所谓最小系统，就是足以使项目在SpringMVC框架下成功跑起来，并且做一些足够简单的事情（比如访问页面）的系统。

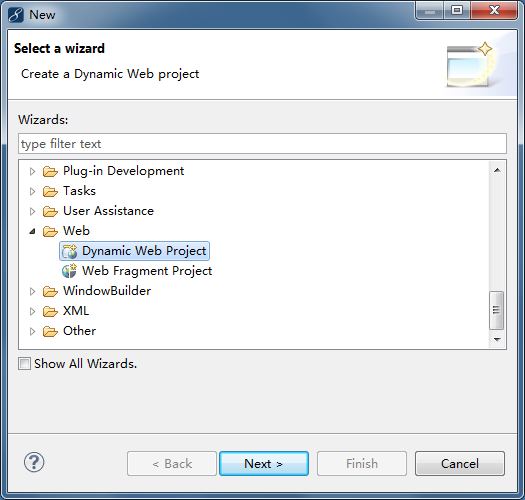
其它环境：

操作系统：Windows 7

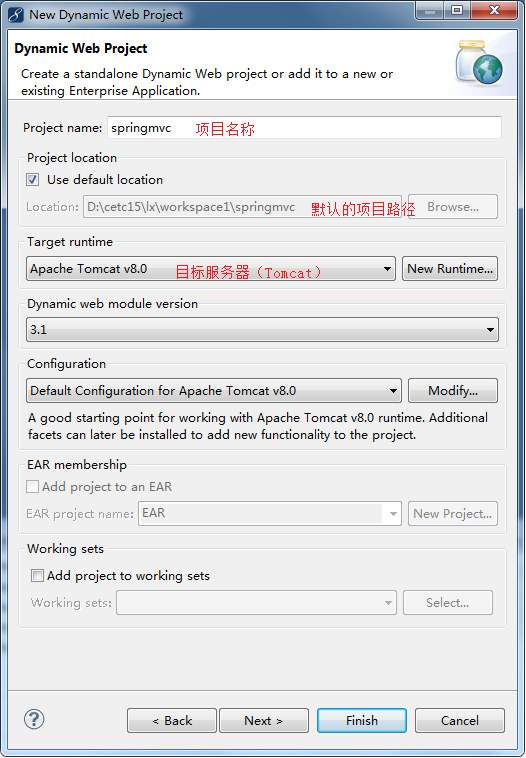
Tomcat v8.0

JDK：1.8

1. 新建一个项目

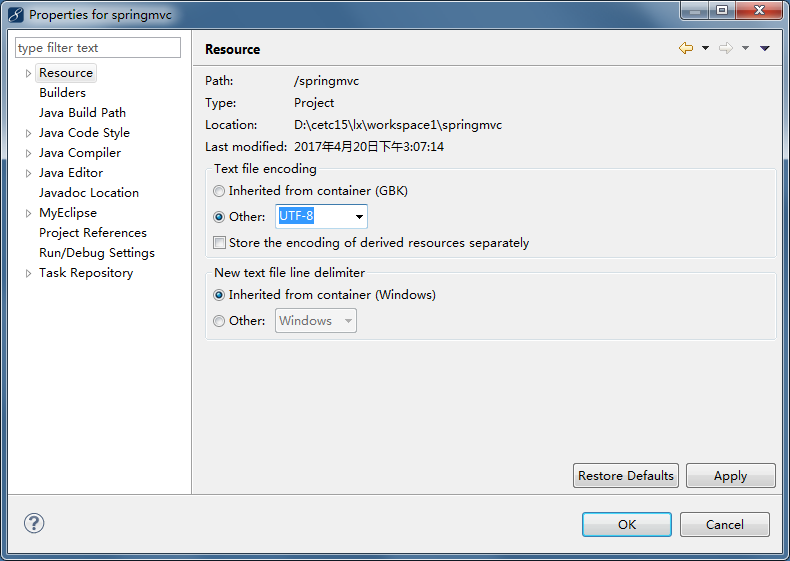


用MyEclipse新建项目，选择Dynamic Web Project（动态的Web项目）。点击Next



填写红字标记部分，点击Finish。OK，项目就建好了。

接下来一定要将项目的字符集改为UTF-8：右键项目-properties，修改---OK。

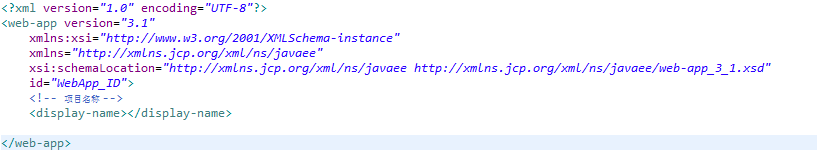


1. 编写web.xml

当我们打开WebRoot/WEB-INF目录的时候，发现里面只有一个lib目录，这是存放各种jar包的地方。我们知道一个web项目必须要有一个web.xml文件才行。

既然没有，我们自己写一个咯。

新建一个web.xml文件，点击Finish。填写如下内容。



这就完成了基本的配置，我的意思是说，现在这个项目就已经是一个标准的web项目了。

1. 验证web项目是否搭建成功

为了验证到目前为止的正确性，我们在WebRoot目录下面新建一个jsp文件。名字就叫index.jsp。



将项目部署到Tomcat来验证是否可以跑起来。

在项目上右键---Debug As---Debug on Server

直接点击Finish

经过一段时间，控制台开始打印日志信息，当看到这些信息的时候说明Tomcat已经启动完毕了。

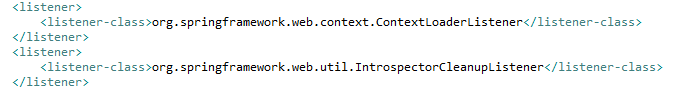
