groupId>org.apache.derby</groupId>
20     <artifactId>derbyclient</artifactId>
21     <version>10.14.2.0</version>
22   </dependency>
23   <dependency>
24     <groupId>org.eclipse.persistence</groupId>
25     <artifactId>eclipselink</artifactId>
26     <version>2.7.5</version>
27   </dependency>
28   <dependency>
29     <groupId>javax.servlet</groupId>
30     <artifactId>jstl</artifactId>
31     <version>1.2</version>
32   </dependency>
33   <dependency>
34     <groupId>javax</groupId>
35     <artifactId>javaee-web-api</artifactId>
36     <version>7.0</version>
37     <scope>provided</scope>
38   </dependency>
39 </dependencies>
```

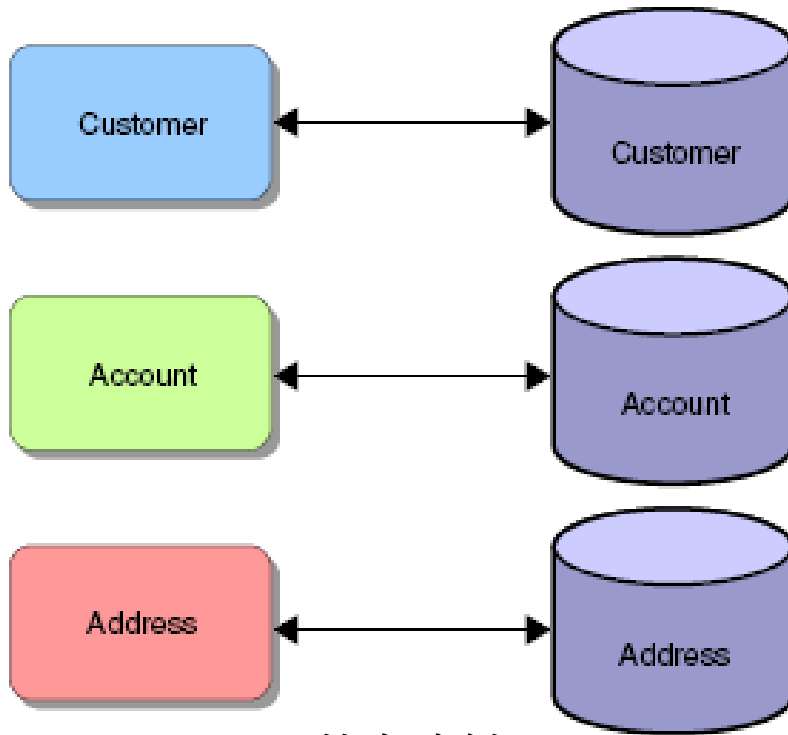
Projects X Files Services

Airplanes

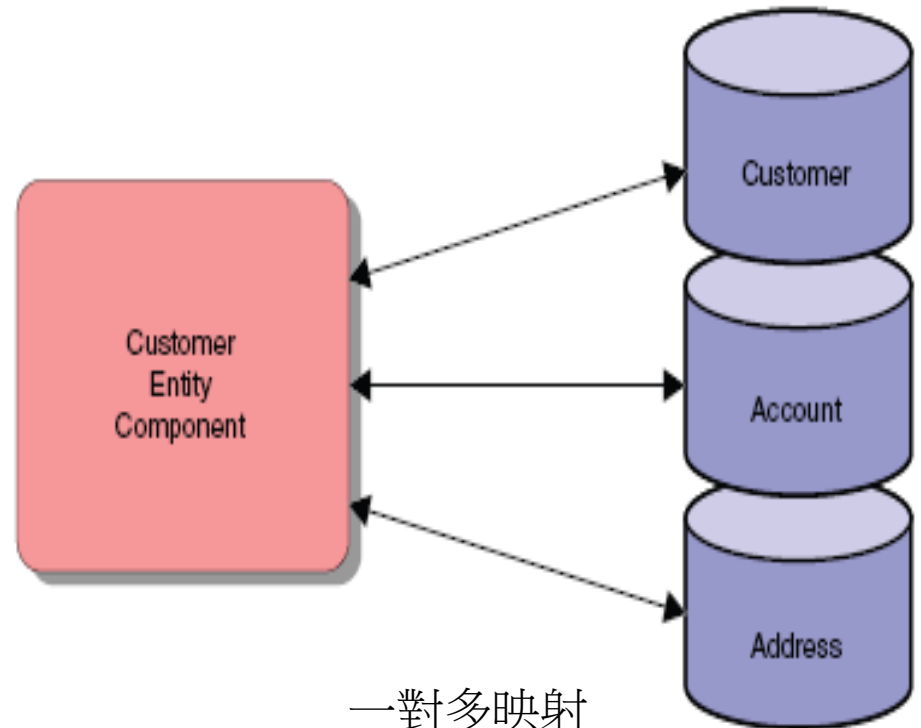
- Web Pages
  - META-INF
    - context.xml
  - WEB-INF
    - web.xml
    - ExceptionHandler.jsp
    - ListView.jsp
    - index.jsp
- RESTful Web Services
- Source Packages
  - entity
    - Airplane.java
  - web
    - AddServlet.java
    - ListServlet.java
- Test Packages
- Other Sources
  - src/main/resources
    - META-INF
      - persistence.xml
- Dependencies
  - derbyclient-10.14.2.0.jar
  - eclipselink-2.7.5.jar
  - javaee-web-api-7.0.jar
  - jstl-1.2.jar
  - commonj.sdo-2.1.1.jar
  - jakarta.persistence-2.2.3.jar
- Java Dependencies
  - JDK 11 (Default)
  - javaee-endorsed-api-7.0.jar
- Project Files
  - pom.xml
  - nb-configuration.xml



# Object Relational映射方式



基本映射

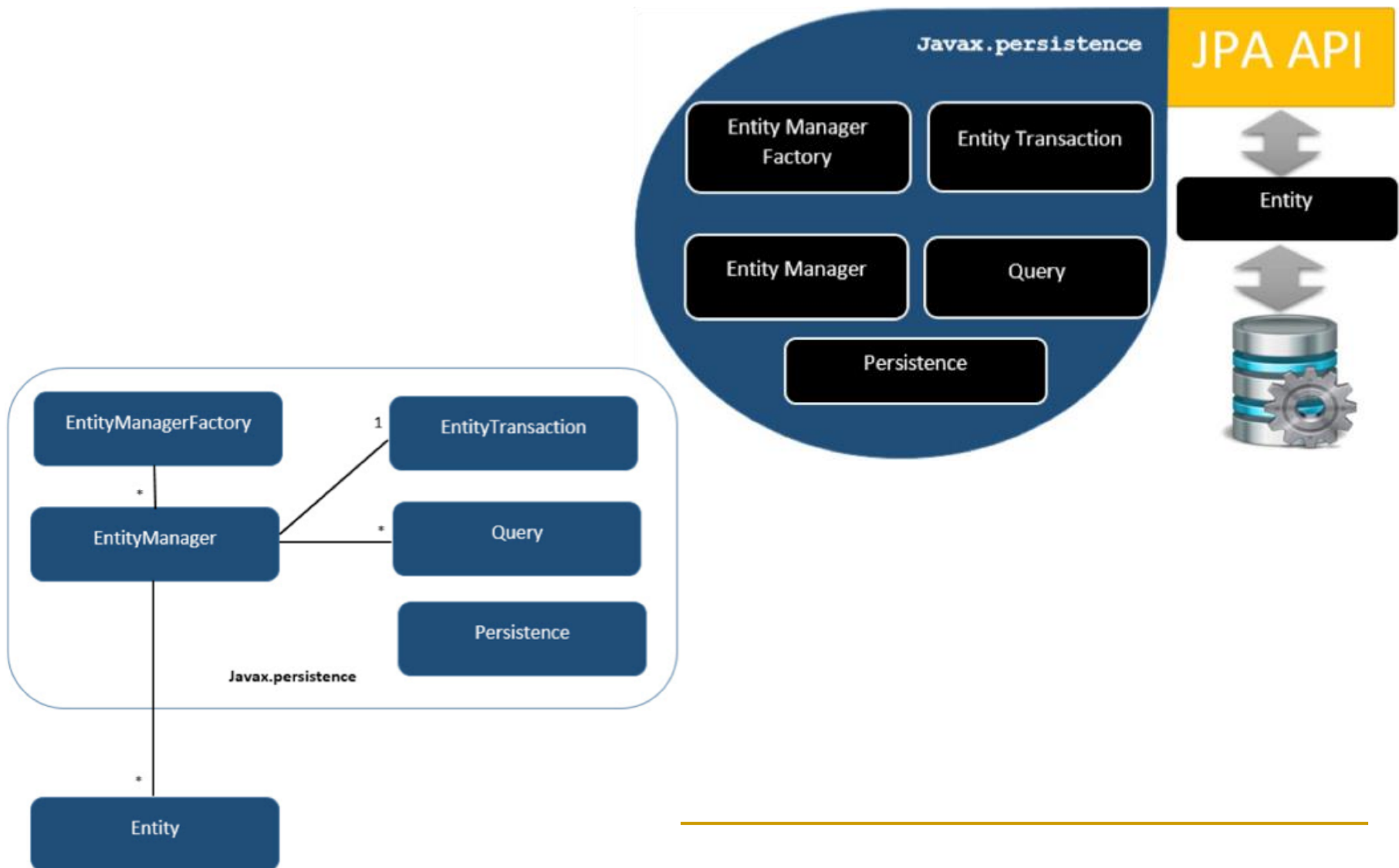


一對多映射

# Java Persistence API 架構

名稱	描述
Persistence Unit	連結指定永續性單元 靜態方法用來取得EntityManagerFactory
Entity	永續性物件實體 代表儲存在 Database 中的 Table
EntityManagerFactory	EntityManager的工廠類別 建構並管理多個EntityManager實體
EntityManager	管理的永續性單元操作的介面 工作原理類似工廠的查詢物件
Query	控制資料庫查詢操作的介面 JPA供應商提供實作物件，可執行JPQL查詢
