链滴

翻新第一个私活 CMS 项目

作者：crick77
原文链接：https://ld246.com/article/1524327067337
来源网站：链滴

1. 项目背景

项目地址: https://github.com/wangyuheng/Haro-CMS

最近因为换电脑整理资料的过程中，发现了自己之前写过的一些项目，现在准备拿出来升级一下，构自己的技术体系。看着三个月前的代码，已经似曾相识了，那三年前的呢?

这个项目是用来做企业展示的简易CMS系统，后台管理员可以编辑展示内容，发布新闻；访问用户可查看企业信息，并反馈建议。这个系统当时应该熬了几个通宵，因为5000大洋实在太吸引人。。而且是三个人分。。。人的时间好像是越往后越值钱，现在看可能觉得不值，但还是很感谢熬夜编码的自。

项目当时考虑了很多：分布式，前后端分离，甚至是saas化，希望根据用户反馈，或者再接到类似的目，可以进一步完善，但是并没有什么后续。所以项目历史就是一个基于web.xml配置的spring单机用。

1.1 依赖项目

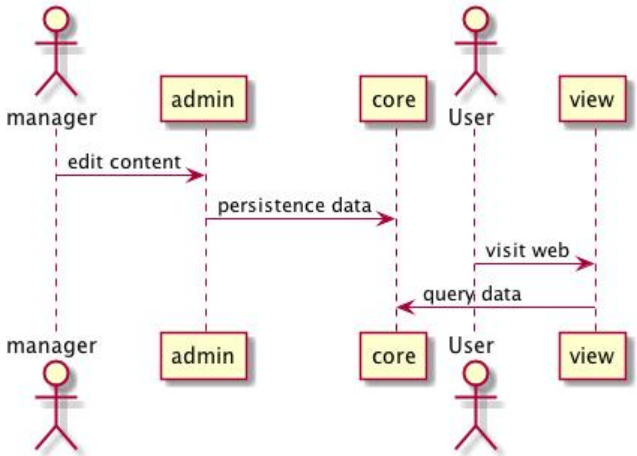
1. spring-boot
2. h2
3. gson

目前将其升级为spring-boot应用，并且为了开发演示方便，开发环境使用了H2数据库。一直觉得，目不需要修改任何配置就能跑起来很重要。

2. 项目code

企业信息展示，分为3个module

1. **admin** 管理人员编辑展示内容
2. **view** 展示企业信息及新闻
3. **core** 业务数据层



新版spring boot已不推荐使用jsp，带来很多不便，如： 不能直接运行java， 需要使用maven spring boot 插件运行。

mvn spring-boot:run

admin 和 view 只负责业务渲染与鉴权，业务操作放在core中，方便后期进行前后端分离。
生产环境数据库使用mysql，为了方便演示，开发环境使用H2内嵌数据库。

admin

view

原文链接：翻新第一个私活 CMS 项目

3 问题

之前的项目基于spring开发，采用web.xml配置。spring-boot通过约定大于配置，极大的简化了这分工作，但是有时候又会因为不熟悉带来迷茫。明明我没有这个配置，怎么就生效了。。而且jsp又推荐使用，所以虽然大部分是在删代码，单还是遇到了很多问题。

3.1 设置视图解析器InternalResourceViewResolver

原项目在mvc-dispatcher-servlet.xml文件中配置

```
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/pages/"/>
    <property name="suffix" value=".jsp"/>
</bean>
```

spring-boot提供了默认配置prefix = "spring.mvc"，详情可以参看org.springframework.boot.autoconfigure.web.servlet.WebMvcProperties类， 并在org.springframework.boot.autoconfigure.web servlet.WebMvcAutoConfiguration类中自动注入。

```
@Configuration
@ConditionalOnWebApplication(type = Type.SERVLET)
@ConditionalOnClass({ Servlet.class, DispatcherServlet.class, WebMvcConfigurer.class })
@ConditionalOnMissingBean(WebMvcConfigurationSupport.class)
@AutoConfigureOrder(Ordered.HIGHEST_PRECEDENCE + 10)
@AutoConfigureAfter({ DispatcherServletAutoConfiguration.class,
    ValidationAutoConfiguration.class })
public class WebMvcAutoConfiguration {
...
    @Bean
    @ConditionalOnMissingBean
    public InternalResourceViewResolver defaultViewResolver() {
        InternalResourceViewResolver resolver = new InternalResourceViewResolver();
        resolver.setPrefix(this.mvcProperties.getView().getPrefix());
        resolver.setSuffix(this.mvcProperties.getView().getSuffix());
        return resolver;
    }
...
}
```

所以只需在application.properties中添加配置项

```
spring.mvc.view.prefix=/WEB-INF/pages/
spring.mvc.view.suffix=.jsp
```

3.2 静态资源请求

未实现前后端分离的项目，css、js等静态资源仍保存在项目中。spring-boot建议保存在resource/static目录下，并通过spring.resources.static-locations属性设置目录位置，并且需要制定mapping规。

```
spring.mvc.static-path-pattern=/static/**
spring.resources.static-locations=classpath:/static/
```

3.3 spring-boot 运行war项目

需要添加tomcat-embed依赖，并且设置servlet初始化。如果使用jsp标签，一般也需要添加jstl依赖。

```
<dependencies>
    <dependency>
        <groupId>jstl</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>
    <!-- Provided -->
    <dependency>
        <groupId>org.apache.tomcat.embed</groupId>
        <artifactId>tomcat-embed-jasper</artifactId>
        <scope>provided</scope>
    </dependency>
</dependencies>
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <executions>
                <execution>
                    <goals>
                        <goal>repackage</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

```
@SpringBootApplication
@ComponentScan(basePackages={"wang.crick.business.haro"})
public class ViewAPP extends SpringBootServletInitializer {

    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(ViewAPP.class);
    }

    public static void main(String[] args) {
        SpringApplication.run(ViewAPP.class, args);
    }

}
```

需要通过spring-boot maven插件运行。

```
mvn spring-boot:run
```

3.4 自定义请求后缀匹配

旧项目中为了提高seo通过servlet-mapping指定请求路径为.html

```
<servlet>
    <servlet-name>mvc-dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
    <servlet-name>mvc-dispatcher</servlet-name>
    <url-pattern>*.html</url-pattern>
</servlet-mapping>
```

spring-boot在org.springframework.boot.autoconfigure.web.ServerProperties提供了servlet相配置，可以通过application.properties配置

```
server.servlet.path=*.html
```

但是目前希望忽略后缀，即请求index和index.html都可以路由到对应的页面，此时可以通过设置pat match属性实现

```
/**

 * Whether to use suffix pattern match (".*") when matching patterns to requests.
 * If enabled a method mapped to "/users" also matches to "/users.*".
 */

spring.mvc.pathmatch.use-suffix-pattern=true
```

3.5 使用内存数据库H2

为了让程序可以“无痛”运行，在代码中使用了基于内存的H2数据库，生产环境可以考虑mysql等关型数据库，只需修改对应的配置即可。

需要添加H2依赖，引入驱动。

```
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
    </dependency>
```

在配置文件application.properties中直接配置路径、驱动、以及账户信息

```
spring.datasource.url=jdbc:h2:mem:~/.h2/haro
spring.datasource.schema=classpath:db/schema.sql
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.username=root
spring.datasource.password=123456
```

db/schema.sql放置在resource目录下，保存了DDL以及开发环境所需数据。DDL支持情况为标准sq语句，部分工具导出的脚本需要简单修改。在程序启动过程中，会初始化执行，如果有异常，会在控台中有详细提示。