# SpringBoot

2014年启动基于spring4.0。

目前springboot进入2.0时代，相对稳定版本为1.5.8

**Springboot的优点**

1. 简化JavaEE的应用开发，零配置，快速集成主流框架
2. 使用嵌入式的servlet容器，打成jar包，使用java -jar命令直接执行，无需外部的web容器
3. 提供各种启动器，自动导入各种依赖和控制版本
4. 提供大量默认的配置，简化开发，默认配置、默认值可以修改
5. 减少大量的xml配置文件
6. 提供运行时监控
7. 天然集成云计算。

**Springboot的缺点**

入门容易，深入难。原因：使用大量的默认配置，难于理解底层.

## 入门

官方入门demo <https://start.spring.io/>

或者

Idea新建项目from spring initializr

### 热部署

1、配置依赖

<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-devtools</**artifactId**>  
 <**optional**>true</**optional**>  
</**dependency**>

2、设置相关参数

Application.properties

*#禁止thymeleaf缓存（建议：开发环境设置为false，生成环境设置为true）***spring.thymeleaf.cache**=**false  
  
spring.devtools.restart.enabled**=**true***#添加那个目录的文件需要restart***spring.devtools.restart.additional-paths**=**src/main/java***#排除那个目录的文件不需要restart  
#spring.devtools.restart.exclude=static/\*\*,public/\*\**

*#延时启动，避免出现因为编译不及时启动造成的404***spring.devtools.restart.poll-interval**=**3000ms  
spring.devtools.restart.quiet-period**=**2999ms**

### 资源文件映射实体

1. 配置依赖

<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-configuration-processor</**artifactId**>  
 <**optional**>true</**optional**>  
</**dependency**>

1. 新建属性文件jdbc.properties

**com.jdbc.url**=**www.baidu.com  
com.jdbc.userName**=**zhangsan  
com.jdbc.password**=**password  
com.jdbc.driver**=**jdbc.mysql.Driver**

1. 新建属性实体类 Recource

@Data  
@Configuration  
@PropertySource(value = **"classpath:jdbc.properties"**)  
@ConfigurationProperties(prefix = **"com.jdbc"**)  
**public class** Resource {  
 **private** String **url**;  
 **private** String **userName**;  
 **private** String **password**;  
 **private** String **driver**;  
}

1. 使用资源实体类

@Autowired  
**private** Resource **resource**;

### Tomcat和服务端配置

*###############################  
#Server 服务端相关配置  
###############################  
#配置访问端口***server.port**=**8888***#配置context-path项目名***server.servlet.context-path**=**/demo-spring-boot***#session最大超时时间***server.servlet.session.timeout**=**60***#启动服务器绑定IP，特殊情况下使用  
#server.address=192.168.2.225都  
###############################  
#Server tomcat相关配置  
###############################  
#tomcat的URI编码***server.tomcat.uri-encoding**=**utf-8***#tomcat的最大线程数量,最大200***server.tomcat.max-threads**=**150**

### 整合模板

#### freemarker

1. 配置依赖

<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-freemarker</**artifactId**>  
</**dependency**>

1. 设置相关参数application.properties

*###############################  
#freemarker 静态资源配置  
###############################  
#设定ftl文件路径***spring.freemarker.template-loader-path**=**classpath:/freemaker***#关闭缓存，即时刷新，上线生产环境改为true***spring.freemarker.cache**=**false  
spring.freemarker.charset**=**utf-8  
spring.freemarker.check-template-location**=**true  
spring.freemarker.content-type**=**text/html  
spring.freemarker.expose-request-attributes**=**true  
spring.freemarker.expose-session-attributes**=**true  
spring.freemarker.request-context-attribute**=**request  
spring.freemarker.suffix**=**.ftl**

1. 编写.ftl模板文件，类似jstl

<**html**>  
<**body**>  
<**h1**>我是index</**h1**>  
<**h1**>**${resource**.**url}**</**h1**>  
<**h1**>**${resource**.**userName}**</**h1**>  
<**h1**>**${resource**.**password}**</**h1**>  
<**h1**>**${resource**.**driver}**</**h1**>  
</**body**>  
</**html**>

1. 编写controller，访问模板文件

**package** com.wxx.demo.controller;  
**import** com.wxx.demo.entity.Resource;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.ui.ModelMap;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
*/\*\*  
 \** ***@author*** *wangxinxin07  
 \** ***@date*** *2020/1/30  
 \*/*@RequestMapping(**"/freemaker"**)  
@Controller  
**public class** FreemakerController {  
 @Autowired  
 **private** Resource **resource**;  
 @RequestMapping(**"/index"**)  
 **public** String index(ModelMap modelMap) {  
 modelMap.addAttribute(**"resource"**, **resource**);  
 **return "index"**;  
 }  
 @RequestMapping(**"/center"**)  
 **public** String center() {  
 **return "/center/center"**;  
 }  
}

