

Pages / ... / swoole

swoft

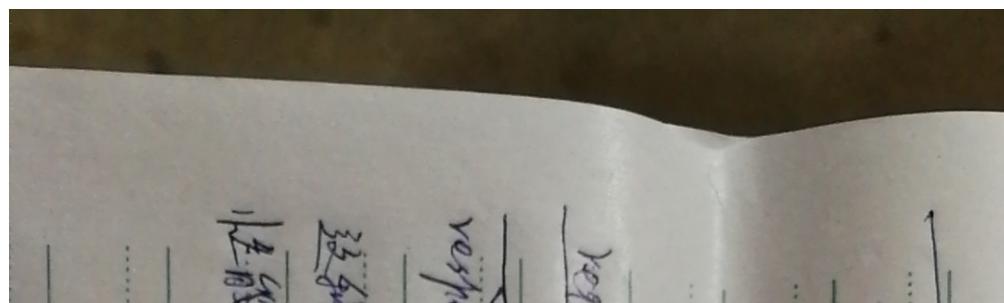
Created by 杨超, last modified on 2020 Feb 27

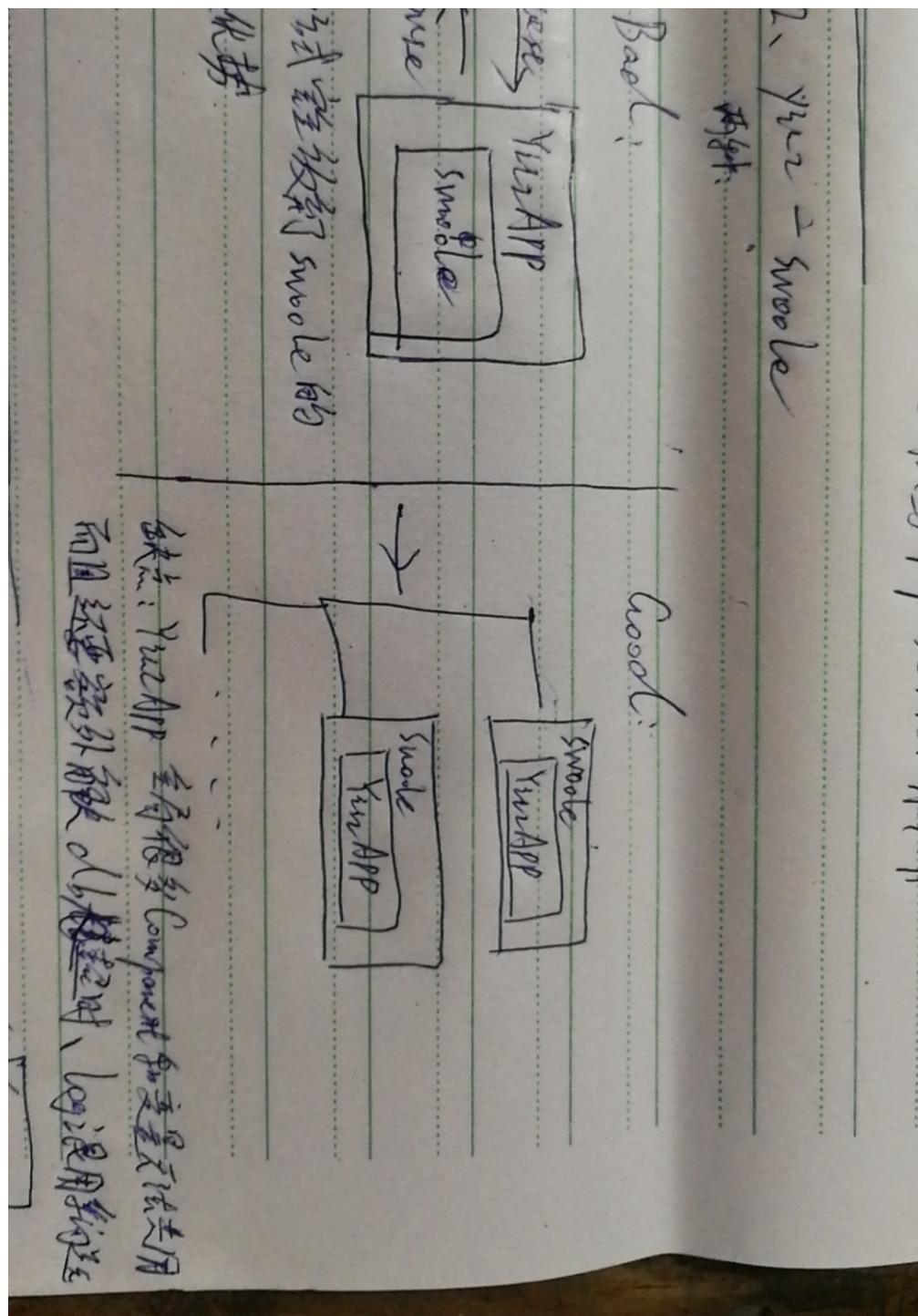
- 前言
- Swoft
- 代码阅读
 - Annotation注解实现
 - 启动过程