EntityTransaction	控制資料庫交易操作的介面 與EntityManager是一對一的對應關係

# Java Persistence API 架構



# Java Persistence API 架構

- Persistence Unit 永續性單元
  - 數個Entity 類別及一個persistence.xml組成
  - 通常打包為EJB-JAR, WAR 或 JAR檔
- Entity類別
  - 程式設計師定義
    - 代表物件導向觀點的商業領域物件
  - 建立資料表欄位與Entity類別屬性對應關係
    - Entity類別之屬性前加上Annotations標註

# Java Persistence API 架構

## ■ Persistence.xml

### □ Persistence Unit的設定檔

- 組成Entity類別
- DataSource設定
  - URL/帳號/密碼/Driver
  - JNDI DataSource

## ■ Entity Manager

- 管理Entity實體生命週期的服務元件
- 在JPA架構下用來操作新增、查詢、修改及刪除資料庫資料的物件

# Java Persistence API 架構

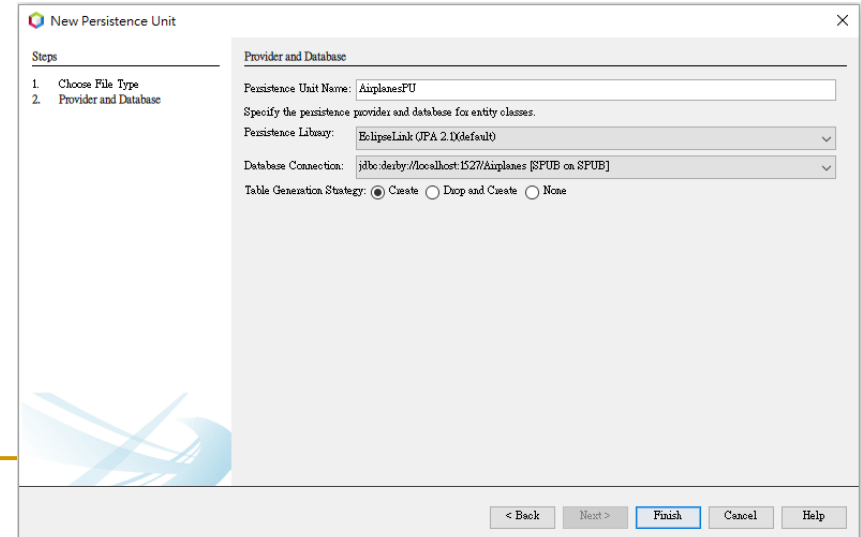
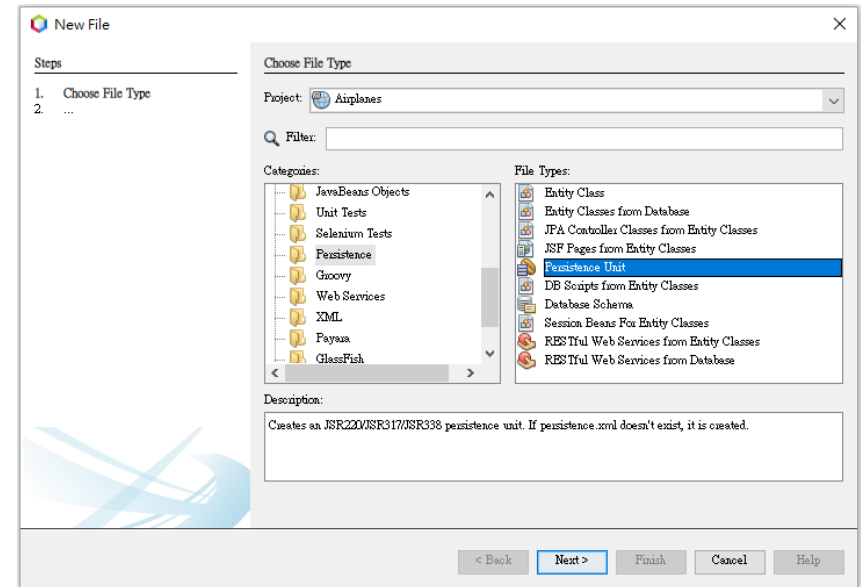
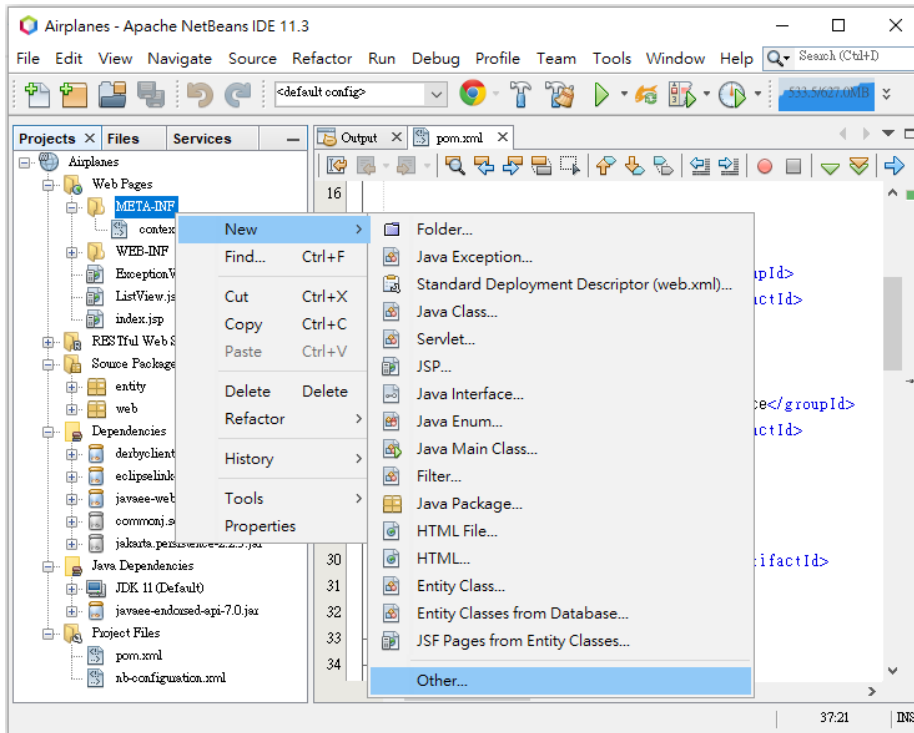
## ■ Persistence Context 永續性實體

- 永續性單元(Persistence Unit)的工作實體
  - 每一個EntityManager取得一份永續性實體
- 生命週期通常持續資料交易範圍
  - 由 EntityManager 管理
- 多個不同Persistence ID 的Entity物件組成

## ■ Persistence Identity

- 網路元件容器用來映射Entity物件至資料表一筆對應資料的id值

# 新增persistence.xml



# META-INF/persistence.xml

## ■ 用途

- 設定資料庫連結與底層實作的一些細節
  - JDBC URL、Driver、使用者名稱、密碼等資訊
  - 指定容器端管理的Data Source之JNDI名稱
- 須先在容器端設定JDBC與Connection Pool等資源

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence.xml 2.1">
3   <persistence-unit name="AirplanesPU" transaction-type="RESOURCE_LOCAL">
4     <exclude-unlisted-classes>false</exclude-unlisted-classes>
5     <properties>
6       <property name="javax.persistence.jdbc.url" value="jdbc:derby://localhost:1527/Airplanes"/>
7       <property name="javax.persistence.jdbc.user" value="SPUB"/>
8       <property name="javax.persistence.jdbc.driver" value="org.apache.derby.jdbc.ClientDriver"/>
9       <property name="javax.persistence.jdbc.password" value="SPUB"/>
10      <property name="javax.persistence.schema-generation.database.action" value="create"/>
11    </properties>
12  </persistence-unit>
13 </persistence>
```



