**HADES帮助文档**

**58同城**

出版日期 2012年8月

[1. 阅读向导 3](#_Toc335141912)

[1. HADES是什么 3](#_Toc335141913)

[2. Why HADES 3](#_Toc335141914)

[2. HADES框架概览 5](#_Toc335141915)

[1. 设计原则 5](#_Toc335141916)

[2. HADES架构 5](#_Toc335141917)

[MVCFilter 5](#_Toc335141918)

[MvcDispatcher 5](#_Toc335141919)

[MvcController 5](#_Toc335141920)

[BeatContext 6](#_Toc335141921)

[ActionResult 6](#_Toc335141922)

[3. HADES依赖 6](#_Toc335141923)

[Memcache客户端 6](#_Toc335141924)

[commons-fileupload 7](#_Toc335141925)

[javassist 7](#_Toc335141926)

[org.apache.tomcat 7](#_Toc335141927)

[hibernate-validator 7](#_Toc335141928)

[使用velocity作为HADES页面展示。 8](#_Toc335141929)

[Log4j 8](#_Toc335141930)

[commons-beanutils、commons-lang、commons-collections 8](#_Toc335141931)

[3. HADES基本功能 9](#_Toc335141932)

[初始化 9](#_Toc335141933)

[资源管理 10](#_Toc335141934)

[UI 11](#_Toc335141935)

[日志管理 12](#_Toc335141936)

[路径匹配 12](#_Toc335141937)

[拦截器 13](#_Toc335141938)

[缓存： 15](#_Toc335141939)

[4. HADES扩展功能 18](#_Toc335141940)

[数据绑定对象 18](#_Toc335141941)

[数据校验 19](#_Toc335141942)

[5. 开发规范 19](#_Toc335141943)

[1. 配置 19](#_Toc335141944)

[2. Controller 19](#_Toc335141945)

[3. ActionResult 20](#_Toc335141946)

[6. HADES应用实作 20](#_Toc335141947)

[1. 准备工作： 20](#_Toc335141948)

[安装JDK 20](#_Toc335141949)

[安装和配置maven 20](#_Toc335141950)

[安装集成开发环境 20](#_Toc335141951)

[2. 构建应用： 21](#_Toc335141952)

[创建Maven项目 21](#_Toc335141953)

[填写基本信息 21](#_Toc335141954)

[添加引用 23](#_Toc335141955)

[创建namespace 24](#_Toc335141956)

[创建包 24](#_Toc335141957)

[创建类 25](#_Toc335141958)

[部署应用 27](#_Toc335141959)

[7. 附录 29](#_Toc335141960)

[1. HADES与Spring比较 29](#_Toc335141961)

[2. 可选组件 30](#_Toc335141962)

[memcache客户端 30](#_Toc335141963)

[数据访问层 30](#_Toc335141964)

[支持HA数据库连接池 30](#_Toc335141965)

[3. 参考文档 30](#_Toc335141966)

[4. 术语与缩写技术 30](#_Toc335141967)

# 阅读向导

• 如果你希望马上尝试用HADES来搭建应用，请转至“HADES应用实作”。

• 如果你想了解HADES的整体设计思想，请转至“HADES框架概览”。

## HADES是什么

HADES是一套基于Java的通用Web框架。它在58同城内部被广泛使用。从2012年开始向社会开放源码。

## Why HADES

HADES是一款更适合互联网应用的web 框架。没有复杂的结构和冗余的功能，其架构简单，运行时高效稳定、具备高可扩展性。HADES支持参数绑定、路径绑定、数据校验、超时输出等基本功能。同时HADES也以独特的方式提供了其他框架中的优秀功能。

相比其他框架HADES有如下特点：

1.Interceptor：

HADES并未直接实现AOP功能，取而代之的是使用Interceptor来解决面向切面编程的需求。Interceptor可以被定义为Before，Result和Exception，3种类型，分别用于处理前置拦截，响应拦截，和异常拦截，Interceptor的作用域为在ActionMethod执行之前、之后和发生异常时。

