**Maven, SpringMVC, Hibernate學習筆記**

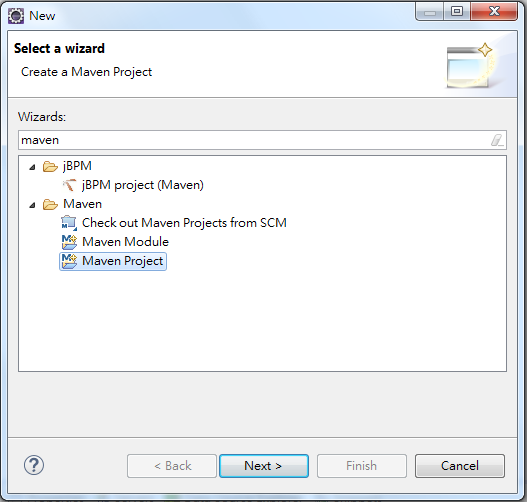
1. 軟體及版本需求

(1)Windows 7

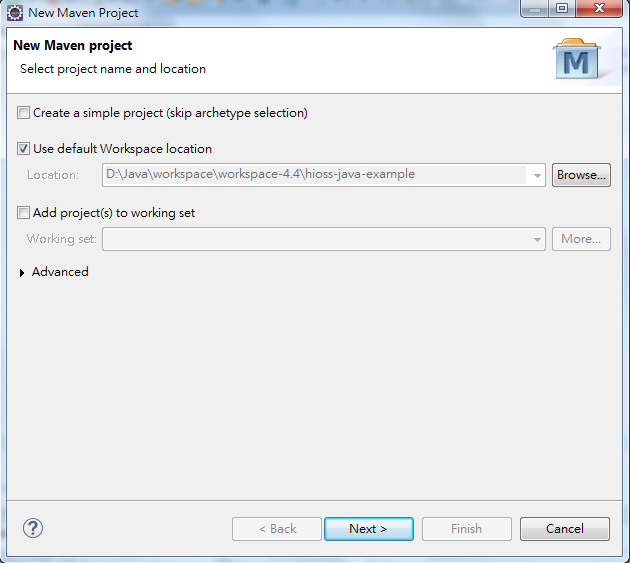
(2)Oracle JDK 1.7

(3)Eclipse 4.4

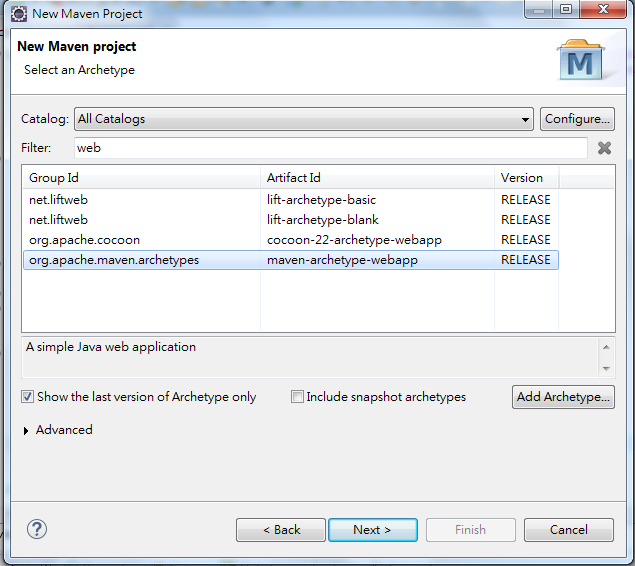
1. 新增Maven專案步驟
   1. 新增Web專案
      1. 點選功能選單[File]->[New]，選取Maven Project。



* + 1. 不須修改任何設定，直接按下[Next]。



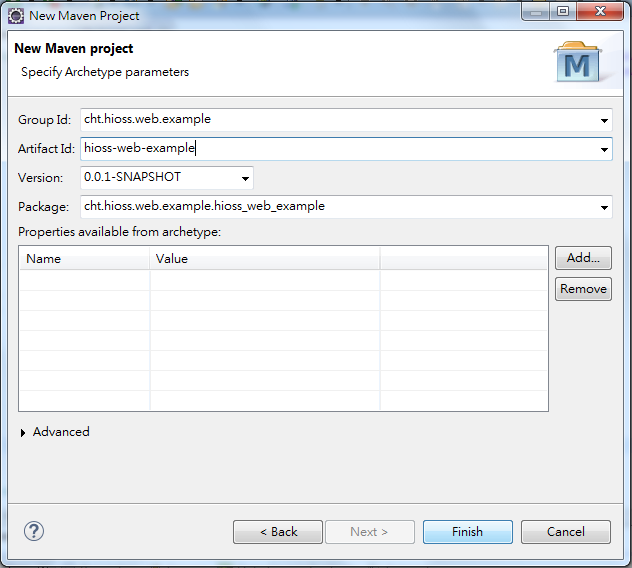
* + 1. 選取maven-archetype-webapp為範本，來建立網頁程式專案。