# Entity 類別規範

- 撰寫 Entity 類別
  - 需宣告為 **public** 類別
  - 可以是 **abstract** 類別
  - 不可為 **final** 類別, 也不可以有**final** 方法
  - 不可為巢狀**inner**類別**Class**
  - 實作**Serializable**介面
  - 要有**public**或**protected**的無參數建構子, 或預設建構子
  - 需加上下列標註
    - **@Entity** 標註此類別為JPA的Entity類別
    - **@Table (name = *TableName*)**標註Entity類別對應的資料表格
      - 類別名稱與表格名稱相同, 則可以省略

# Entity 類別規範

## ■ Entity 類別屬性欄位

- 屬性不可為 `public`
- 可宣告需要的存取方法或其他商業邏輯方法
  - 不可改寫 `finalize()` 方法
- 可使用下列標註
  - `@Id` 屬性對應資料表的主鍵
  - `@GeneratedValue(strategy = GenerationType.AUTO)` 定義主鍵由ORMS自動選擇最適產生策略
  - `@Column(name = ColumnName)` 定義屬性對應之資料表欄位
    - 屬性名稱與資料表欄位名稱相同時可以省略

# 驗證和覆蓋預設映射關係

## ■ 物件與關連資料庫映射關係

物件層元件	資料層元件
Entity 類別	資料表
Entity 類別屬性欄位	資料表欄位
Entity 物件實體	資料表一筆資料(一行 row)

