Netty+Spring的web控制器框架

# 背景介绍

目前我们web服务使用Tomcat + Java Servlet + SpringMVC框架，存在以下问题：

* Tomcat配置复杂，学习成本较高，难以掌握和性能优化。
* 目前我们的web业务主要是提供各种REST接口，重点在于controller的编写与提供。Spring MVC显得较重。

Netty作为高性能的网络框架，很简单就可以实现一个优秀的web容器。对于我们的业务需求，可以设想基于Netty的web容器，直接使用SpringMVC的controller机制。但是由于SpringMVC与Servlet耦合，因此难以直接将Netty与SpringMVC做整合。不过我们可以利用Spring框架的IoC特性，单单实现一个controller框架，简单易用。

基于Spring框架松耦合的特性，具体每个controller的业务实现，仍可沿用现有的DAO层不变。

# 框架解析



# 框架使用说明

框架工程名为“HAPPOR”，意思是“HAPPy ORange”的缩写。

以一个简单的服务作为例子：用户请求“/test”，Web服务器以“text/plain”格式返回“hello world”。

首先建立工程环境，编写xml，填入框架所需的bean。XML格式如下：

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans *此处省略xmlns配置*>

<context:annotation-config />

<bean id=*"server"* class=*"cn.lechange.happor.HapporWebserver"*>

<property name=*"port"* value=*"8080"* />

</bean>

<bean id=*"channelInitializer"* class=*"cn.lechange.happor.HapporChannelInitializer"*>

<property name=*"executeThreads"* value=*"200"* />

<property name=*"maxHttpSize"* value=*"20000000"* />

<lookup-method name=*"getControllerContainer"* bean=*"controllerContainer"*/>

</bean>

<bean id=*"controllerContainer"* class=*"cn.lechange.happor.ControllerContainer"* scope=*"prototype"*>

<property name=*"controllers"*>

<list>

<!-- add your controller into netty channel pipeline -->

<ref bean=*"testWebController"* />

</list>

</property>

</bean>

<!-- define your controllers below -->

<bean id=*"testWebController"* class=*"cn.lechange.happor.TestWebController"* scope=*"prototype"*>

<property name=*"method"* value=*"GET"* />

<property name=*"uriPattern"* value=*"^/test"* />

</bean>

</beans>

说明：

* XML文件格式遵循Spring的ApplicationContext格式定义。
* 有三个必须的bean：
  + server：webserver对象，可修改端口配置。
  + channelInitializer：Netty的通道初始化对象，可修改执行线程数和最大HTTP包长限制。
  + controllerContainer：控制器容器对象，由用户将定义的控制器写入列表。注意此对象的scope必须是prototype。
* 定义完三个必须的bean之后，由用户自定义所需开发的控制器的bean。控制器通过HTTP的method和uri进行匹配，其中uriPattern属性填入需要匹配的正则表达式。注意控制器对象的scope也必须是prototype。

定义完所有bean后，保存为web.xml（文件名可任意）。

然后需要由使用者实现控制器，即上面的TestWebController类。所有控制器都需要从抽象类HttpBaseController继承，然后以下抽象方法：

* handleRequest：处理请求，返回FullHttpResponse。
* postHandle：后处理，框架会在返回response后调用。可以在此方法内做一些异步处理。

最后，在main函数中启动webserver即可：

* HapporHelper.runServer("web.xml");