spring4 + mybatis+mysql

<http://www.concretepage.com/spring-4/spring-4-mybatis-3-example-with-mapperscan-and-sqlsessionfactorybean>

<http://edwin.baculsoft.com/2015/01/a-simple-spring-4-and-mybatis-transaction-example/>

<http://www.javacodegeeks.com/2014/02/building-java-web-application-using-mybatis-with-spring.html>

spring security

<http://haohaoxuexi.iteye.com/category/182468>

spring 例子\*\*\*\*\*\*\*\*

<http://websystique.com/>

<http://websystique.com/springmvc/spring-4-mvc-form-validation-with-hibernate-jsr-validator-resource-handling-using-annotations/>

spring文档

<http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/#jdbc-core>

<https://github.com/hantsy/spring4-sandbox>

spring mvc: \*\*\*

<http://wenku.baidu.com/link?url=Pyyu4gGTjKfeTGDuOyIpXItStLAgvt3pPpdGc0wPcbQRlfvbnB4Z28DLQMrAX9hNQ83OGn57LFVrj7gNcSSaWabR_2YDQUxPWcO80_LZUkO>

web安全：

<http://www.cnblogs.com/Mainz/archive/2012/11/01/2749874.html>

java实例:

<http://www.concretepage.com/>

Spring 登录权限实例：

<http://websystique.com/spring-security/spring-security-4-custom-login-form-annotation-example/>

<http://www.mkyong.com/spring/spring-propertysources-example/>

mysql hibernate security:

<http://www.javacodegeeks.com/2013/05/spring-mvc-security-with-mysql-and-hibernate.html>

<http://www.concretepage.com/spring/spring-security/spring-mvc-security-jdbc-authentication-example-with-custom-userdetailsservice-and-database-tables-using-java-configuration>

<http://howtodoinjava.com/spring/spring-security/custom-userdetailsservice-example-for-spring-3-security/>

<http://howtodoinjava.com/spring/spring-security/spring-3-security-siteminder-pre-authentication-example/>

AOP实现

<http://www.codeweblog.com/%E4%BD%BF%E7%94%A8-%E9%93%BE%E5%BC%8F%E4%BB%A3%E7%90%86-%E5%AE%9E%E7%8E%B0-aop/>

设计模式

<http://www.cnblogs.com/java-my-life/archive/2012/06/08/2538146.html>

js

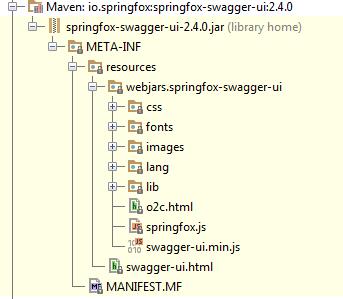
<http://lib.csdn.net/base/18>

Spring MVC MultiActionController example

<http://www.mkyong.com/spring-mvc/spring-mvc-multiactioncontroller-example/>

这次标题够长的,在Servlet3协议规范中,包含在JAR文件/META-INF/resources/路径下的资源可以直接访问了。这么说一说，可能感觉不到到底有什么好处，以往的JSP或者HTML页面只能存在站点的目录下，或者在WEB-INF目录下（只是不能直接访问）。

规范说，${jar}/META-INF/resources/被视为根目录，假设home.jsp被放在${jar}/META-INF/resources/home.jsp，用户可以直接通过 http://域名/home.jsp 访问了。



Servlet 3.0 which Tomcat 7 supports includes the ability to package jsps into a jar.

You need to:

place your jsps in META-INF/resources directory of your jar

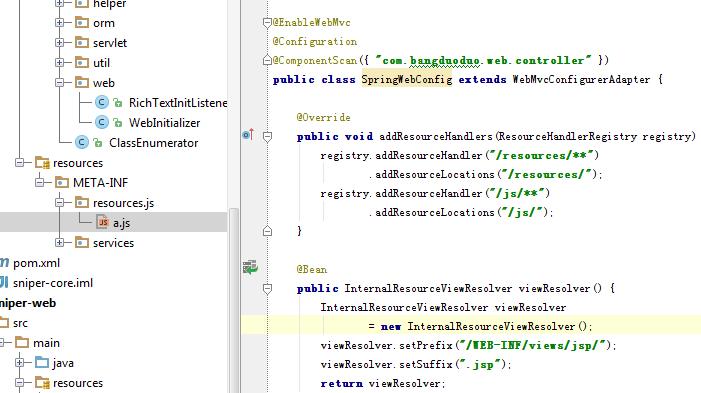
optionally include a web-fragment.xml in the META-INF directory of your jar