- 預設資料表名稱與Entity類別名稱相同
  - **@Table (name = TableName)** 修改Entity類別對應的資料表格
- 預設資料表欄位名稱與Entity類別屬性名稱相同
  - **@Column(name = ColumnName)** 修改屬性對應之資料表欄位名稱

# 主鍵值 Primary Key 產生

- Entity類別必須有一唯一的識別屬性
  - 對應資料表的主鍵
  - **@Id** 宣告此屬性
  - **@GeneratedValue(strategy = GenerationType.AUTO)**  
定義主鍵值由ORMS自動選擇最適產生策略
    - **GenerationType.AUTO** : Entity類別主鍵值產生策略由ORMS實作自行決定
      - Oracle Toplink 實作,會以SEQUENCE方式實作,在資料庫中增加一個Sequence資料表
    - **GenerationType.IDENTITY** : ORMS實作必須將資料表之主鍵值欄位指定為identity 欄位.
    - **GenerationType.SEQUENCE** : ORMS實作必須在資料庫中以一序號表來產生資料表主鍵值.
    - **GenerationType.TABLE** : ORMS實作必須在資料庫中使用一資料表來確保其他資料表主鍵值的唯一性.

# 新增 Entity

New Entity Class

Steps

1. Choose File Type
2. Name and Location

Name and Location

Class Name:

Project:

Location:

Package:

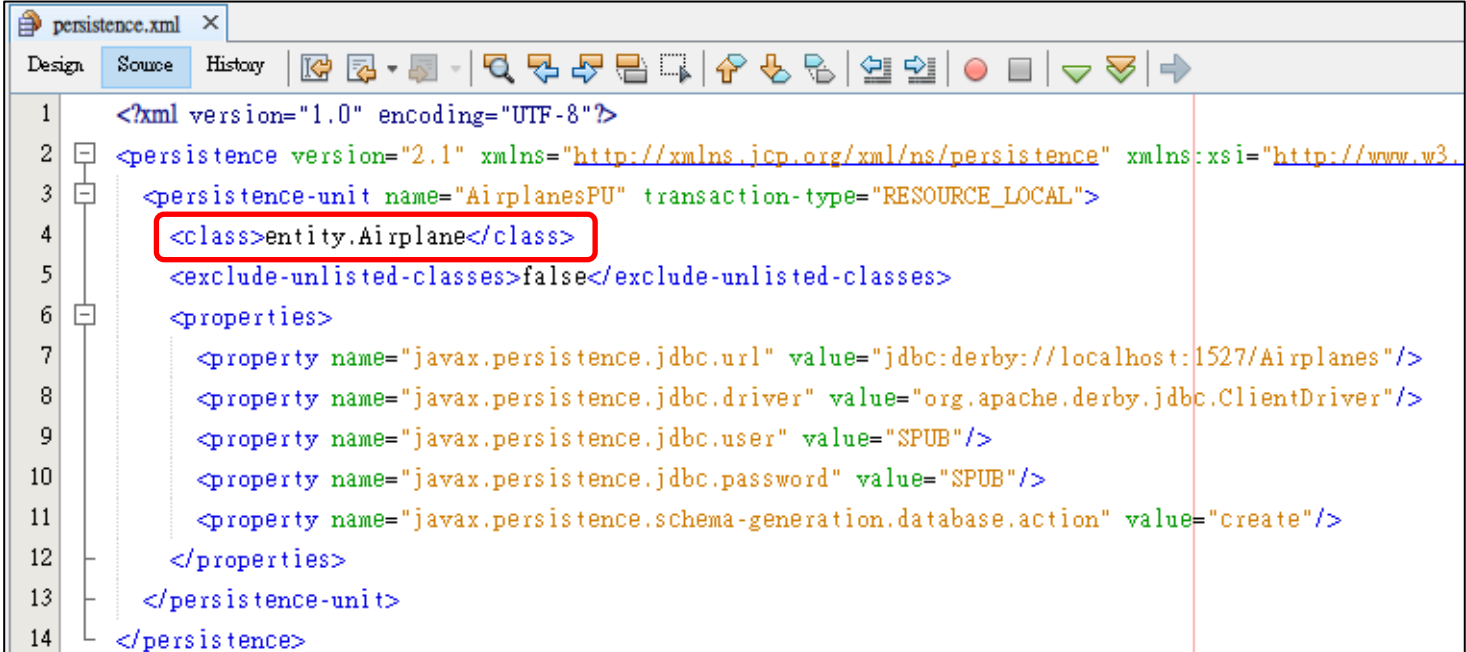
Created File:

Primary Key Type:

< Back Next > Finish Cancel Help

```
Airplane.java
Source History
1 package entity;
2
3 import ...6 lines
9
10 @Entity
11 public class Airplane implements Serializable {
12
13     private static final long serialVersionUID = 1L;
14     @Id
15     @GeneratedValue(strategy = GenerationType.AUTO)
16     private String id;
17     @Column private String type;
18     @Column private int engines;
19
20     public String getId() {...3 lines }
23     public void setId(String id) {...3 lines }
26
27     public String getType() {...3 lines }
30     public void setType(String type) {...3 lines }
33
34     public int getEngines() {...3 lines }
37     public void setEngines(int engines) {...3 lines }
40
41     @Override
42     public int hashCode() {...5 lines }
47     @Override
48     public boolean equals(Object object) {...10 lines }
58     @Override
59     public String toString() {...3 lines }
62
63 }
```

# persistence.xml



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.
3 <persistence-unit name="AirplanesPU" transaction-type="RESOURCE_LOCAL">
4   <class>entity.Airplane</class>
5   <exclude-unlisted-classes>false</exclude-unlisted-classes>
6   <properties>
7     <property name="javax.persistence.jdbc.url" value="jdbc:derby://localhost:1527/Airplanes"/>
8     <property name="javax.persistence.jdbc.driver" value="org.apache.derby.jdbc.ClientDriver"/>
9     <property name="javax.persistence.jdbc.user" value="SPUB"/>
10    <property name="javax.persistence.jdbc.password" value="SPUB"/>
11    <property name="javax.persistence.schema-generation.database.action" value="create"/>
12  </properties>
13 </persistence-unit>
14 </persistence>
```

# 宣告 Entity 類別

#	ID	Type	Engines
1	CI005	Boeing 747	4
2	CI007	Airbus A-380	6
3	CI009	MD-90	2
4	CI011	Boeing 737	2
5	CI013	Airbus A-300	4

