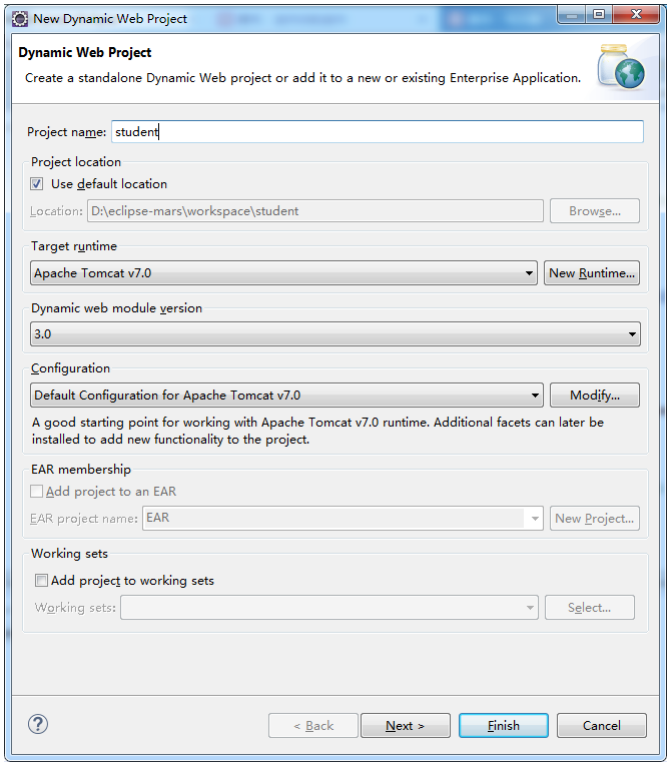
【Java框架型项目从入门到装逼】第七节 - 学生管理系统项目搭建

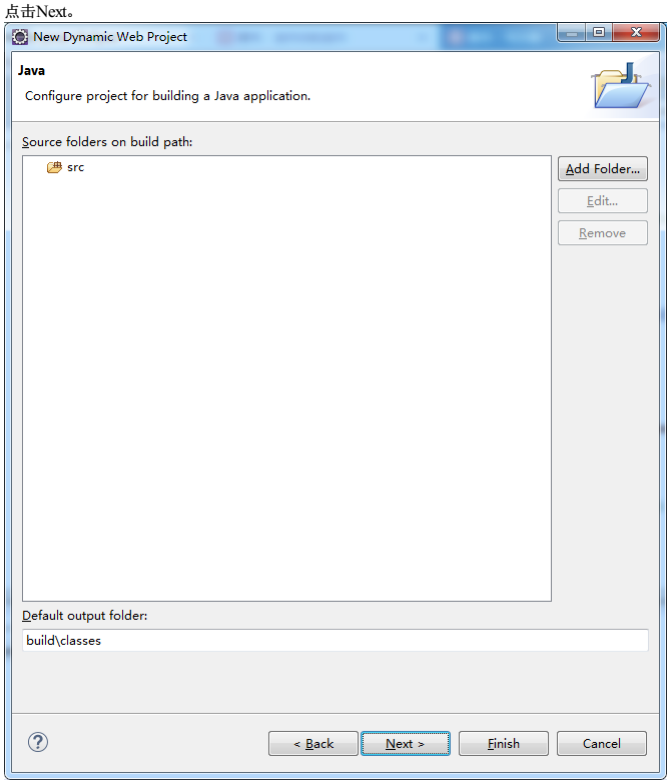
本次的教程是打算用Spring，SpringMVC以及传统的jdbc技术来制作一个简单的增删改查项目，对用户信息进行增删改查，就这么简单。