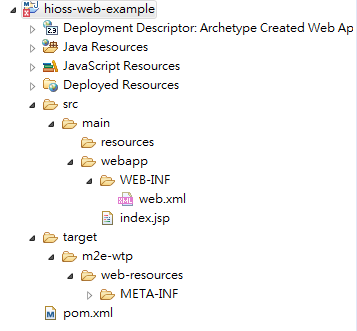
* + 1. 輸入專案資訊，其中Group Id和Artifact Id為必填欄位。

[Group Id]: 填入Java package

[Artifact Id]: 填入專案名稱



* + 1. 完成專案建置，自動產生目錄結構如下圖：



* + 1. 手動編輯pom.xml，參考以下範例加入properties、dependencies及build區塊設定。

<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>my.test.web.example</groupId>

<artifactId>mytest-web-example</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>mytest-web-example Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<!-- Generic properties -->

<java.version>1.7</java.version>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<!-- Web -->

<jsp.version>2.2</jsp.version>

<jstl.version>1.2</jstl.version>

<servlet.version>2.5</servlet.version>

</properties>

<dependencies>

<!-- Other Web dependencies -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>${jstl.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>${servlet.version}</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>jsp-api</artifactId>

<version>${jsp.version}</version>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<finalName>mytest-web-example</finalName>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.2</version>

<configuration>

<source>${java.version}</source>

<target>${java.version}</target>

</configuration>

</plugin>

<plugin>

<artifactId>maven-war-plugin</artifactId>

<version>2.6</version>

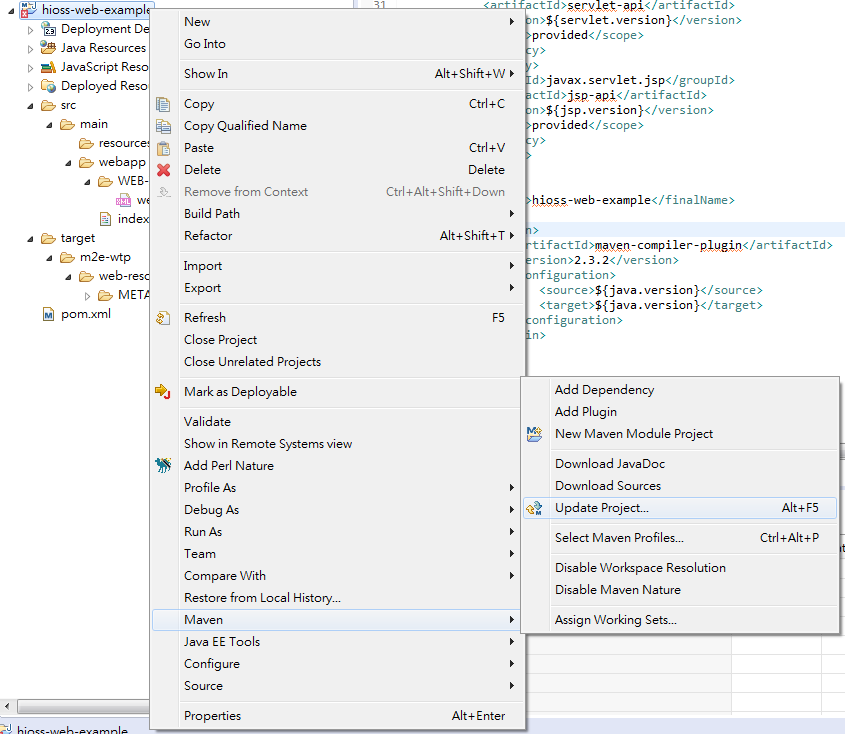
</plugin>

</plugins>

</build>

</project>

* + 1. 在專案上按下滑鼠右鍵，選取[Maven]->[Update Project]，以更新專案狀態。



* 1. 修改web.xml

<!-- xml版本要改為3.0 -->

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd" id="WebApp\_ID"

version="3.0">

<display-name>Archetype Created Web Application</display-name>

<display-name>Spring MVC Application</display-name>

<!-- 設定DispatcherServlet -->

<servlet>

<servlet-name>DispatcherServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<!-- spring 設定檔名稱預設為[DispatchServvlet Name]-servlet.xml -->

<!-- 如要自行指定spring設定檔，須設定listener 及 spring 設定檔名稱參數 -->

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring-config.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>DispatcherServlet</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