```
package entity;
```

```
import java.io.Serializable;  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Id;
```

```
@Entity
```

```
public class Airplane implements Serializable {  
    private static final long serialVersionUID = 1L;  
    @Id  
    @GeneratedValue(strategy = GenerationType.AUTO)  
    private String id;  
    @Column private String type;  
    @Column private int engines;
```

# 宣告 Entity 類別

```
public String getId() {  
    return id;  
}  
  
public void setId(String id) {  
    this.id = id;  
}  
  
public String getType(){  
    return type;  
}  
  
public void setType(String type){  
    this.type = type;  
}  
  
public int getEngines() {  
    return engines;  
}  
  
public void setEngines(int engines){  
    this.engines = engines;  
}
```

```
@Override  
public boolean equals(Object object) {  
    boolean result = false;  
    if ((object !=null && object instanceof Airplane)) {  
        Airplane other = (Airplane) object;  
        if (this.id!=null && this.id.equals(other.id)) {  
            result = true;  
        }  
    }  
    return result;  
}  
  
@Override  
public int hashCode() {  
    int hash = 0;  
    hash += (id != null ? id.hashCode() : 0);  
    return hash;  
}  
  
@Override  
public String toString() {  
    return "entity.Airplane[id=" + id + "];"  
}  
}
```



# Persistence 類別

## ■ javax.persistence.Persistence 類別

### □ 取得 EntityManagerFactory 物件

常用方法	傳回值	說明
createEntityManagerFactory( String persistenceUnitName)	static EntityManagerFactory	取得指定PersistenceUnit 之 EntityManager 工廠的類別方法

# EntityManagerFactory 介面

## ■ javax.persistence.EntityManagerFactory 介面

### □ 取得 EntityManager 物件

常用方法	傳回值	說明
createEntityManager()	EntityManager	建立應用程式管理的EntityManager

# EntityManager 類別

- javax.persistence.EntityManager類別
  - 提供Entity物件的生命週期管理、資料表的對應、資料庫的存取等功能

常用方法	傳回值	說明
find(Class<T> entityClass, Object primaryKey)	<T>	以指定主鍵值搜尋並載入Entity實體 (Read)
createQuery(String qlStr)	Query	建立一個Query物件,用以執行Persistence查詢命令,使用Query.getResultList()取得結果 (Select)
persist(Object entity)	void	將Entity物件資料新增至對應資料表中 (Create)
merge(T entity)	<T> T	將Entity物件資料更新至對應資料表中 (Update)
remove(Object entity)	void	刪除對應資料表中Entity物件資料 (Delete)
getTransaction()	EntityTransaction	取得EntityTransaction交易物件
flush()	void	永續性實體 persistence context與對應資料庫同步

# Query 查詢介面

## ■ javax.persistence.Query 介面

### □ 控制資料庫查詢操作

常用方法	傳回值	說明
getResultList()	List	執行SELECT查詢，以無泛型的List傳回查詢結果
getSingleResult()	Object	執行SELECT查詢，傳回一個Object型態查詢結果

# EntityTransaction 交易類別

## ■ javax.transaction.EntityTransaction 類別

### □ 控制資料庫交易操作的起始與確認

常用方法	傳回值	說明
begin()	void	建立一個連結當前執行緒操作的交易
commit()	void	確認自上一次確認/回復後進行的所有變更
rollback()	Void	取消在當前交易中進行的所有更改
setRollbackOnly()	void	設定當前交易之操作的結果必定為取消變更

# JPQL 語法

- JPQL (Java Persistence Query Language)
  - 類似SQL語法
  - 查詢目標是持久化物件(Entity)
    - 非資料庫本身
  - JPQL語法分為三類：
    - 查詢 SELECT 語法
    - 更新 UPDATE / 刪除 DELETE 語法

# JPQL 查詢 SELECT 語法

## ■ 使用主鍵值查詢

- `<T> find(Class<T> entityClass, Object primaryKey)`

## ■ 使用 SELECT 語法

- SELECT 子句 FROM 字句  
[WHERE 子句]  
[GROUP BY 子句]  
[HAVING 子句]  
[ORDER BY 子句]
- 條件表示語法

類型	語法	類型	語法
範圍	[NOT] BETWEEN	相似	[NOT] LIKE
NULL	IS [NOT] NULL	空集合	IS [NOT] EMPTY
包含	[NOT] IN	存在	[NOT] EXISTS
比較	= 、 > 、 >= 、 < 、 <= 、 <>		

# JPQL 查詢範例

```
User user = em.find(User.class, 1L);
```

```
Query query = em.createQuery("SELECT user FROM User user");  
List list = query.getResultList();
```

```
Query query = em.createQuery("FROM User user", User.class);  
List list = query.getResultList();
```

```
String sql = "SELECT user FROM User user WHERE user.age = :age";  
Query query = em.createQuery(sql);  
query.setParameter("age", 30);  
List list = query.getResultList();
```

```
String sql = "SELECT user FROM User user WHERE user.age = :age";  
TypedQuery<User> query = em.createQuery(sql, User.class);  
query.setParameter("age", 30);  
User user = query.getSingleResult();
```

# JPQL 修改 / 刪除

- UPDATE 語法更新資料：

- UPDATE 子句 SET 子句  
[WHERE 子句]

```
UPDATE User u SET u.name='Sean'  
WHERE u.name='David'
```

- DELETE 語法刪除資料：

- DELETE子句 [WHERE 子句]

