

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。

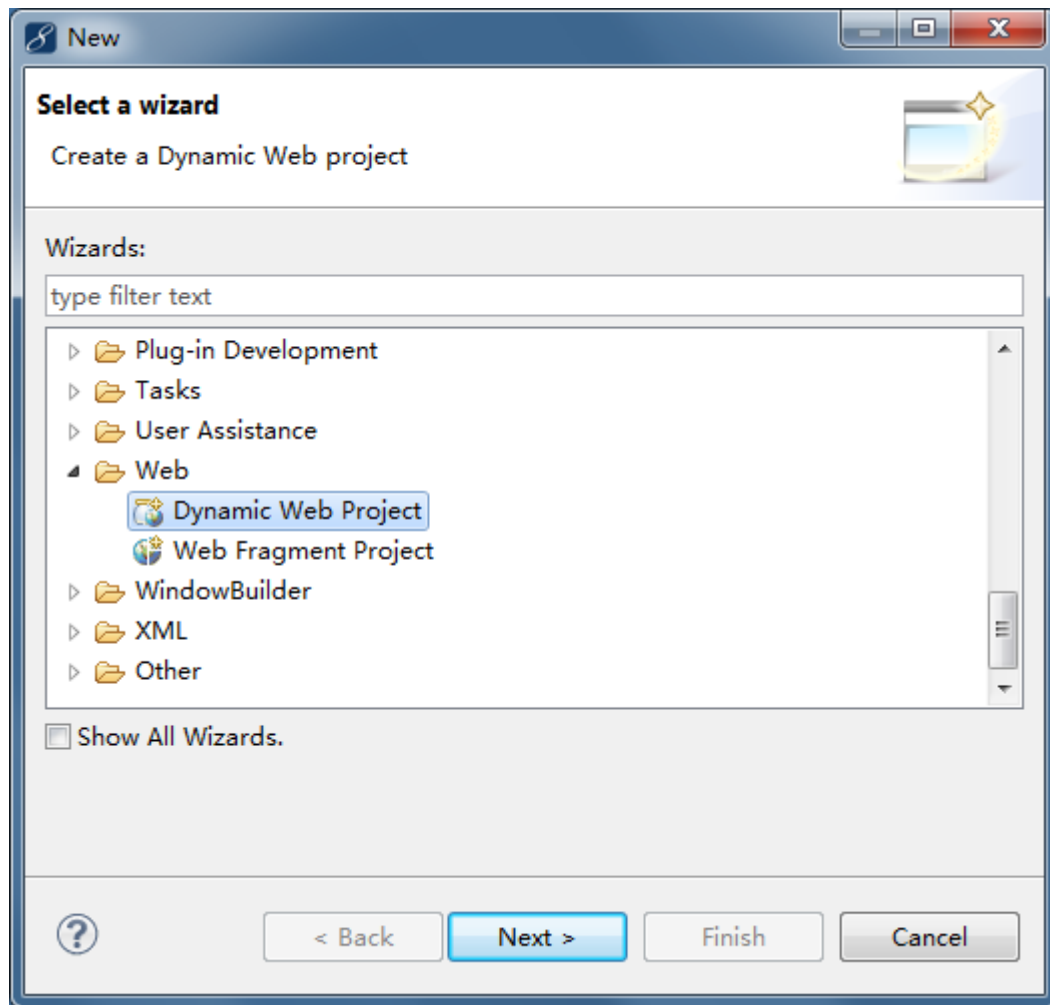
其它环境：

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

**New Dynamic Web Project**

**Dynamic Web Project**

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:  项目名称

Project location

☒ Use default location

Location:  默认的项目路径

Target runtime

目标服务器 (Tomcat)

Dynamic web module version

Configuration

A good starting point for working with Apache Tomcat v8.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

☐ Add project to an EAR

EAR project name:

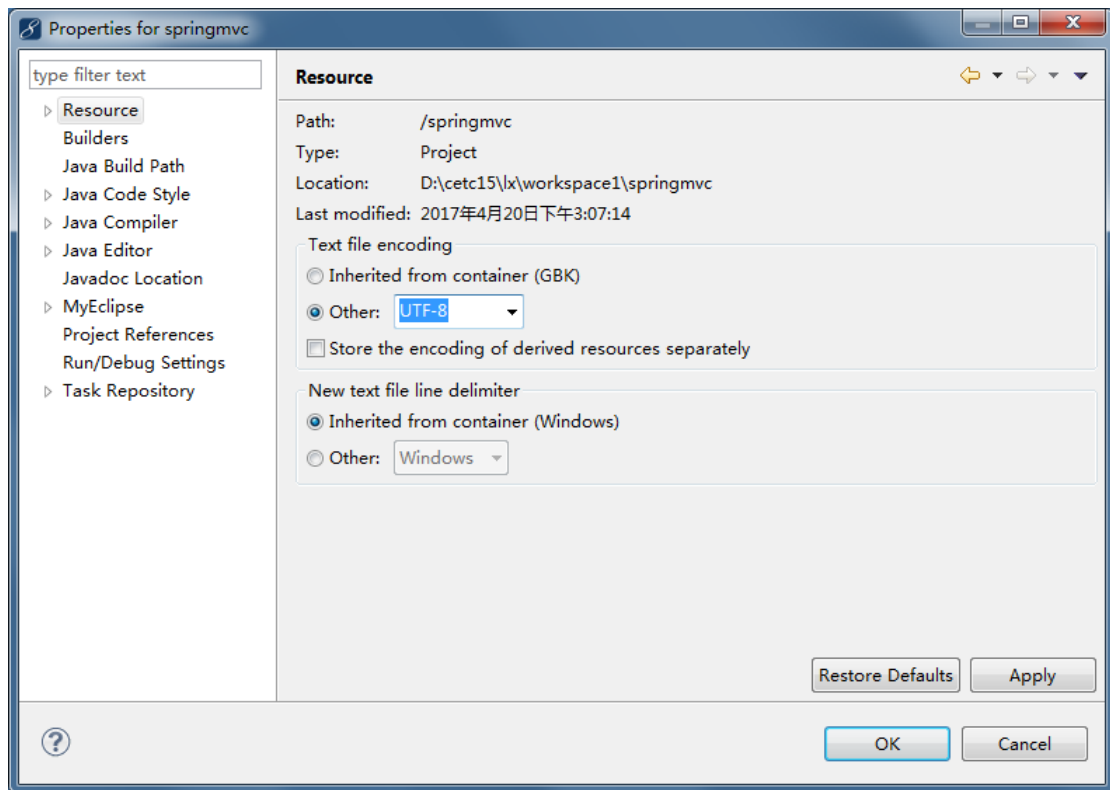
Working sets

☐ Add project to working sets

Working sets:

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。



## 2. 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。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  id="WebApp_ID">
  <!-- 项目名称 -->
  <display-name></display-name>
</web-app>
```

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。

## 3. 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。

```

<%@ page language="java" import="java.util.*" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+ "://" + request.getServerName() + ":" + request.getServerPort() + path + "/" ;
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<base href="<%=basePath%>">

<title>My JSP 'index.jsp' starting page</title>

<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
<meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
<meta http-equiv="description" content="This is my page">
<meta charset="UTF-8" />
<!--
<link rel="stylesheet" type="text/css" href="styles.css"
-->

</head>

<body>
恭喜，web项目已经成功搭建！。 <br>
</body>
</html>

```

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。

#### 4. 集成 SpringMVC

需要在 web.xml 里添加下面的配置

##### a) 配置监听器

```

<listener>
<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<listener>
<listener-class>org.springframework.web.util.IntrospectorCleanupListener</listener-class>
</listener>

```

##### b) 配置过滤器，解决 POST 乱码问题

