Samurai 手册

- 版本: 1.0
- 作者: omsfuk
- 日期: 2017-06-15
- https://github.com/omsfuk/Samurai

前言

- `Samurai` 是一款轻量级MVC框架，注解驱动，配置简单，容易上手。
- 此框架尚处于初级阶段，BUG众多，简陋无比，仅用于交流学习，切勿商用^-^
- 建议想阅读此源码的猿们（假设有的话^-^），先看看使用方法，再去读源码，虽然源码丑的一逼，，，，，
- 顺便说下，代码量不大，去掉测试和空行（貌似还去掉注释），只有不到4000行（十分不爽，断断续续写了几周就写了这么几行代码），适合想深入了解Spring实现的童鞋，作为阅读Spring源码之前的跳板
- 特意撰此手册，为想阅读此框架源码的猿们提供基础用法的指引。
- 另外，此项目用Maven搭建，里面有两个模块，其中web模块用做测试之用，可作为demo参考

Features

- 依赖注入
- 面向切面
- 内置orm
- 事务管理
- restful风格支持
- 自定义视图

快速上手

添加Maven依赖

```
<dependency>
    <groupId>cn.omsfuk.samurai</groupId>
    <artifactId>framework</artifactId>
    <version>1.0-SNAPSHOT</version>
</dependency>
```

添加一个 `samuiai.properties` 文件，内容如下

```
component.scan.path=cn.omsfuk.demo # bean扫描路径
response.view.json=cn.omsfuk.samurai.framework.mvc.view.JsonRespo
nseView # json视图解析器
# 以下用于orm
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/smart?useSSL=true
jdbc.username=root
jdbc.password=root
```

创建一个 `DemoController` 类，加入如下方法

```
@RequestMapping("/index")
@View("json")
public String index() {
    return "hello";
}
```

用浏览器访问 `http://localhost:8080/index`

配置文件

默认的配置文件名为 `samruai.properties` ，如果是Maven项目，则需要放在 `resource` 文件夹下， `ContextListener` 监听器会自动搜寻此文件，并将用到的一些属性自动填充到类中。

```
component.scan.path=cn.omsfuk.demo # 扫描路径
response.view.json=cn.omsfuk.samurai.framework.mvc.view.InternalJ
sonViewResolver # 视图解析器，此解析器为json解析器
jdbc.driver=com.mysql.jdbc.Driver # mysql驱动
jdbc.url=jdbc:mysql://localhost:3306/smart?useSSL=true # mysql 连
接字符串
jdbc.username=root # 用户名
jdbc.password=root # 密码
```

其中解析器可以通过实现 `ViewResolver` 接口来自定义视图解析器，并且在samurai.properties文件中加入一个以 `response.view` 开头的键，例如 `response.view.freemaker` ，值为自定义解析器的全类名。
假定自定义了 `response.view.freemaker` ，此解析器会被作为bean注册，id为 `freemaker` 。
在 `@View` 注解中可以直接用 `@View("freemaker")`

已经内置了 `jsp` 和 `json` 的解析器。其中jsp解析器不需要手动注册，但是json的解析器需要手动注册，就像下面这样

```
response.view.json=cn.omsfuk.samurai.framework.mvc.view.InternalJsonViewResolver
```

IOC

概述

支持三种作用域 `Singleton` ， `Request` ， `Prototype` 。
支持按照类的继承关系来自动查找并注入。

基本用法

- 声明一个Bean
  `@Component`
  `@Controller`
  `@Bean` 此注解加在方法上，静态方法，实例方法均可。bean默认id为返回的类的名称（简称）
  `@Service`
  `@Repository`
  其中每个注解都有value属性，用于自定义beanid
  在属性上加入 `@Autowire` 注解，可以自动注入
  其中构造器的参数会自动从 `BeanContext` 中获取，并自动注入。
  如果不在注解中显式声明Scope属性，则默认为 `Singleton` 作用域

`BeanContextManager` 用于获取当前的 `BeanContext`
`BeanContext` 是Bean容器，提供对Bean的一些基本操作，例如添加bean，获取bean。

AOP

在一个POJO上加上@Aspect注解，就成为了一个切面，此注解有 `order` 属性，可以定义切面优先级，数字越大，优先级越高，默认为1。 `ParameterAspect` （自动注入请求参数切面）的优先级为100000。

- `@Before`
- `@After`
- `@Around`

这三个注解，可以加在加了 `@Aspect` 注解的切面类的方法上，意义十分明显，其中Before和After无参数，而Around规定有两个参数， `Invocation` ， `ProxyChain` ，其中Invocation存储调用信息，ProxyChain用来控制代理链

Aspect类的属性不会自动注入，可以通过 `BeanContextManager` 来获取当前的 `BeanContext` ，进而获得所需要的Bean

Controller

用 `@Controller` 注解来声明一个 `Controller` ， `@RequestMapping` 注解来声明请求映射， `@View` 声明视图解析器，不加此注解则默认为jsp。

可以再控制器请求方法的参数列表中加入Model类型的参数，此参数类似于于Spring中的Model，可以便捷的操作jsp中的model。其中 `HttpServletRequest` 和 `HttpServletResponse` 会在进入到控制器之前被加入到 `BeanContext` 中，所以可以自动注入。

ORM

无需配置文件，只需要在一个接口上声明 `@Repository` ， `OrmContext` 会自动生成实现类，并注册到BeaContext中。用法和Mybatis与Spring整合后的使用方法一样，只需要声明一个此接口类型的属性，Samurai就会自动注入此实现类的实例，生成的Bean默认为 `Singleton` 作用域。

有四大注解
`@Select`
`@Update`
`@Delete`
`@Insert`
用法和Mybatis大体相同，例如声明一个方法，

- 获得某个id的用户

```
@Select("select * from user where id = #{id}")
User getUser(User user);
```

- 获得所有用户的信息

```
@Select("select * from user")
List<User> getUsers();
```

Transaction

Samurai实现了简易的事务管理，只需要在方法上声明 `@Transactional` 。
Samurai参考了Spring，有7大传播级别，请参见Spring的传播级别。但是无隔离级别。

实例

```
@Transactional(propagation = Propagation.MANDATORY)
```