```
DELETE User u WHERE u.name='David'
```

# Entity 生命週期

## ■ Entity 生命週期

### □ New 狀態

- 建立Entity物件
- 還沒有與資料庫發生關聯

### □ Managed 狀態

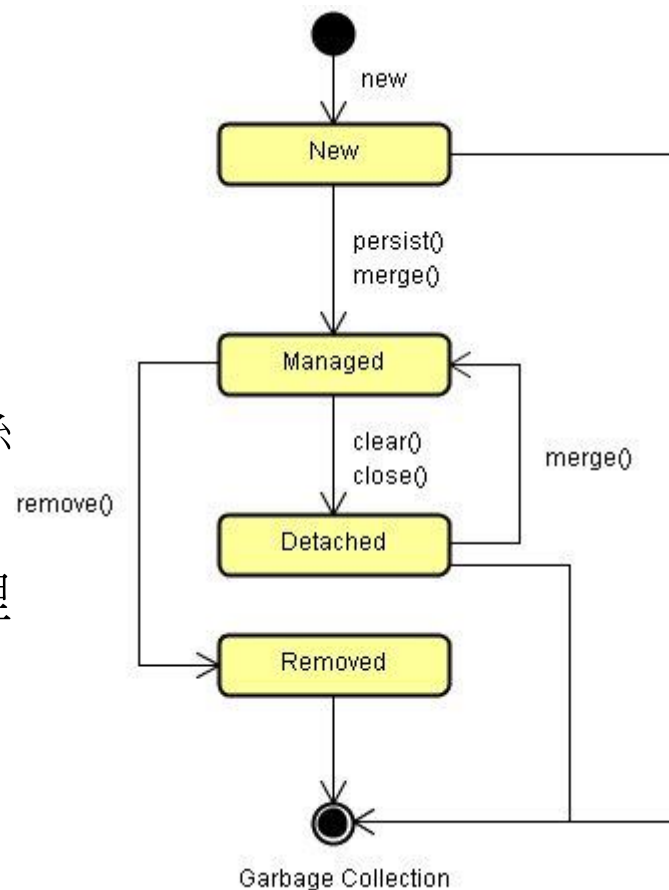
- 物件與資料庫中的資料產生對應關係

### □ Detached 狀態

- 物件脫離 EntityManager 實體的管理
- 物件將會被GC回收

### □ Removed 狀態

- 資料被刪除，物件失去對應對象
- 物件將會被GC回收





# persist() 方法

## ■ void persist(Object entity)

- 將新的物件實體插入資料庫

- 通常用於新增

- 修改資料

- 依據主鍵值到資料庫中取得物件

- `User user = em.find(User.class, id);`

- `setXXX()` 將設定資料欄位

- `user.setName(...);`

- `user.setAge(...);`

- `persist()` 修改資料

- `em.persist(user);`

# merge()方法

## ■ <T> T merge(<T> Entity)

- 依主鍵值在資料庫找到資料後，以物件內容更新欄位
- 若該物件內容不存在於資料庫則新增該筆紀錄

```
User user = new User();  
user.setName(...);  
user.setAge(...);  
em.merge(user);
```

# 操作範例

## ■ 新增資料

```
User user = new User();  
user.setId(1L);  
user.setName("Sean");  
user.setAge(new Long(30));  
EntityTransaction etx = em.getTransaction();  
etx.begin();  
em.persist(user); //em.merge(user);  
etx.commit();
```

## ■ 刪除資料

```
User user = em.find(User.class, 1L);  
EntityTransaction etx = entityManager.getTransaction();  
etx.begin();  
em.remove (user);  
etx.commit();
```

# 操作範例

## ■ 修改資料

```
User user = em.find(User.class, 1L);  
user.setName("Tom")  
user.setAge("10")  
EntityTransaction etx = entityManager.getTransaction();  
etx.begin();  
em.merge(user); //em.persist(user);  
etx.commit();
```

# Java Persistence API 範例

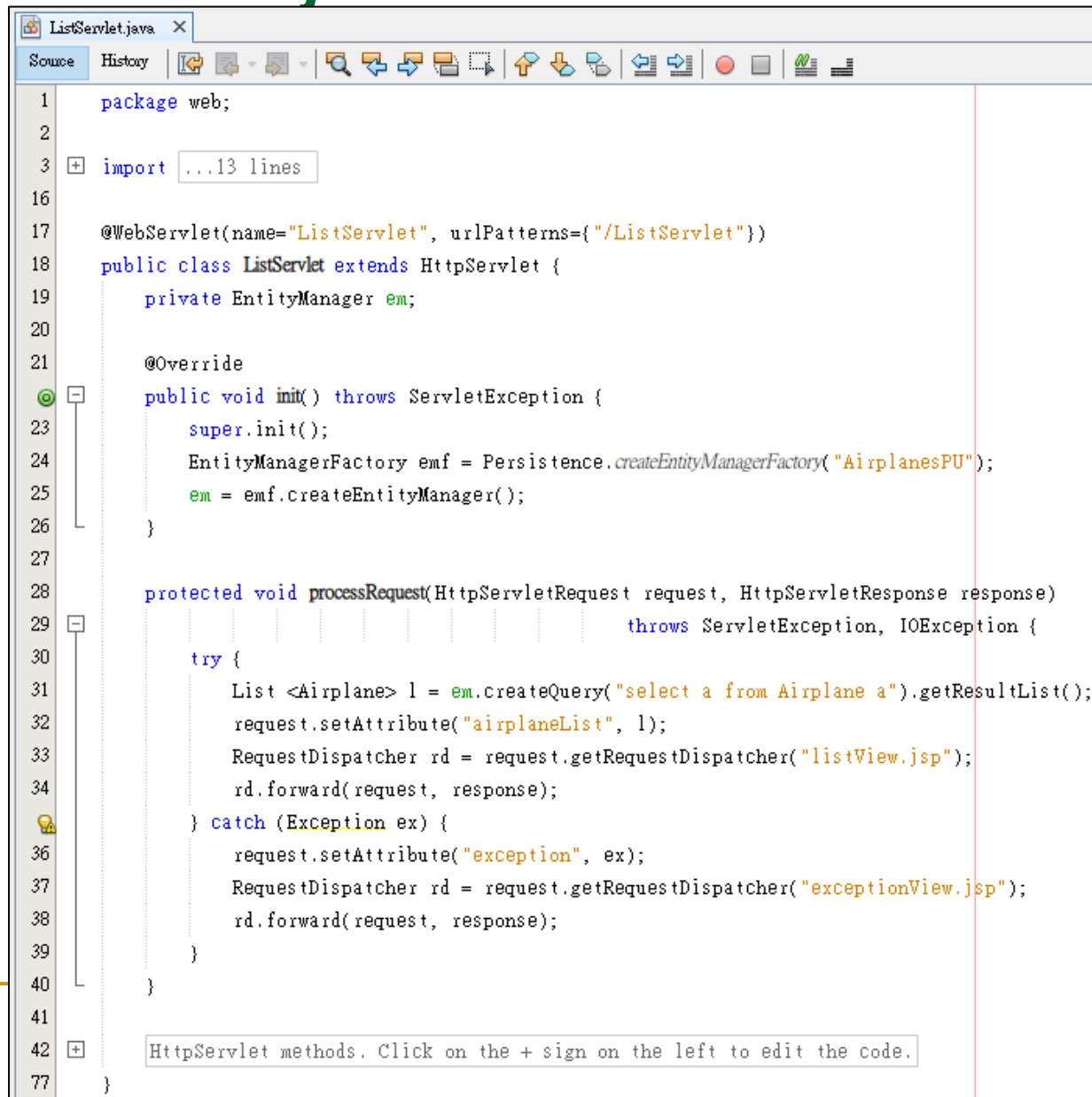
```
index.jsp
Source History
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Airplanes Page</title>
</head>
<body>
<h1>Airplanes Page</h1>
<a href="ListServlet">List All Airplanes</a><br>