```

<filter>
<filter-name>encoding</filter-name>
<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
<init-param>
<param-name>encoding</param-name>
<param-value>UTF-8</param-value>
</init-param>
</filter>
<filter-mapping>
<filter-name>encoding</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>

```

##### c) 配置 SpringMVC 分发器，拦截所有请求

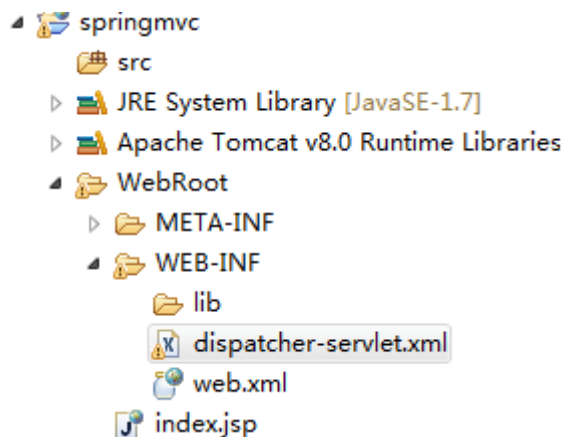
```

<servlet>
  <servlet-name>springmvc</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>namespace</param-name>
    <param-value>dispatcher-servlet</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>springmvc</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>

```

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。

#### d) 编写 dispatcher-servlet.xml

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

配置如下：

```

<?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">

    <!-- 开启注解模式驱动 -->
    <mvc:annotation-driven></mvc:annotation-driven>
    <!-- 扫描 -->
    <context:component-scan base-package="com.springmvc.*"></context:component-scan>
    <!-- 静态资源过滤 -->
    <mvc:resources location="/resources/" mapping="/resources/**" />
    <!-- 视图渲染 jsp/freemaker/velocity -->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <!-- 制定页面存放的路径 -->
        <property name="prefix" value="/WEB-INF/pages"></property>
        <!-- 文件的后缀 -->
        <property name="suffix" value=".jsp"></property>
    </bean>
</beans>

```

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。

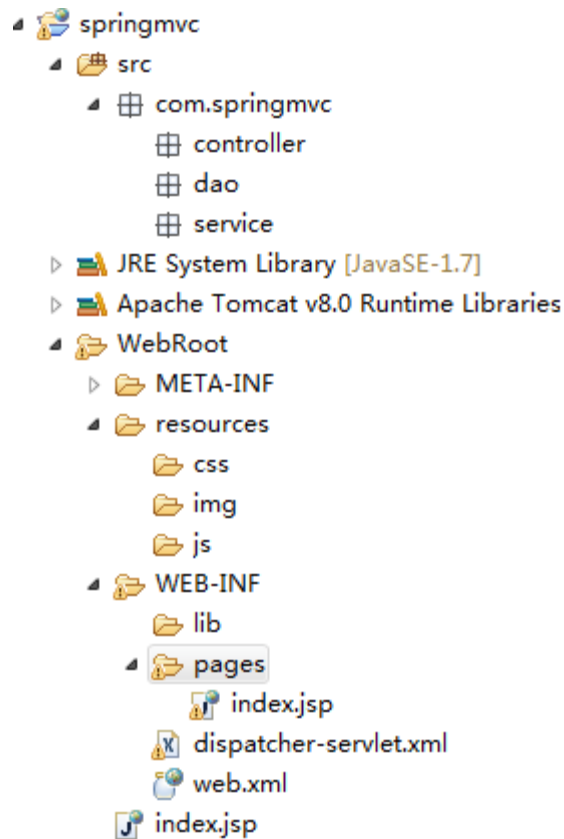
首先是 Java 文件的目录，在 `src` 处右键，新建一个 `com` 包，再在里面建一个 `springmvc` 包。

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。

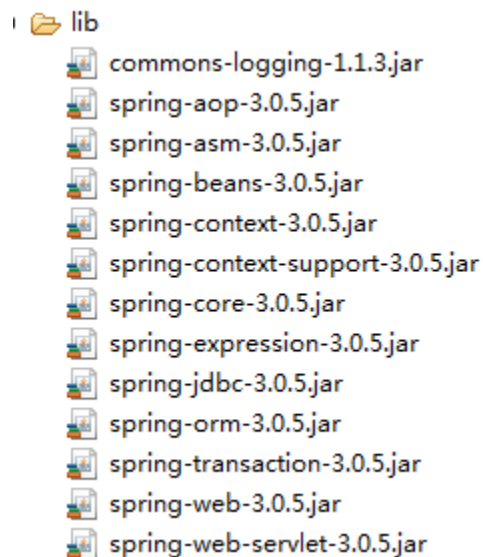
接下来，来建静态资源的目录。在 `WebRoot` 目录下新建一个 `resources` 文件夹。然后顺便把 `js`, `css`, `img` 的文件夹都建一下，这里就存放我们的静态资源文件。

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。



## 5. 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