2.零配置：

HADES的MVC核心采用零配置的概念，不需要任何配置文件。通过约束、规范，以及Annotation等获取所需信息。

3.防注入、防XSS：

HADES通过配置的形式对所有接收到的参数进行过滤，可以防止SQL注入和潜在的XSS跨站攻击。

4.超时记录：

HADES在运行时会将响应时长超过100ms的请求URL输出，方便开发者调优。

5.统一部署：

HADES的部署文件和配置文件分别在统一的目录下，方便管理。

6.信息封装

HADES将请求端和响应的信息区分为server与client进行封装，开发者可以很方便的获取如cookie，context等信息。

7.Trace

HADES实现了.Net中的Trace功能，开发者可以将Trace代码嵌入到自己的开发代码中，在运行时HADES会收集Trace信息，并且在页面打印，可以更方便的排查错误和超时等情况，并且Trace功能对于性能的损耗极小。

8.页面缓存

HADES集成了页面缓存机制，可以将已经生成的页面存储到缓存服务器，在调用时进行时间校验，并决定是从缓存中取还是生成新的页面。

9.线程安全

HADES使用本地线程变量对每一个请求进行封装，很好的解决了高并发下多线程之间的安全性问题。

小结：

HADES目前已经在58同城内部绝大部分项目中使用。从业务复杂的电商平台，到性能要求极高的主站系统，再到小型的管理平台，HADES都有良好的表现。

案例一：主站

58同城的主站已经完全使用HADES进行开发，平均每天的PV值已经达到4亿，如此庞大的流量已经足以充分证明HADES的稳定性和处理高并发的能力。

案例二：电商平台

58同城的电商平台是58同城内部业务最复杂的系统之一，在电商平台的开发过程中HADES获得丰富的功能扩展，如分页组件、JSON格式化、异常处理等等。这些公用组件的出现，节省了开发人员大量的时间。

# HADES框架概览

## 设计原则

简单。

高效。

零配置。

## HADES架构

HADES框架由以下几个主要的部分组成。

### MVCFilter

HADES通过Filter实现对所有请求的拦截，并将符合规范的请求交给MvcDispatcher进行处理。

### MvcDispatcher

这个类不仅负责实现请求转发，还负责启动一个MvcWebAppContext容器。MvcWebAppContext是Web应用的核心容器，一些App级别的信息将被逐步增加到这个容器中。MvcDispatcher的主要职责是找到合适的请求Action并调用该方法返回结果。

执行过程：

1.封装请求信息到本地环境变量，从而保证线程安全。

2.匹配合适的路由信息找到RouteResult。

3.执行路由信息中的Action方法并得到ActionResult。

4.进行跳转

5.对本地环境变量进行清除。

### MvcController

这个类是所有Controller的基类，只有继承自这个类的Controller才能够被HADES所扫描，并加载该类中的Action信息，MvcController中封装了日志组件，并且封装了一个应用的上下文变量，在MvcController中添加的可以被其他Controller 使用

### BeatContext

这是一个接口类，其中定义了每个请求被HADES封装后所能提供的数据接口。

### ActionResult

这是一个接口类，用户可以自己定义返回的结果类型。

下图为基于HADES的应用的一般架构：

其中的基础服务和BizController都是业务相关类。



## HADES依赖

### Memcache客户端

Memcache主要是用来支持页面缓存的分布式存储，由于通常页面的输出对象都很大，使用java虚拟机的缓存可能会增加服务器的负载，导致服务器宕机，所以将缓存的对象存储到memcache是不错的选择。

依赖pom：

<dependency>

<groupId>com.bj58.spat</groupId>

<artifactId>com.bj58.spat.memcachedclient</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