<form action="AddServlet" method="post">
  ID: <input type="text" name="id"><br>
  Type: <input type="text" name="type"><br>
  Engines: <input type="text" name="engines"><br>
  <input type="submit" value="Add"><br>
</form>
</body>
</html>
```

```
web.xml
Source General Servlets Filters Pages References Security History
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3 <servlet>
4   <servlet-name>ListServlet</servlet-name>
5   <servlet-class>web.ListServlet</servlet-class>
6 </servlet>
7 <servlet>
8   <servlet-name>AddServlet</servlet-name>
9   <servlet-class>web.AddServlet</servlet-class>
10 </servlet>
11 <servlet-mapping>
12   <servlet-name>ListServlet</servlet-name>
13   <url-pattern>/ListServlet</url-pattern>
14 </servlet-mapping>
15 <servlet-mapping>
16   <servlet-name>AddServlet</servlet-name>
17   <url-pattern>/AddServlet</url-pattern>
18 </servlet-mapping>
19 <session-config>
20   <session-timeout>
21     30
22   </session-timeout>
23 </session-config>
24 </web-app>
```

# ListServlet.java



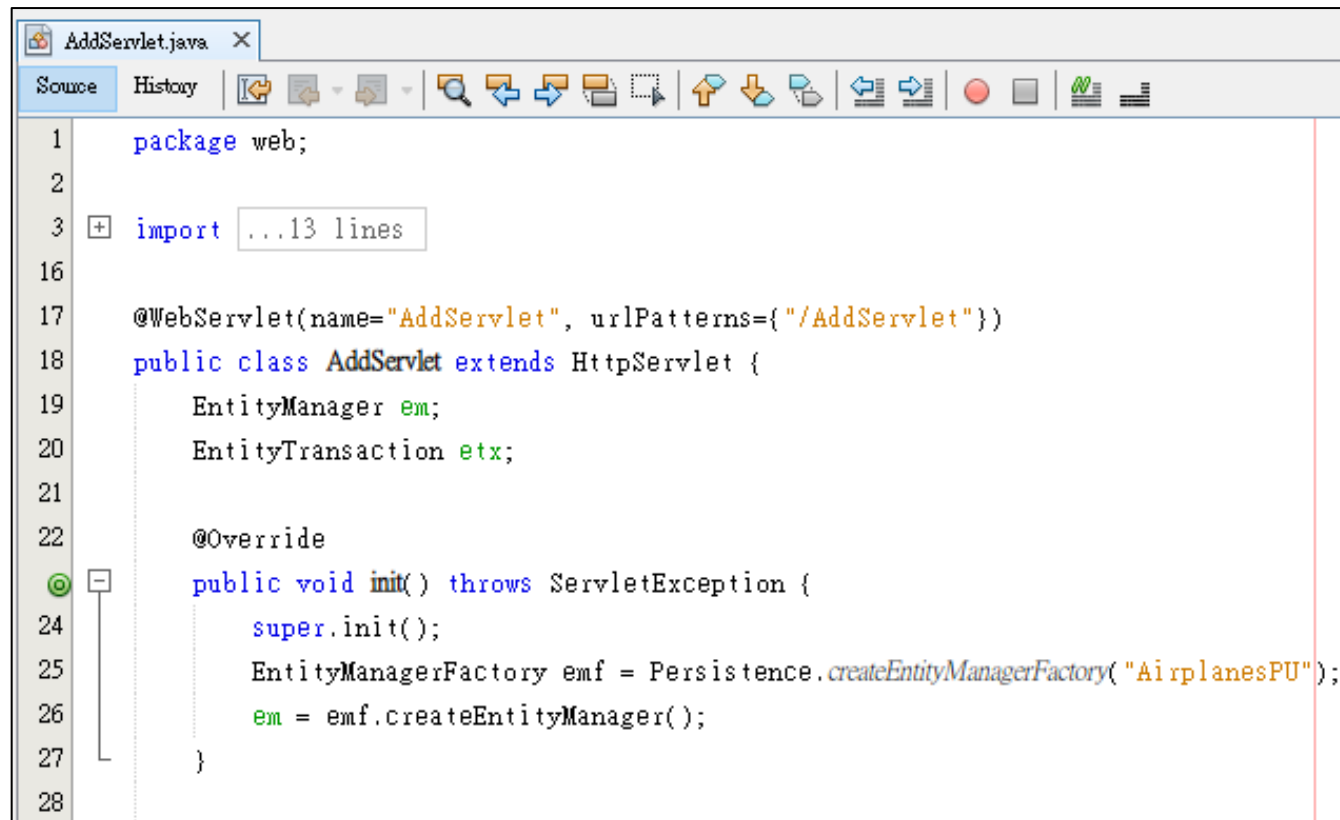
```
1 package web;
2
3 import ...13 lines
4
5
6
7
8
9
10
11
12
13
14
15
16
17 @WebServlet(name="ListServlet", urlPatterns={"/ListServlet"})
18 public class ListServlet extends HttpServlet {
19     private EntityManager em;
20
21     @Override
22     public void init() throws ServletException {
23         super.init();
24         EntityManagerFactory emf = Persistence.createEntityManagerFactory("AirplanesPU");
25         em = emf.createEntityManager();
26     }
27
28     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
29         throws ServletException, IOException {
30         try {
31             List<Airplane> l = em.createQuery("select a from Airplane a").getResultList();
32             request.setAttribute("airplaneList", l);
33             RequestDispatcher rd = request.getRequestDispatcher("listView.jsp");
34             rd.forward(request, response);
35         } catch (Exception ex) {
36             request.setAttribute("exception", ex);
37             RequestDispatcher rd = request.getRequestDispatcher("exceptionView.jsp");
38             rd.forward(request, response);
39         }
40     }
41
42     HttpServlet methods. Click on the + sign on the left to edit the code.
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77 }
```

# listView.jsp / exceptionView.jsp

