**class** Route **extends** Facade

实例化think/route

实例化domain extends rulegroup extends rule

$this->group = new domain

Think/route->get(‘think’,closure)

$this->**group**->addRule（‘think’,closure）{

**new** RuleItem($this->**router(think/route实例)**, $this（domain实例）, $name（null）, $rule（‘think’）, $route（closure）, $method(get), $option, $pattern);

**class** RuleItem **extends** Rule（ruleitem是每次都实例化）

setRule

$this->setRuleName()//*设置路由标识 用于URL反解生成*

Container::*get*(**'rule\_name'**)->setRule($this->**rule**, $this(ruleitem));（rulename是单例模式）

**class** RuleName{

**public function** setRule($rule(‘think’), $route)  
{  
 $this->**rule**[$route->getDomain()(‘localhost’)][$rule(‘think’)][$route->getMethod()(‘get’)] = $route（ruleitem）;(rulename里有ruleitem)  
}

$this->addRuleItem($ruleItem, $method);（rulegroup）

$this->rules[‘get’][] = $ruleitem

}(domain/group里面有ruleitem)

Route::*rule*(**'trace'**,**'index/index/testTrace'**,**'GET'**)->middleware(**'Auth:bigGuy'**);

**public function think\route\**rule

$this->**group**->addRule(‘trace’,’index/index/testTrace’,get)

$ruleItem = **new** RuleItem($this->**router(think/route实例)**, $this(domain实例), $name(**index/index/testTrace**), $rule(trace), $route(**index/index/testTrace**), $method(get), $option, $pattern);

$this->setRule($rule);

*// 生成路由标识的快捷访问*$this->setRuleName();

$value = [$this->**rule**, $vars, $this->**parent**->getDomain(), $suffix, $this->**method**];  
  
Container::*get*(**'rule\_name'**)->set($name, $value, $first);

$this->**item**[$name][] = $value;

注：name=**index/index/testTrace**

**Value=[trace,0,localhost]**

Container::*get*(**'rule\_name'**)->setRule($this->**rule(trace)**, $this(RuleItem));

$this->**rule**[$route->getDomain()][$rule][$route->getMethod()] = $route;

(domain)$this->addRuleItem($ruleItem, $method);

(domain)$this->**rules**[$method][] = $rule(item);

返回ruleitem实例化

**总结一下：每次实例化RuleItem（非单例，一个ruleitem代表一个定义的rule规则）,然后实例化RuleName（单例）,调用setRule(trace,RuleItem实例),然后设置：**$this->**rule**[$route->getDomain()‘localhost’][$rule’trace’][$route->getMethod()’get’] = $route(ruleItem);

即:domain里有RuleItem,RuleName里有RuleItem

Route:check()

**Domain->check()**

**//插入中间件（这里是域名中间件）**

Domain->getMethodRules();返回的就是上面那个

(domain)$this->**rules**[$method][] = $rule(item);

然后调用item的check方法

*// 检查分组路由***foreach** ($rules **as** $key => $item) {  
 $result = $item->check($request, $url, $completeMatch);

$item->checkrule();

$match = $this->match($url, $option, $completeMatch);

检查是否匹配

解析路由规则

parseRule（）

*// 发起路由调度***return** $this->dispatch($request, $route, $option);

(如果是闭包函数，则执行之)

*// 路由到模块/控制器/操作*$result = $this->dispatchModule($request, $route);

*// 路由到模块/控制器/操作***return new** *ModuleDispatch*($request, $this（ruleitem）, [$module, $controller, $action], [**'convert'** => **false**]);

**class** Module **extends** Dispatch

返回think\route\dispath\module extends dispath

调用init方法

在这里才调用中间件（查看一下中间件怎么添加的）

总结：通过getMethodRules返回各种rules，然后匹配，匹配成功返回dispath类，然后塞进中间件的队列$this->queue里面，最终实现方法。