打开浏览器，在地址栏输入以下信息：<http://localhost:8080/springmvc/index.jsp>检测是否能够成功访问页面。

能够成功访问页面了，说明到目前为止的操作是正确的。

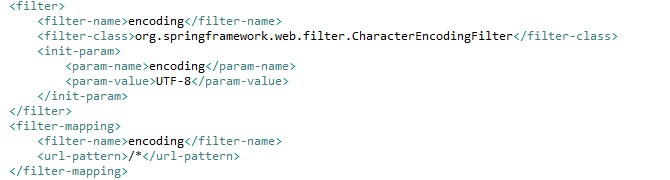
1. 集成SpringMVC

需要在web.xml里添加下面的配置

* 1. 配置监听器



* 1. 配置过滤器，解决POST乱码问题



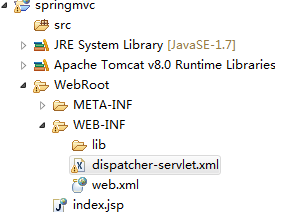
* 1. 配置SpringMVC分发器，拦截所有请求



在这个配置中，规定了DispatcherServlet的关联XML文件名称叫做dispatcher-servlet。

注意，这里的路径是相对于web.xml来说的，也就是说，这个文件也在WEB-INF的根目录下。

所以，需要在WEB-INF的根目录下新建一个dispatcher-servlet.xml文件。



至此，web.xml文件的编写就告一段落了。

* 1. 编写dispatcher-servlet.xml

dispatcher-servlet.xml的作用就是配置SpringMVC分发器。

配置如下：



根据配置，有3个需要注意的地方

A它会扫描com.springmvc包下所有的Java类，但凡是遇到有注解的，比如@Controller,@Service,@Autowired,就会将它们加入到Spring的bean工厂里面去。

B所有的静态资源文件，比如说js,css,images都需要放在/resources目录下，这个目录现在我们还没有建。

C所有的展示页面，比如jsp文件，都需要放置在/WEB-INF/pages目录下，这个目录现在我们也没有建。

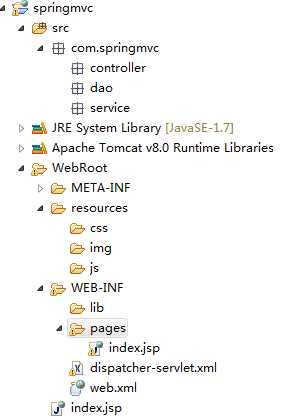
首先是Java文件的目录，在src处右键，新建一个com包，再在里面建一个springmvc包。