```
listView.jsp
Source History
1 <%@page contentType="text/html" pageEncoding="UTF-8"%>
2 <%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core">
3 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional
4 "http://www.w3.org/TR/html4/loose.dtd">
5
6 <html>
7 <head>
8     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
9     <title>Airplane List</title>
10 </head>
11 <body>
12     <h1>Airplanes</h1>
13     <ul>
14         <c:forEach var="airplane" items="${airplaneList}">
15             <li>
16                 Airplane id = ${airplane.id} is a ${airplane.type} with ${airplane.engines} engines
17             </li>
18         </c:forEach>
19     </ul>
20 </body>
21 </html>
```

```
exceptionView.jsp
Source History
1 <%@page contentType="text/html" pageEncoding="UTF-8"%>
2 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
3 "http://www.w3.org/TR/html4/loose.dtd">
4
5 <html>
6 <head>
7     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8     <title>Something went wrong</title>
9 </head>
10 <body>
11     <h1>That broke</h1>
12     Here's the error report<br>
13     ${exception}
14 </body>
15 </html>
```

# AddServlet.java



```
1  package web;
2
3  + import ...13 lines
16
17  @WebServlet(name="AddServlet", urlPatterns={"/AddServlet"})
18  public class AddServlet extends HttpServlet {
19      EntityManager em;
20      EntityTransaction etx;
21
22      @Override
23      public void init() throws ServletException {
24          super.init();
25          EntityManagerFactory emf = Persistence.createEntityManagerFactory("AirplanesPU");
26          em = emf.createEntityManager();
27      }
28
```



```

29     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
30     {
31         try {
32             String id = request.getParameter("id");
33             String type = request.getParameter("type");
34             int engines = Integer.parseInt(request.getParameter("engines"));
35             Airplane a = new Airplane();
36             a.setId(id);
37             a.setType(type);
38             a.setEngines(engines);
39
40             EntityManager etx = em.getTransaction();
41             etx.begin();
42             em.persist(a);
43             etx.commit();
44
45             List<Airplane> l = em.createQuery("select a from Airplane a").getResultList();
46             request.setAttribute("airplaneList", l);
47             RequestDispatcher rd = request.getRequestDispatcher("listView.jsp");
48             rd.forward(request, response);
49         } catch (Exception ex) {
50             request.setAttribute("exception", ex);
51             RequestDispatcher rd = request.getRequestDispatcher("exceptionView.jsp");
52             rd.forward(request, response);
53         }
54     }
55
56     // HttpServlet methods. Click on the + sign on the left to edit the code.
57
58     // doGet()
59
60     // doPost()
61
62     // destroy()
63
64     // init()
65
66     // isReady()
67
68     // service()
69
70     // setRequest()
71
72     // setResponse()

```

# 執行結果

Airplanes Page

localhost:8080/Airpl...

## Airplanes Page

[List All Airplanes](#)

ID:

Type:

Engines:

Add

Airplanes Page

localhost:8080/Airpl...

## Airplanes Page

[List All Airplanes](#)

ID:

Type:

Engines:

Add

Airplane List

localhost:8080/Airpl...

## Airplanes

- Airplane id = CI005 is a Boeing 747 with 4 engines
- Airplane id = CI007 is a Airbus A-380 with 6 engines
- Airplane id = CI009 is a MD-90 with 2 engines

Airplane List

localhost:8080/Airpl...

## Airplanes

- Airplane id = CI005 is a Boeing 747 with 4 engines
- Airplane id = CI007 is a Airbus A-380 with 6 engines
- Airplane id = CI009 is a MD-90 with 2 engines
- Airplane id = CI011 is a Boeing 787 with 8 engines

# Lab 2

- 使用 DVDMaven 專案
- 新增 Exercise 資料庫
- 刪除原 Model.DVDItem.java
- 建立 Entity 類別 Model.DVDItem
  - 類別名稱為預設(與資料表名稱相同)
  - 主鍵值型態為 long, 產生策略為容器自選
  - 屬性包含 String title, year, genre
    - 資料表欄位為 title, yearstr, genre
  - 提供屬性存取方法
- 檢視 persistence.xml

# Lab 2

- 新增 controller.ListDVDServlet.java
  - EntityManager 屬性
  - public void init()
    - 建立 EntityManager 物件
  - protected void processRequest(...)
    - JPQL 查詢字串: "SELECT dvd FROM DVDItem dvd"
    - 建立Query物件: EntityManager物件createQuery()
    - 取得查詢結果: Query物件getResultList()
    - 將List查詢結果加入ServletContext屬性DVDList
    - 轉送至list\_library.jsp頁面
- 修改 index.jsp
  - 顯示 所有DVD 超連結改為List\_DVD

# Lab 2

- 修改 `controller.AddDVDServlet.java`
  - EntityManager及EntityTransaction 物件屬性
  - `public void init()`
    - 建立 EntityManager 物件
  - `protected void processRequest(...)`
    - 保留請求參數取得及驗證程式碼
      - 取得請求參數 `title, year, genre`
      - 驗證請求參數
      - 驗證失敗轉送`error.jsp`
    - 建立DVDItem物件
      - 無參數建構子建構
      - 請求參數`title, year, genre` 設定DVDItem 物件對應屬性

# Lab 2

- 新增DVDItem資料
  - 取得交易物件 EntityTransaction
  - 啟動交易 (EntityTransaction物件 begin() 方法)
  - 資料新增至對應資料表中 (EntityManager 物件persist() 方法)
  - 確認變更 (EntityTransaction物件 commit() 方法)
- 查詢所有DVDItem列表
  - JPQL 查詢字串: "SELECT dvd FROM DVDItem dvd"
  - 建立Query物件: EntityManager物件createQuery()
  - 取得查詢結果: Query物件getResultList()
- 將DVDItem列表查詢結果加入ServletContext屬性DVDList
- 將建立的DVDItem物件加入Request屬性dvd
- 轉送至success.jsp頁面

# Lab 2

- 修改 `web.InitializeLibrary.java`
  - 刪除讀取`library.txt`部分程式
  - 僅保留`genreList`相關程式
- 刪除`library.txt`
- 修改`web.xml`
  - 刪除 `libraryFile` 初始參數
- 測試執行
- 檢視`DVDItem`資料表