1.新建项目

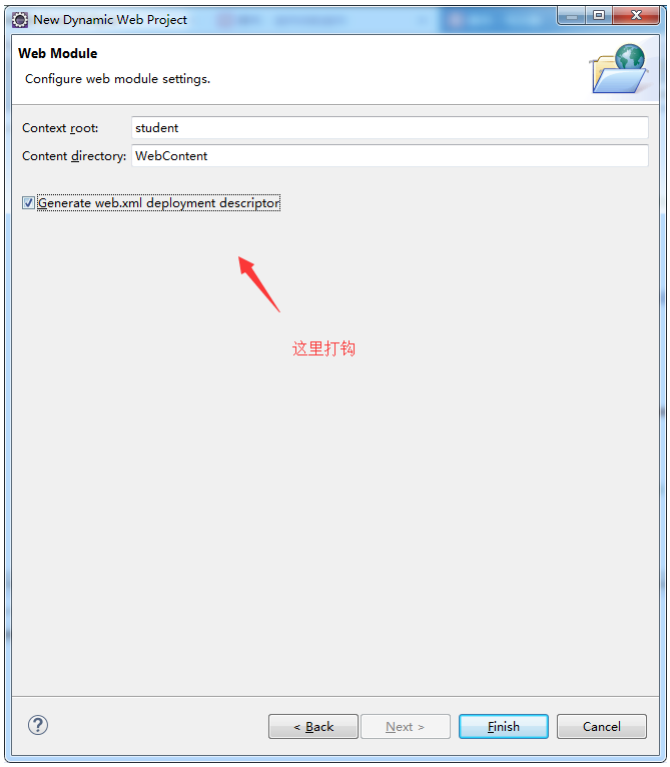
首先，打开eclipse，新建一个web项目。项目名称就叫做student，注意，新建项目的时候，因为是eclipse。所以你需要选择Dynamic Web Project。



点击Next。



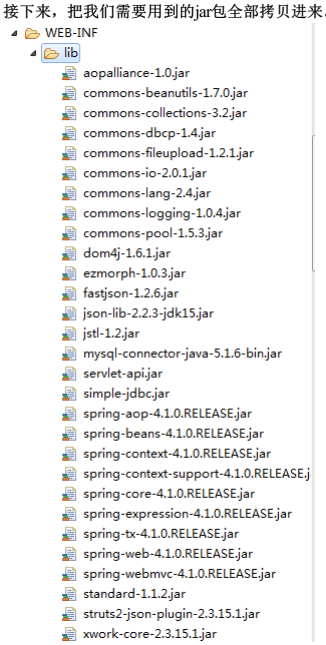
不管，继续Next。



Finish完成。

项目的目录结构如下：

接下来，把我们需要用到的jar包全部拷贝进来。

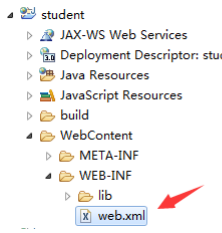


其中，simple-jdbc是我自己封装了一些jdbc操作，可以看成是一个小型的jdbc框架，具体如何使用会在以后讲到。

这些jar包，有的会用到，有的可能用不到。现在先不管，就把他们全部拷贝进来。

2.集成SpringMVC

SpringMVC在本项目中起到的作用就是一个请求分发器，所有的请求，我们都通过SpringMVC来分发。打开web.xml：




```
<?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
  <display-name>student</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

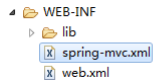
welcome-file-list是欢迎页的配置，我们不管，在前加上如下配置：

```
 <!-- 配置SpringMVC分发器 -->
     <servlet>
         <servlet-name>springmvc</servlet-name>
         <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
         <init-param>
             <param-name>namespace</param-name>
             <param-value>spring-mvc</param-value>
         </init-param>
      </servlet>

    <servlet-mapping>
         <servlet-name>springmvc</servlet-name>
         <url-pattern>*.do</url-pattern>
     </servlet-mapping>
```

这和配置普通Servlet的方式是一样，其中