place the jar in WEB-INF/lib directory of your war

You should then be able to reference your jsps in your context. For example if you have a jsp META-INF/resources/test.jsp you should be able reference this at the root of your context as test.jsp

<https://blogs.oracle.com/alexismp/entry/web_inf_lib_jar_meta>

如果使用了spring mvc就需要用到<mvc:resources mapping="/images/\*\*" location="/static\_resources/images/"/>



一個Web應用程式基本上會由以下項目所組成：

靜態資源（HTML、圖片、聲音等）

Servlet

JSP

自定義類別

工具類別

部署描述檔（web.xml等）、設定資訊（Annotation等）

Web應用程式存在既定的目錄結構。舉例來說，如果一個應用程式的環境路徑（Context path）是/openhome，則所有的資源項目必須以/openhome為根目錄依一定的結構擺放。基本上根目錄中的資源可以直接下載，例如若index.html位在/openhome下，則可以直接以/openhome/index.html來取得。

Web應用程式存在一個特殊的**/WEB-INF**目錄，此目錄中存在的 資源項目不會被列入應用程式根目錄中可直接存取的項目，也就是說，客戶端（例如瀏覽器）不可以直接請求/WEB-INF中的資源（直接在網址上指明存取 /WEB-INF），否則就是404 Not Found的錯誤結果。/WEB-INF中的資源項目有著一定的名稱與結構。例如：

**/WEB-INF/web.xml** 是部署描述檔

**/WEB-INF/classes**用來放置應用程式所用到的自定義類別（.class），必須包括套件（Package）結構

**/WEB-INF/lib** 用來放置應用程式所用到的JAR（Java ARchive）檔案

Web應用程式所用到的JAR檔案，當中可以放置Servlet、JSP、自定義類別、工具類別、部署描述檔等，應用程式的類別載入器可以從JAR中載入對應的資源。

你可以在JAR檔案的**/META-INF/resources**目 錄中放置靜態資源或JSP等，例如若在/META-INF/resources中放個index.html，若請求的URL中包括 /openhome/index.html，但實際上/openhome根目錄底下不存在index.html，則JAR中的/META- INF/resources/index.html會被使用。

如果你要到用到某個類別，則Web應用程式會到/WEB-INF/classes中試著載入類別，若無，再試著從/WEB-INF/lib的JAR檔案中尋找類別檔案（若還沒有找到，則會到容器實作本身存放類別或JAR的目錄中尋找，但位置視實作廠商而有所不同）。

客戶端不可以直接請求/WEB-INF中的資源，但你可以透過程式面的控管，讓程式來取得/WEB-INF中的資源，像是使用ServletContext的getResource()與getResourceAsStream()，或是透過RequestDispatcher請求調派。

如果對Web應用程式的URL最後是以/結尾，而且確實存在該目錄，則Web容器必須傳回該目錄下的歡迎頁面, 你可以在部署描述檔web.xml中包括以下的定義，指出可用的歡迎頁面名稱為何，Web容器會依序看看是否有對應的檔案存在，如果有則傳回給客戶端：

**<welcome-file-list>**

**<welcome-file>index.html</welcome-file>**

**<welcome-file>default.jsp</welcome-file>**

**</welcome-file-list>**

如果找不到以上的檔案，則會嘗試至JAR的/META-INF/resources中尋找已置放的資源頁面。如果URL最後是以/結尾，但不存在該目錄，則會使用預設Servlet（如果有定義的話，參考 [URL 模式](http://openhome.cc/Gossip/ServletJSP/URLPattern.html) 中的說明）。

整個Web應用程式可以被封裝為一個WAR（Web ARchive）檔案，例如openhome.war，以便部署至Web容器。