前言



yii2-swoole



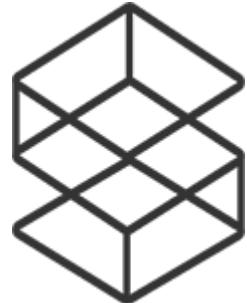


综合之前的两次分享 swoole和swoole与yii2的结合 然后我就开始调研基于swoole的php框架

swoole官方出了一个框架hyperf 但是商业化的

其他的第三方的框架很多 有很多参差不齐的 大体用的多的就是两个easyswoole (近似thinkphp风格)和swoft(承袭spring风格)

Swoft



<https://www.swoft.org/documents/v2/index.html>

swoft: 基于swoole的高性能php框架 除了基本的db/DI容器化支持之外 还原生提供了注解、AOP、数据库连接池等特性 另外代码风格承袭spring 较为严格和规范

==> 下面几块内容比较简单 照着文档肯定能搞出来

安装-略

目录结构-略

注解和ORM使用-略

代码阅读

Annotation注解实现

swoft大量使用了注解 在这里面 注解==代码

实现起来也比较简单 就是利用了反射

1 主要Swoft\Bean\Definition\Parser\AnnotationObjParser

在handle->container启动的时候读取配置

```

parseMethodAnnotations
parsePropertyAnnotations
...

parseClassAnnotations
$parserClassName = $this->parsers[$annotationClass];
$annotationParser = $this->getAnnotationParser($classAry, $parserClassName);

$data = $annotationParser->parse(Parser::TYPE_CLASS, $annotation);

```

2 parse函数: Swoft\Bean\Annotation\Parser\InjectParser

```

public function parse(int $type, $annotationObject): array
{
    // **
    // Parse php document
    $phpReader      = new PhpDocReader();
    $reflectProperty = new ReflectionProperty($this->className, $this->propertyName);
    $docInject      = $phpReader->getPropertyClass($reflectProperty);
    if (empty($docInject)) {
        throw new BeanException(`@Inject` must be define inject value or `@var type` ``);
    }

    return [$docInject, true];
}

```

3 PhpDocReader 是php-di里面的一个类 本质上是通过正则匹配来获取注解内容

```

public function getPropertyClass(ReflectionProperty $property)
{
    // Get the content of the @var annotation
    if (preg_match('/@var\s+([^\s]+) /', $property->getDocComment(), $matches)) {
        list($type) = $matches;
    } else {

```

```

    return null;
}

// Ignore primitive types
if (in_array($type, $this->ignoredTypes)) {
    return null;
}

// Ignore types containing special characters ([] , <> ... )
if (! preg_match('/^[\w\W]+$/i', $type)) {
    return null;
}

$class = $property->getDeclaringClass();

```

这样就可以把注解内容读取出来

启动过程

1 bin/swoft – 类似yii 入口文件

```

<?php
// Bootstrap
require_once __DIR__ . '/bootstrap.php';

Swoole\Coroutine::set([
    'max_coroutine' => 300000,
]);

// Run application
(new \App\Application())->run();

```

两步

bin/bootstrap.php: autoloader&composer 主要就是加载第三方的类
2 SwiftApplication

(new \App\Application())->run();

主要的实现代码“藏”在了init函数里面

```
init:  
// Init system path aliases  
$this->findBasePath();  
$this->setSystemAlias();  
  
$processors = $this->processors();  
  
$this->processor = new ApplicationProcessor($this);  
$this->processor->addFirstProcessor(...$processors);
```

```
$this->setSystemAlias();  
func:  
$basePath      = $this->getBasePath();  
$appPath       = $this->getAppPath();  
$configPath    = $this->getConfigPath();  
$runtimePath   = $this->getRuntimePath();  
$resourcePath  = $this->getResourcePath();  
  
Swoft::setAlias('@base', $basePath);  
Swoft::setAlias('@app', $appPath);  
Swoft::setAlias('@config', $configPath);  
Swoft::setAlias('@runtime', $runtimePath);  
Swoft::setAlias('@resource', $resourcePath);
```

// 启动过程中的日志信息

```
CLog::info('Project path: @base=%s', $basePath);  
CLog::info('Set alias @app=%s', $appPath);  
CLog::info('Set alias @config=%s', $configPath);  
CLog::info('Set alias @runtime=%s', $runtimePath);
```

```
$processors = $this->processors();
/*
return [
    new EnvProcessor($this),
    new ConfigProcessor($this),
    // 加载注解类方法 上面提到过
    new AnnotationProcessor($this),
    // 启动Bean 并且执行AnnotationProcessor刚刚加载的注解类方法进行解析
    new BeanProcessor($this),
    new EventProcessor($this),
    new ConsoleProcessor($this),
];
} */

// 这里获取processor 其中注解的processor也在里面 然后下面遍历和实例化这些processor并且执行handle方法
// BeanProcessor 里面具体执行注解的解析
$this->processor = new ApplicationProcessor($this);
$this->processor->addFirstProcessor(...$processors);
```

Like 崔志华 likes this

No labels

以太资本由艾普拉斯投资顾问(北京)有限公司运营，提供早期互联网项目的投融资对接服务

©2014-2017 以太资本 京ICP备14028208号