#### thymeleaf

1. 配置依赖

<**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-thymeleaf</**artifactId**>  
</**dependency**>

1. 设置相关参数application.properties

*###############################  
#thymeleaf 静态资源配置  
###############################  
#设定文件路径***spring.thymeleaf.prefix**=**classpath:/template/  
spring.thymeleaf.suffix**=**.html***#关闭缓存，即时刷新，上线生产环境改为true***spring.thymeleaf.cache**=**false  
spring.thymeleaf.encoding**=**utf-8  
spring.thymeleaf.mode**=**HTML5  
spring.thymeleaf.servlet.content-type**=**text/html**

1. 编写html模板文件

<**html xmlns:th="http://www.w3.org/1999/xhtml"**>  
<**body**>  
thymeleaf 模板引擎  
<**h1 th:text="${name}"**>hello world!!</**h1**>  
</**body**>  
</**html**>

1. 编写controller

**package** com.wxx.demo.controller;  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.ui.ModelMap;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
  
*/\*\*  
 \** ***@author*** *wangxinxin07  
 \** ***@date*** *2020/2/4  
 \*/*@RequestMapping(**"/th"**)  
@Controller  
**public class** ThymeleafController {  
 @RequestMapping(**"/index"**)  
 **public** String index(ModelMap map){  
 map.addAttribute(**"name"**, **"zhangsan"**);  
 **return "thymeleaf/index"**;  
 }  
 @RequestMapping(**"/center"**)  
 **public** String center(){  
 **return "thymeleaf/center/center"**;  
   
}

### 全局异常配置

配置异常handler类，区分是否ajax

**package** com.wxx.demo.handler;  
**import** com.wxx.demo.entity.common.RestReturnJson;  
**import** lombok.extern.slf4j.Slf4j;  
**import** org.springframework.web.bind.annotation.ControllerAdvice;  
**import** org.springframework.web.bind.annotation.ExceptionHandler;  
**import** org.springframework.web.bind.annotation.RestControllerAdvice;  
**import** org.springframework.web.servlet.ModelAndView;  
**import** javax.servlet.http.HttpServletRequest;  
**import** javax.servlet.http.HttpServletResponse;  
*/\*\*  
 \** ***@author*** *wangxinxin07  
 \** ***@date*** *2020/2/4  
 \*/*@RestControllerAdvice  
@Slf4j  
**public class** ControllerExceptionHandler {  
 @ExceptionHandler(value = Exception.**class**)  
 **public** Object handle(HttpServletRequest request, HttpServletResponse response, Exception e) {  
 e.printStackTrace();  
 **if** (isaJax(request)) {  
 System.***out***.println(**"这是ajax请求错误"**);  
 **return** RestReturnJson.*error*(**"错误原因："** + e.getMessage());  
 } **else** {  
 System.***out***.println(**"这是页面访问错误"**);  
 ModelAndView modelAndView = **new** ModelAndView();  
 modelAndView.addObject(**"exeption"**, e);  
 modelAndView.addObject(**"url"**, request.getRequestURL());  
 modelAndView.addObject(**"uri"**, request.getRequestURI());  
 modelAndView.setViewName(**"error"**);  
 **return** modelAndView;  
 }  
 }  
 **private boolean** isaJax(HttpServletRequest request) {  
 String requestedWith = request.getHeader(**"x-requested-with"**);  
 **if** (requestedWith != **null** && requestedWith.equalsIgnoreCase(**"XMLHttpRequest"**)) {  
 **return true**;  
 } **else** {  
 **return false**;  
 }  
 }  
}

## 整合mybatis

## 整合redis

## 整合定时任务

1. 开启定时任务，自动进行扫描

*//开启定时任务，用在application启动类上*@EnableScheduling

1. 定义job类，交给spring容器管理

@Component

1. 方法上使用定时表达式

*//每隔3秒执行任务*@Scheduled(fixedRate = 3000)

*//cron表达式*@Scheduled(cron = **"0/3 \* \* \* \* ?"**)

## 整合异步任务

1. 开启异步任务

*//开启异步调用方法*@EnableAsync

1. 定义异步task类，交给spring容器管理

@Component  
**public class** AsyncFirstTask {}

1. 实现异步方法

@Async  
**public** Future<Boolean> doTask11() **throws** Exception {  
 Thread.*sleep*(1000);  
 System.***out***.println(**"dotask11执行完毕"**);  
 **return new** AsyncResult<>(**true**);  
}

使用场景

1. 发送短信
2. 发送邮件
3. APP消息推送

## 整合拦截器