```
java.io.FileNotFoundException: Could not open ServletContext resource [/WEB-INF/applicationContext.xml]
```

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。

```
<?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:aop="http://www.springframework.org/schema/aop"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:util="http://www.springframework.org/schema/util"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-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">

</beans>
```

## 6. 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。



New Java Class

Java Class

Create a new Java class.



Source folder:

springmvc/src

Browse...

Package:

com.springmvc.controller

Browse...

☐ Enclosing type:

Browse...

Name:

ViewController

Modifiers:

☒ public

☐ default

☐ private

☐ protected

☐ abstract

☐ final

☐ static

Superclass:

java.lang.Object

Browse...

Interfaces:

Add...

Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments



Finish

Cancel

```

package com.springmvc.controller;

import javax.servlet.http.HttpServletRequest;

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

@Controller
public class ViewController {

    @RequestMapping("/view")
    public ModelAndView view(HttpServletRequest request) {
        String path = request.getParameter("path") + "";
        ModelAndView mav = new ModelAndView();
        mav.setViewName(path);
        return mav;
    }
}

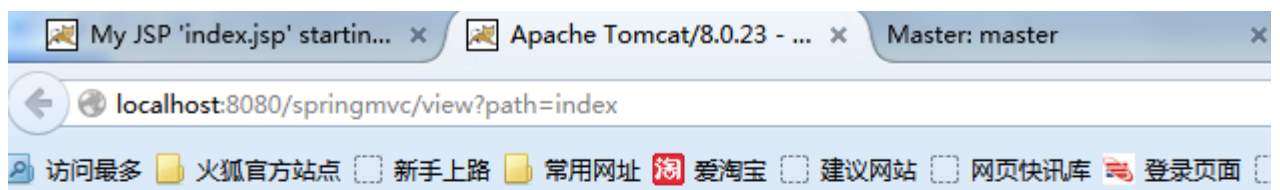
```

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：



## HTTP Status 404 - /springmvc/WEB-INF/pagesindex.jsp

**type** Status report

**message** /springmvc/WEB-INF/pagesindex.jsp

**description** The requested resource is not available.

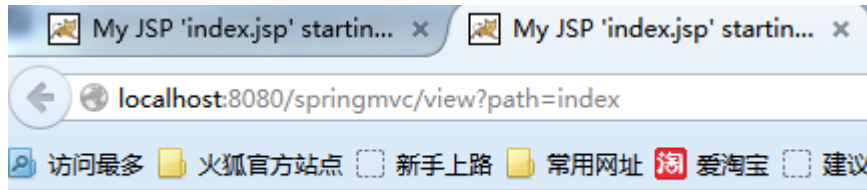
### Apache Tomcat/8.0.23

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 `/`。

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

保存后重启 Tomcat，一切继续！



恭喜，web项目已经成功搭建！.

成功了！

## 7. 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。

```
<%@ page language="java" import="java.util.*" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+ "://" + request.getServerName() + ":" + request.getServerPort() + path + "/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<base href="<%=basePath%>">

<title>My JSP 'index.jsp' starting page</title>

<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
<meta http-equiv="keywords" content="keyword1, keyword2, keyword3">
<meta http-equiv="description" content="This is my page">
<meta charset="UTF-8" />
<!--
<link rel="stylesheet" type="text/css" href="styles.css">
-->
<style>
body {
background: url(${contextPath}/resources/img/Chrysanthemum.jpg);
background-size: 100% 100%;
}
</style>

</head>

<body>
<!-- 恭喜，web项目已经成功搭建！. <br> -->
</body>
</html>
```

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

成功了！