* 1. 修改pom.xml，加入spring 套件

<?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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>my.test.test.web</groupId>

<artifactId>test-springmvc2</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>test-springmvc2 Maven Webapp</name>

<!-- FIXME change it to the project's website -->

<url>http://www.example.com</url>

<properties>

<!-- Generic properties -->

<java.version>1.7</java.version>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<!-- Web -->

<jsp.version>2.2</jsp.version>

<jstl.version>1.2</jstl.version>

<servlet.version>2.5</servlet.version>

<spring.version>4.3.17.RELEASE</spring.version>

</properties>

<dependencies>

<!-- Other Web dependencies -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>${jstl.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>${servlet.version}</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>jsp-api</artifactId>

<version>${jsp.version}</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>${spring.version}</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

<scope>compile</scope>

<exclusions>

<exclusion>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-expression</artifactId>

<version>${spring.version}</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>${spring.version}</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>${spring.version}</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>${spring.version}</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aspects</artifactId>

<version>${spring.version}</version>

</dependency>

</dependencies>

<build>

<finalName>test-springmvc</finalName>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.2</version>

<configuration>

<source>${java.version}</source>

<target>${java.version}</target>

</configuration>

</plugin>

<plugin>

<artifactId>maven-war-plugin</artifactId>

<version>2.6</version>

</plugin>

</plugins>

</build>

</project>

* 1. 建立Spring設定檔(spring-config.xml)

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:p="http://www.springframework.org/schema/p"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xmlns:beans="http://www.springframework.org/schema/beans"

xmlns:jee="http://www.springframework.org/schema/jee"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc.xsd">

<!-- 掃描指定的package註冊含有@Component, @Controller, @Repository, @Service等地class為bean -->

<context:component-scan base-package="my.test" />

<context:annotation-config />

<!-- 使用Annotation來作URL Mapping -->

<mvc:annotation-driven />

<!-- resources通常也會作mapping，以免網頁存取不到相對路徑的靜態檔案如css檔以及圖檔等 -->

<mvc:resources location="/resources/" mapping="/resources/\*\*" />

<!-- 宣告render jsp網頁的bean -->

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<!-- 使用的JSTL標籤故一併宣告 -->

<property name="viewClass" value="org.springframework.web.servlet.view.JstlView" />

<property name="prefix" value="/WEB-INF/views/" />

<property name="suffix" value=".jsp" />

</bean>

</beans>

* 1. Controller 範例

package my.test.test.web.controller;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

@Controller

public class IndexController {

@RequestMapping(value = "/", method = RequestMethod.GET)

public String index(Model model) {

model.addAttribute("username", "Lawren");

return "index";

}

}

* 1. index.jsp 範例

檔案放在WEB-INF/views/ 目錄下。

<html>

<body>

<h2>${username}, Hello World!</h2>

</body>

</html>

1. logging
   1. 修改pom.xml，加入slf4j 及log4j

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.25</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.2</version>

</dependency>

* 1. 加入log4j設定檔

檔案放在src/main/resources/log4j.xml 目錄下。

<?xml version="1.0" encoding="UTF-8"?>

<!--宣告log4j.xml的文件型別定義(Document Type Definition, DTD) -->

<!DOCTYPE log4j:configuration SYSTEM "log4j.dtd">

<!--所有log4j的設定都必須放在 <log4j:configuration>...</log4j:configuration>之內 -->

<log4j:configuration xmlns:log4j="http://jakarta.apache.org/log4j/">

<!-- Appenders -->

<appender name="Console" class="org.apache.log4j.ConsoleAppender">

<!-- 設定appender的layout為PatternLayout -->

<layout class="org.apache.log4j.PatternLayout">

<!-- 設定layout的pattern，參數名為conversionPattern，value為想要的輸出格式 -->

<param name="conversionPattern" value="%-d{yyyy-MM-dd HH:mm:ss} [%C{5}]-[%p] %m%n" />

</layout>

</appender>

<!-- DailyRollingFileAppender的第一個log檔並不會加上DatePattern的樣式， -->

<!-- 必須在下一次週期產生的log檔才會加上DatePattern的樣式。 -->

<!--所以基本上當天產生的log檔算是第一次的log檔，並不會加上Pattern樣式 -->

<appender name="File" class="org.apache.log4j.DailyRollingFileAppender">

<param name="File" value="**${catalina.home}/logs/server\_log.txt**" /> <!--檔案路徑 -->

<!--DatePattern是DailyRollingFileAppender的一個重要參數，用來設定檔案輸出的週期 -->

<param name="DatePattern" value="'.'yyyy-MM-dd-HH'.log'" />

<layout class="org.apache.log4j.PatternLayout">

<param name="ConversionPattern" value="%-d{yyyy-MM-dd HH:mm:ss,SSS} [%C{5}]-[%p] %m%n" />