根据SpringMVC的分层，我们在springmvc包下面建3个包，分别是controller,service,dao。

这样的话，当项目一旦启动，springmvc就会扫描这3个包，将里面但凡是有注解的类都提取起来，放进Spring容器（或者说Spring的bean工厂），借由Spring容器来统一管理。这也就是你从来没有去new一个Controller的原因。

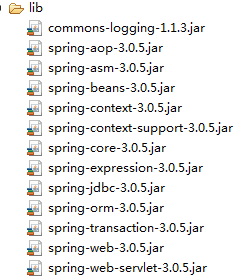
接下来，来建静态资源的目录。在WebRoot目录下新建一个resources文件夹。然后顺便把js,css,img的文件夹都建一下，这里就存放我们的静态资源文件。

最后，在WEB-INF目录下建一个pages文件夹，作为展示页面的存放目录。将之前的index.jsp拷贝进来。这样就配置的差不多了。



1. 导包和验证

将需要的jar包放到lib目录：



然后启动项目，验证一下到目前为止的构建是否正确。

发现报错了

C:\Users\Administrator\AppData\Roaming\feiq\RichOle\2359252872.bmp

它说我们在WEB-INF下面少了一个applicationContext.xml这个文件，原来，我们少了对SpringBean工厂的配置，它的意思就是说，我们要规定一下，在Spring容器启动的时候，需要自动加载哪些东西？

于是，加上applicationContext.xml。里面啥也不配置，再次启动Tomcat，这回不报错了。



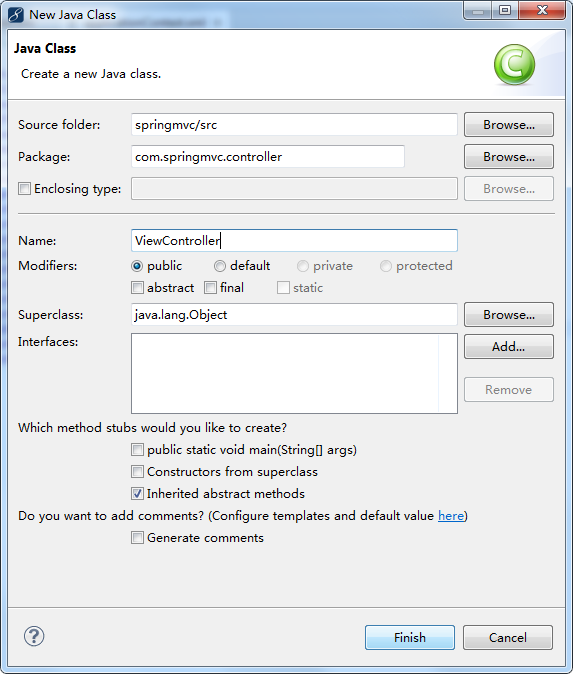
1. 配置ViewController

WEB-INF目录下的任何资源都是无法直接通过浏览器的url地址去访问的，保证了安全性。这也是我们为什么把页面都放在该目录下的原因。

为了有所区分，还单独建立了一个pages文件夹，将这些页面保存起来。

现在为了访问这个页面，我们需要用到SpringMVC的页面跳转机制。

我们在Controller包下新建一个ViewController，点击Finish。





我们只需要将想要访问的页面放在path里面，通过url传进来就行了。

因为添加了Java类，因此我们重新启动Tomcat。

启动完成后，在地址栏输入<http://localhost:8080/springmvc/view?path=index>

结果：

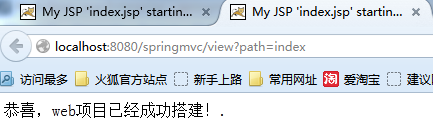


pagesindex.jsp是什么鬼？

原来我们在dispatcher-servlet.xml中少写了一个”/”。pages后面要有/。



保存后重启Tomcat，一切继续！



成功了！

1. 引入静态资源

比如在resources/img目录下放了一张图片，怎么引入到index.jsp呢？

background : *url(http://localhost:8080/springmvc/resources/img/hbase\_logo\_small.png)*;

background-size: *100% 100%*;

这的确是一种方式，可是它有一个缺点就是根路径写死了。

其实，可以在viewController里面拿到项目的根路径，然后传递到jsp页面就OK了。



${contextPath}可以取到Controller传过来的contextPath值。

成功了！