atitit mvc的使用总结v4

mvc框架有  xxp模式（jsp php aspx等），tp springmvc struts, vue等。

项目中用到的有struts,springmvc,vue这三个mvc框架

Catalog

[1. #注册总体拦截器 1](#_Toc4189)

[拦截器模式vs 指定公共index分发器模式 1](#_Toc24038)

[##Nginx拦截器 1](#_Toc30507)

[##Xxp模式是约定大于配置模型，不需要特殊实现 1](#_Toc21678)

[##Struts设置 1](#_Toc29532)

[2. ##springmvc2.5.6 2](#_Toc6176)

[## springmvc2.5兼容Jdk8补丁修正 2](#_Toc29728)

[Vue的配置 2](#_Toc23333)

[##Tp5 直接访问公共index 3](#_Toc26418)

[##LARAVEL 貌似是index公共 3](#_Toc10537)

[Java项目也可使用公共index url指定模式 3](#_Toc29852)

[3. #约定vs注册路由 请求url与对应的响应代码块 3](#_Toc21290)

[4. 设置显示范围 路由出口渲染 5](#_Toc22014)

[访问url模式 5](#_Toc3708)

[OTHER 5](#_Toc3265)

[Ref 5](#_Toc6820)

# #注册总体拦截器

## 拦截器模式vs 指定公共index分发器模式

一搬根据拦截扩展名判断，mime内容注册机制，不同的扩展名使用不同的mvc框架拦截解析

  也有根据前缀路径判断的

也可使用公共index 分发器（php系列使用这种方法）

## ##Nginx拦截器

## ##Xxp模式是约定大于配置模型，不需要特殊实现

## ##Struts设置

web.xml中添加如下代码，拦截对应的请求，

<filter>

<filter-name>struts</filter-name>

<filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class> </filter>

<filter-mapping>

<filter-name>struts</filter-name> <url-pattern>/\*.do</url-pattern> </filter-mapping>

## ##springmvc2.5.6

<!--spr mvc cfg name just same with this svltname -->

  <servlet>

     <servlet-name>spring-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>spring-mvc-dispatcher</servlet-name>

     <url-pattern>\*.spr</url-pattern>

  </servlet-mapping>

Note::Spring2.5.6 有bug不兼容jdk8 ,需要小调整

## ## springmvc2.5兼容Jdk8补丁修正

<!--spr mvc v100 for jdk8-->

<servlet>

  <servlet-name>spring-mvc-dispatcherV100</servlet-name>

  <servlet-class>

     org.springframework.web.servlet.DispatcherServletV100

  </servlet-class>

  <load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

  <servlet-name>spring-mvc-dispatcherV100</servlet-name>

  <url-pattern>\*.ati</url-pattern>

</servlet-mapping>

## Vue的配置

// 2. 定义路由 客户端hash 路由

//vue-router 默认 hash 模式 —— 使用 URL 的 hash 来模拟一个完整的 URL，于是当 URL 改变时，页面不会重新加载。另外一种使用h5 history模式

 //  file:///D:/xxx.html#/bar

const routes = [

    { path: '/foo', component: { template: '<div>foo............</div>' } },

    { path: '/bar', component:  { template: '<div>bar...................</div>' } }

]

var Vue1 = new Vue({

  /\*

router: new VueRouter({

    routes // （缩写）相当于 routes: routes

})

,

\*/

  el: '#app',

  data: {

    message: 'Hello Vue!',objlist:[],userlist:[]

  }

})

## ##Tp5 直接访问公共index

· http://localhost/project\_name/public/index.php/Index/Test/index 可以发现页面上输出了 **Hello World!**

· · 这里你应该就发现了URL的规律了，在index.php后面加上**/模块名/控制器名/方法名**就可以调用相应的方法了。

## ##LARAVEL 貌似是index公共

## Java项目也可使用公共index url指定模式

请求全都指向一个公共的index，然后在区分发

# #约定vs注册路由 请求url与对应的响应代码块

有约定模式和注册配置模式

Xxp模式是约定大于配置模型，不需要特殊实现，使用url路径直接对应

Struts需要在struts.xml里面去配置 url与java代码块方法对应关系

<action name="xxxapi" class="xx" method="mtd2"></action>

Springmvc在其配置文件配置也是或者注解扫描实现

Vue也需要类似的配置

const routes = [

    { path: '/foo', component: { template: '<div>foo............</div>' } },

    { path: '/bar', component:  { template: '<div>bar...................</div>' } }

]

Tp5是约定模式。。。

· http://localhost/project\_name/public/index.php/Index/Test/index 可以发现页面上输出了 **Hello World!**

· · 这里你应该就发现了URL的规律了，在index.php后面加上**/模块名/控制器名/方法名**就可以调用相应的方法了。

·   
 namespace app\index\controller;

class Test {

public function index(){

return 'Hello World!';

}}

LaraVEL 注册模式

我们在 app/Http/routes.php 中注册了我们首页的路由：

Route::get('/','ArticleController@index');

class ArticleController extends Controller

{

public function index()

{

$articles = Article::all();

return $articles;

}

————————————————

使用配置url请求表映射，url映射方法引用，或使用反射法，或使用动态调用代码法

# 设置显示范围 路由出口渲染

后端mvc默认显示到所有。需要配置头跨域。。。如需要设置显示范围，也可特定json标签指明

Vue配置

 <!-- 路由出口 -->

  <!-- 路由匹配到的组件（代码块）将渲染在这里 -->

  <router-view></router-view>

# 访问url模式

原始模式一般是  **Host/** index?pkg=xxx&controler=xxx&fun=xxxx

**或者url转写为 Host/模块名/控制器名/方法名 模式**

**Host/某个注册url**

# OTHER

LARAVEL 1. 注册路由

来到 app/Http/routes.php 中，我们增加一个路由：

Route::get('articles/{id}','ArticleController@show');

上面的路由 articles/{id} 指定我们需要加载 ArticleController 中的 show() 方法。这里需要注意的是 {id} 这个表达：这是表示 id 是一个路由变量，也就是当我们访问类似下面这两个路由的时候：

————————————————

# Ref

快速上手ThinkPHP 5.0 - 简书