</layout>

</appender>

<appender-ref ref="Console" />

<appender-ref ref="File" />

<!--RootLogger(全專案套用) -->

<!--rootLogger的LEVEL設定為DEBUG，DEBUG以上等級的資訊也都會寫出 -->

<!--主要class搭配：public static Logger logger = Logger.getLogger(MyPractice.class); -->

<root>

<priority value="DEBUG" /> <!--印出INFO以上的資訊 -->

<appender-ref ref="Console" /> <!--將rootLogger的appender參考至剛設定好的Console appender -->

<appender-ref ref="File" /> <!--將rootLogger的appender參考至剛設定好的File appender -->

</root>

<logger name="tw.blogger.springtech.springmvc.mvc">

<level value="debug" />

<appender-ref ref="Console" />

</logger>

</log4j:configuration>

* 1. Copy log4j.dtd

If you are using XML-based log4j, then yes, you will need the DTD file. Copy this DTD file into your workspace:

http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/xml/doc-files/log4j.dtd

Make sure it sits at the same directory as your log4j XML file.

1. Spring Config
   1. Config DispatchServlet

不使用web.xml，有兩種方式：

(1) 繼承AbstractDispatcherServletInitializer 來載入，設定檔格式為xml

public class WebAppInitializer extends AbstractDispatcherServletInitializer {

@Override

protected WebApplicationContext createServletApplicationContext() {

XmlWebApplicationContext ctx=new XmlWebApplicationContext();

ctx.setConfigLocation("/WEB-INF/springconfig/dispatchservletcontext.xml");

return ctx;

}

@Override

protected String[] getServletMappings() {

return new String[] {"/"};

}

@Override

protected WebApplicationContext createRootApplicationContext() {

XmlWebApplicationContext ctx=new XmlWebApplicationContext();

ctx.setConfigLocation("/WEB-INF/springconfig/service.xml");

//如果有多個xml要載入的話，可以選擇setConfigLocations，以陣列型式傳入

return ctx;

}

}

(2) 繼承AbstractAnnotationConfigDispatcherServletInitializer 來載入，設定檔格式為Java Class

public class WebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

@Override

protected String[] getServletMappings() {

return new String[] { "/" };

}

@Override

protected Class<?>[] getRootConfigClasses() {

return new Class<?>[] { RootConfig.class };

}

@Override

protected Class<?>[] getServletConfigClasses() {

return new Class<?>[] { WebConfig.class };

}

}

1. Hibernate
   1. 參考資料1

<http://websystique.com/springmvc/spring-4-mvc-and-hibernate4-integration-example-using-annotations/>

* 1. 修改pom.xml

...

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-entitymanager</artifactId>

<version>${hibernate.version}</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>${hibernate.version}</version>

</dependency>

<dependency>

<groupId>org.hibernate.javax.persistence</groupId>

<artifactId>hibernate-jpa-2.1-api</artifactId>

<version>1.0.0.Final</version>

</dependency>

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.1.4.jre7</version>

</dependency>

<!-- jsr303 validation -->

<dependency>

<groupId>javax.validation</groupId>

<artifactId>validation-api</artifactId>

<version>1.1.0.Final</version>

</dependency>

....

<properties>

<hibernate.version>4.3.11.Final</hibernate.version>

...

</properties>

* 1. 參考2

https://www.journaldev.com/3531/spring-mvc-hibernate-mysql-integration-crud-example-tutorial

1. JPA

https://spring.io/guides/gs/accessing-data-mysql/

1. Spring Security

https://docs.spring.io/spring-security/site/docs/3.1.7.RELEASE/reference/ns-config.html

* 1. web.xml Configuration

<filter>

<filter-name>springSecurityFilterChain</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>springSecurityFilterChain</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

* 1. <http> Configuration

<http auto-config='true'>

<intercept-url pattern="/\*\*" access="ROLE\_USER" />

</http>

To add some users, you can define a set of test data directly in the namespace:

<authentication-manager>

<authentication-provider>

<user-service>

<user name="jimi" password="jimispassword" authorities="ROLE\_USER, ROLE\_ADMIN" />

<user name="bob" password="bobspassword" authorities="ROLE\_USER" />

</user-service>

</authentication-provider>

</authentication-manager>

The <authentication-provider> element creates a DaoAuthenticationProvider bean and

the <user-service> element creates an InMemoryDaoImpl.

1. 參考文件

無痛學習SpringMVC與Spring Security 共 31 篇

https://ithelp.ithome.com.tw/users/20072603/ironman/951