下午看看controller方法

分析出类

App/index/controller/Index

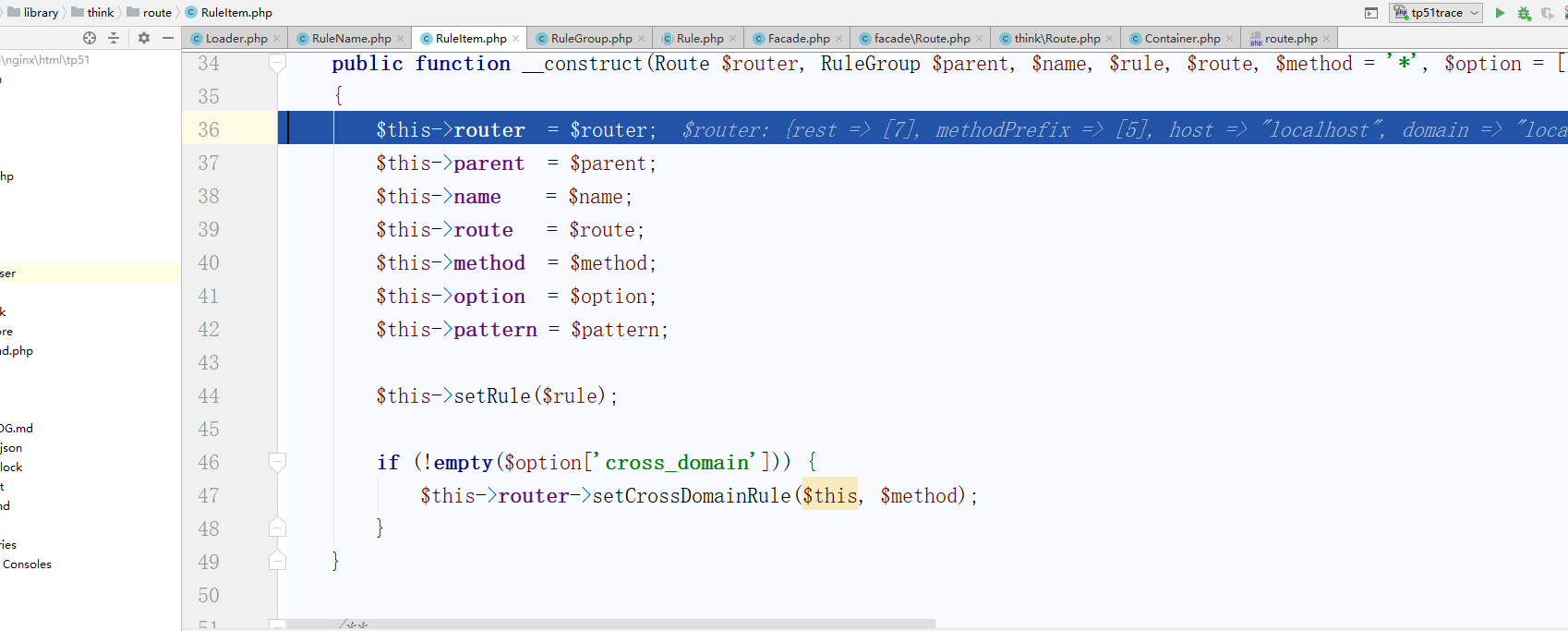
并实例化，如果这个时候有中间件属性则注册

调用middleware->controller();增加一个控制器type中间件

然后再执行middleware->dispatch(request,type=’controller’)

返回数据

完成日志写入，等等功能。



路由反解析：

Think/url:build()

里面调用了

$rule = $this->**app**[**'route'**]->getName($checkName, $checkDomain);

即：think/route：getName

重点来了:

**return** $this->**app**[**'rule\_name'**]->get($name, $domain, $method);

Rulename::get()

返回$this->**item**[$name][] = $value;

然后就是拼装url了