### commons-fileupload

用于做附件上传的基本组件，是apache 的开源项目。

依赖pom：

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>1.2.1</version>

</dependency>

### javassist

用于处理java的类对象，可以解析java的class二进制文件得到信息。

<dependency>

<groupId>javassist</groupId>

<artifactId>javassist</artifactId>

<version>3.8.0.GA</version>

</dependency>

### org.apache.tomcat

使用tomcat7的新的声明包，用于实现MvcFilter 的描述和加载。

<dependency>

<groupId>org.apache.tomcat</groupId>

<artifactId>tomcat-servlet-api</artifactId>

<version>7.0.4</version>

<scope>provided</scope>

</dependency>

### hibernate-validator

使用JSR-303的参考实现hibernate的验证作为HADES校验器的标准实现。

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-validator</artifactId>

<version>4.0.2.GA</version>

</dependency>

Velocity

### 使用velocity作为HADES页面展示。

<dependency>

<groupId>org.apache.velocity</groupId>

<artifactId>velocity</artifactId>

<version>1.7</version>

</dependency>

### Log4j

使用log4j作为日志的记录工具。

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.16</version>

<scope>runtime</scope>

</dependency>

### commons-beanutils、commons-lang、commons-collections

作为工具类使用。

<dependency>

<groupId>commons-collections</groupId>

<artifactId>commons-collections</artifactId>

<version>3.2.1</version>

</dependency>

<dependency>

<groupId>commons-beanutils</groupId>

<artifactId>commons-beanutils</artifactId>

<version>1.8.3</version>

</dependency>

<dependency>

<groupId>commons-lang</groupId>

<artifactId>commons-lang</artifactId>

<version>2.5</version>

</dependency>

# HADES基本功能

### 初始化

#### 配置文件初始化

HADES在启动时会初始化一些基本的配置文件。

##### 日志配置

即指定目录下的bj58log.properties，HADES目前使用的是log4j作为日志的记录者,文件中的配置项请参见log4j的配置文档。通过HADES. getLog4jConfigFilePath（）直接获得完整的配置文件路径。

##### db配置

即指定目录下的db.properties。通过HADES. getDBConfigFilePath()可以获得完整的配置路径。

##### SqlHtmlConverter配置

与日志配置和db配置不同的是SqlHtmlConverter是根据配置文件的存在而决定是否启用的，HTML的转化器的配置文件为html-encode.properties，其中定义了防止XSS攻击所要处理的数据，SQL转化器的配置文件为sql-inject.properties，其中定义了防止SQL注入所要处理的必要的字符。

#### Action信息初始化

在启动时HADES会扫描所有符合条件的Controller，并加载各类中的Action方法，将path信息和方法信息放到路由管理中。

#### WebAppContext初始化

在HADES启动时会创建MvcWebAppContext 的单例，该实例将会在整个应用中唯一存在。

### 资源管理

#### 操作系统相关性

与操作系统相关性主要体现在根目录上。在window下HADES将会取运行时Web容器所在的根目录，如Tomcat在D:则HADES会将D:\opt\hades视为所有HADES应用的公用目录。如果是在linux操作系统，根目录为/opt/hades

#### 名字空间

名字空间在HADES是个非常重要的概念，每个web项目都有他自己的namespace

namespace在项目类的根目录下的META-INF/namespace.properties文件中设置

namespace=<yournamespace>

#### 配置文件

HADES本身不会有任何的配置文件，当引入的工具包中需要配置文件时, 将所需配置文件全部放在/opt/ hades /目录下统一管理 ：

对应于windows开发环境，在web容器启动所在盘符的 /opt/ hades /目录下

对应linux环境，为 /opt/ hades /目录，需要赋予读写权限

读取配置文件路径的办法

// HADES.getNamespace()，对应设置的命名空间值

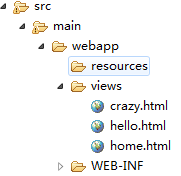
File currentConfigFolder = new File(HADES.getConfigFolder() + HADES.getNamespace());

#### 系统资源

应用的资源文件必须要放在webapp的resources目录下，如页面样式css，脚本js等文件。当系统需要读取文件资源时，HADES会在该目录下进行查找。

HTML页面：

所有的页面放在webapp的views目录下



view的名字可以在ActionResult中指定如：return ActionResult.view("hello"); 这样HADES会将ActionResult绑定hello.html模板。

view是通过velocity解析

view的文件结尾是.html

在页面中可以通过默认的系统变量“\_\_beat”，可以在view中直接使用BeatContext如：

$!{\_\_beat.server.contextPath}获得当前的应用路径

页面实例代码如下：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta content="zh-cn" http-equiv="Content-Language" />

<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />

<title>hello,这是第一个网页</title>

</head>

<body>

<a href="$!{\_\_beat.server.contextPath}/hello/58">hello,58</a>

<form action="$!{\_\_beat.server.contextPath}/hello/login" method = "get">

Name:<input type = "text" name = "name" value = ""/>

Pass:<input type = "text" name = "pass" value = ""/>

<input type = "submit" value = "submit"/>

</form>

</body>

</html>

JS和css文件



js文件可以放在resources目录下。

### UI

HADES采用Velocity作为显示的模板。Velocity是基于java的模板引擎（template engine）。它允许简单的使用模板语言（template language）来引用由java代码定义的对象，当Velocity应用于web开发时，界面设计人员可以和java程序开发人员同步开发一个遵循MVC架构的web站点，也就是说，页面设计人员可以只关注页面的显示效果，而由java程序开发人员关注业务逻辑编码。Velocity将java代码从web页面中分离出来，这样为web站点的长期维护提供了便利，更多的Velocity使用规范，用户可以参考Velocity的开发文档。

