网页静态化

什么是freemarker

> FreeMarker是一个用Java语言编写的模板引擎，它基于模板来生成文本输出。FreeMarker与Web容器无关，即在Web运行时，它并不知道Servlet或HTTP。它不仅可以用作表现层的实现技术，而且还可以用于生成XML，JSP或Java 等。 主要用Freemarker做静态页面或是页面展示

Freemarker的使用方法

> Maven工程添加依赖

```
<dependency>
  <groupId>org.freemarker</groupId>
  <artifactId>freemarker</artifactId>
  <version>2.3.23</version>
</dependency>
```

原理相当于el表达式

使用步骤

```
使用步骤:
第一步：创建一个Configuration对象，直接new一个对象。构造方法的参数就是freemarker对于的版本号。
第二步：设置模板文件所在的路径。
第三步：设置模板文件使用的字符集。一般就是utf-8.
第四步：加载一个模板，创建一个模板对象。
第五步：创建一个模板使用的数据集，可以是pojo也可以是map。一般是Map。
第六步：创建一个Writer对象，一般创建一FileWriter对象，指定生成的文件名。
第七步：调用模板对象的process方法输出文件。
第八步：关闭流.
```

事例模板,文件后缀名**FTL**

```
模板:
${hello}
```

```
    // 第一步：创建一个Configuration对象，直接new一个对象。构造方法的参数就是freemarker对于的版本号。
    Configuration configuration = new Configuration(Configuration.getVersion());
    // 第二步：设置模板文件所在的路径。
    configuration.setDirectoryForTemplateLoading(new File("D:/ftl"));
    // 第三步：设置模板文件使用的字符集。一般就是utf-8.
    configuration.setDefaultEncoding("utf-8");
    // 第四步：加载一个模板，创建一个模板对象。
```

```
    Template template = configuration.getTemplate("hello.ftl");
    // 第五步：创建一个模板使用的数据集，可以是pojo也可以是map。一般是Map。
    Map dataModel = new HashMap<>();
    //向数据集中添加数据
    dataModel.put("hello", "this is my first freemarker test.");
    // 第六步：创建一个Writer对象，一般创建一FileWriter对象，指定生成的文件名。
    Writer out = new FileWriter(new File("D:/out/hello.html"));
    // 第七步：调用模板对象的process方法输出文件。
    template.process(dataModel, out);
    // 第八步：关闭流。
    out.close();
```

模板的语法

访问map中的key

```
${key}
```

访问pojo中的属性

```
${classes.classname}
```

取集合中的数据

```
<#list studentList as student> 遍历studentList 每次对象存为student
${student.id}/${studnet.name}
</#list>
```

取循环中的下标

```
<#list studentList as student>
    ${student_index} 注意是_ 不是.
</#list>
```

判断

```
<#if student_index % 2 == 0>
<#else> 这里面写的和if一致
</#if>
```

日期类型格式化

```
${date?date}
${date?time}
${date?datetime}
${date?string(parten)}  ${date?string("yyyy/MM/dd HH:mm:ss")}
```

Null值的处理

```
${val!"val的值为null"} 如果返回true代表没有值
判断val的值是否为null
<#if val??>  这里和上面不一样,如果返回true代表有值
val中有内容
<#else>
val的值为null
</#if>
```

Include标签

```
<#include “模板名称”>
```

Freemarker整合spring

1引入jar包

```
需要spring-context-support
```

2配置文件

```
<?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:dubbo="http://code.alibabatech.com/schema/dubbo"
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/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd
        http://code.alibabatech.com/schema/dubbo
http://code.alibabatech.com/schema/dubbo/dubbo.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <bean id="freemarkerConfig"

class="org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer">
        <property name="templateLoaderPath" value="/WEB-INF/ftl/" />
        <property name="defaultEncoding" value="UTF-8" />
    </bean>

</beans>
```

3代码测试

```
public String genHtml()throws Exception {
    // 1、从spring容器中获得FreeMarkerConfigurer对象。
    // 2、从FreeMarkerConfigurer对象中获得Configuration对象。
    Configuration configuration = freeMarkerConfigurer.getConfiguration();
    // 3、使用Configuration对象获得Template对象。
    Template template = configuration.getTemplate("hello.ftl");
    // 4、创建数据集
    Map dataModel = new HashMap<>();
    dataModel.put("hello", "1000");
    // 5、创建输出文件的Writer对象。
    Writer out = new FileWriter(new File("D:out/spring-freemarker.html"));
    // 6、调用模板对象的process方法，生成文件。
    template.process(dataModel, out);
    // 7、关闭流。
    out.close();
    return "OK";
}
```