```
 <init-param>
    <param-name>namespace</param-name>
    <param-value>spring-mvc</param-value>
</init-param>
```

这个配置的意思是在创建DispatcherServlet类的时候，就把其中的一个namespace属性赋值“spring-mvc”。这个名字是我们自己定的，你可以取别的名字，也可以就叫做spring-mvc，它对应的是 **WEB-INF** 目录下的 **spring-mvc.xml** 文件。现在，我们是没有这个文件的，所以得新建一个。



将一下代码拷贝进spring-mvc.xml中。

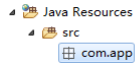
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:util="http://www.springframework.org/schema/util"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context-3.0.xsd
       http://www.springframework.org/schema/util
       http://www.springframework.org/schema/util/spring-util-3.0.xsd
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-mvc.xsd
      ">

      <!-- 包扫描器 -->
      <context:component-scan base-package="com.app.*"></context:component-scan>

      <!-- 开启注解驱动，写了以后，Spring的注解机制就开始生效 -->
      <mvc:annotation-driven >
          <mvc:message-converters register-defaults="true">
                <bean class="org.springframework.http.converter.StringHttpMessageConverter">
                    <property name="supportedMediaTypes">
                        <list>
                            <value>text/plain;charset=UTF-8</value>
                            <value>text/html;charset=UTF-8</value>
                        </list>
                    </property>
                </bean>
                <!-- 配置Fastjson支持 -->
                <bean class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter">
                    <property name="supportedMediaTypes">
                        <list>
                            <value>application/json</value>
                        </list>
                    </property>
                    <property name="features">
                        <!--
                            Fastjson的SerializerFeature序列化属性：
                                QuoteFieldNames———输出key时是否使用双引号,默认为true
                                WriteMapNullValue——是否输出值为null的字段,默认为false
                                WriteNullNumberAsZero—-数值字段如果为null,输出为0,而非null
                                WriteNullListAsEmpty—-List字段如果为null,输出为[],而非null
                                WriteNullStringAsEmpty—字符类型字段如果为null,输出为“”,而非null
                                WriteNullBooleanAsFalse-Boolean字段如果为null,输出为false,而非null
                         -->
                        <list>
                            <value>QuoteFieldNames</value>
                            <value>WriteMapNullValue</value>
                        </list>
                    </property>
                </bean>
            </mvc:message-converters>
        </mvc:annotation-driven>

        <!-- 配置SpringMVC的视图解析器 -->
        <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
            <property name="prefix" value="/" />
            <property name="suffix" value=".jsp" /><!--可为空,方便实现自己的依据扩展名来选择视图解释类的逻辑 -->
            <property name="viewClass" value="org.springframework.web.servlet.view.JstlView" />
        </bean>

</beans>
```

第一个配置，包扫描器：

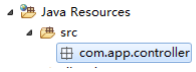
```
<context:component-scan base-package="com.app.*"></context:component-scan>
```

这个是啥意思呢，就是说，Tomcat容器启动的时候，会去扫描com.app下面所有的包和类，如果是符合要求的类，就new一下，装进Spring的bean工厂。

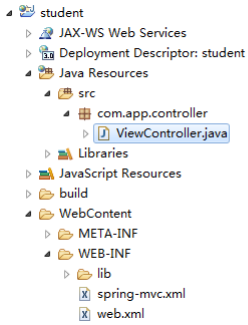
我们先把对应的目录结构建起来：



然后，建一个controller包：



这样一来，这个controller就是将来会被扫描的对象。现在，我们在里面新建一个ViewController类。



代码：

```
package com.app.controller;

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

@Controller
public class ViewController {

    @RequestMapping("/test")
    public void test(){
        System.out.println("test");
    }

}
```

我们给这个类打上@controller标记，到时候Sping框架就会认识他，在Tomcat服务器启动的时候就new出这个类，放到Spring的bean工厂中。

@RequestMapping的含义是请求路径。

关于这两个标记，我在 这篇文章 中已经做了详细的说明，在此就不再赘述。接着，我们可以去发布我们的项目了。

如果你还不会用eclipse和tomcat发布web项目，就看一下 Tomcat的安装配置与JavaWeb入门教程 。我在这里就不详细说了。

启动tomcat，打开浏览器，在地址栏输入：
http://localhost/student/test.do
回车，可以看到在控制台打印出了test字样：

```
一月 09, 2018 10:58:37 上午 org.springframework.web.servlet.ha
INFO: Mapped "{[/test],methods=[],params=[],headers=[],cons
一月 09, 2018 10:58:37 上午 org.springframework.web.servlet.mv
INFO: Looking for @ControllerAdvice: WebApplicationContext
一月 09, 2018 10:58:37 上午 org.springframework.web.servlet.mv
INFO: Looking for @ControllerAdvice: WebApplicationContext
一月 09, 2018 10:58:37 上午 org.springframework.web.servlet.Fr
INFO: FrameworkServlet 'springmvc': initialization complet
test
```

这就说明，流程已经走通了。注意，因为我本地tomcat配置的端口号是80，所以直接写localhost，不需要写localhost:80了。还有，为什么后面跟test.do？那是因为我们在配置Spring分发器的时候就规定了，只拦截所有 *.do 的请求。

```
<!-- 配置SpringMVC分发器 -->
    <servlet>
        <servlet-name>springmvc</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>namespace</param-name>
            <param-value>spring-mvc</param-value>
        </init-param>
     </servlet>

     <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>
```

本章就到这里，先撤了。

我要下载源码