向UI传数据：

在Controller可以通过调用beat.getModel().add(attributeValue) 将由数据添加到beat中去并且传给view层，通过调用beat.getModel().remove(attributeValue)移除添加的Model信息。通过merge整合Model信息，view层可以通过beat.getModel().get(attributeValue)来获得具体的Model

### 日志管理

HADES日志的默认输出目录为/opt/hades/logs,日志会根据/opt/ hades /{namespace}/下的配置项规则，生成相应的日志文件。日志文件的名称与{namespace}的名字相同，在同一服务器上部署多个web 应用可能会产生多应用的日志文件，所有的日志文件都会在这个目录，方便维护。

### 路径匹配

路径的匹配规则是页面与Controller交互的桥梁，用哪个Controller 的哪个方法来处理页面的请求，就是根据Action的路径匹配规则进行处理。

Ant path 匹配原则

优先精确匹配

多个Path匹配时，优先使用最长符合路径模式来匹配一个路径。注意的是，这个原则并没有在任何高级别的接口中指定，但却是一种实现细节。

通配符匹配方法

Table Ant Wildcard Characters

|  |  |
| --- | --- |
| Wildcard | Description |
| ? | 匹配任何单字符 |
| \* | 匹配0或者任意数量的字符 |
| \*\* | 匹配0或者更多的目录 |

Table Example Ant-Style Path Patterns

|  |  |
| --- | --- |
| Path | Description |
| /app/\*.x | 匹配(Matches)所有在app路径下的.x文件 |
| /app/p?ttern | 匹配(Matches) /app/pattern 和 /app/pXttern,但是不包括/app/pttern |
| /\*\*/example | 匹配(Matches) /app/example, /app/foo/example, 和 /example |
| /app/\*\*/dir/file. | 匹配(Matches) /app/dir/file.jsp, /app/foo/dir/file.html,/app/foo/bar/dir/file.pdf, 和 /app/dir/file.java |
| /\*\*/\*.jsp | 匹配(Matches)任何的.jsp 文件 |
|  |  |

### 拦截器

HADES的拦截器是基于注解实现的，使用方式简单,在Controller的类或者方法上添加@InterceptorName声明。

当使用@InterceptorName标记一个Action方法时则Interceptor只对当前Action有效，当标记一个完成的控制器类，这个Action拦截器将会应用到该控制器的所有Action方法上。当有请求到达处理的具体Action时，系统会先对方法和方法所在类的拦截器描述进行扫描如果存在，则先将请求转发给拦截器进行处理，如果拦截器返回ActionResult则返回响应，如果拦截器处理但是未返回ActionResult则HADES会将请求继续交给能够处理该请求的方法，继续进行处理。

每一个Action拦截器都可以有一个 order 属性，用来决定Action拦截器在该范围内的执行顺序，order属性必需是0或者更大的整数值，如果没有order属性，默认为1，同级order执行顺序不定。

使用拦截器解决了部分代码重复的问题。很多日志处理代码和异常处理代码是很相似的，这样就导致了各个Action中存在大量重复代码，从而使 Controller的工作更加明确，只负责表示逻辑，而不被一大堆日志处理代码和try...catch块包围使我们的Action，应该是干净的、工整的、仅包含表示逻辑的Action，并且代码更加优雅。

拦截器的定义代码如下：

@Interceptor(CrazyInterceptor.class)

@Target({ElementType.METHOD, ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

public @interface Crazy {

/\*\*

\* 可以定义拦截器的的优先级，0，1，2,...

\* @return

\*/

int order() default 1;

}

拦截器的实现类：

public class CrazyInterceptor implements ActionInterceptor {

int count = 0;

/\*\*

\* 由于不是线程安全，所以需要同步处理

\* @return

\*/

private synchronized int getCount(){

count ++;

if (count > 4) count = 0;

return count;

}

/\*\*

\* 判断中奖没？:)

\*/

public ActionResult preExecute(BeatContext beat) {

int count = getCount();

return count == 0 ? ActionResult.view("crazy") : null;

}

类拦截器应用示例：

@InterceptorName

public class HelloController extends MvcController{

方法拦截器应用示例：

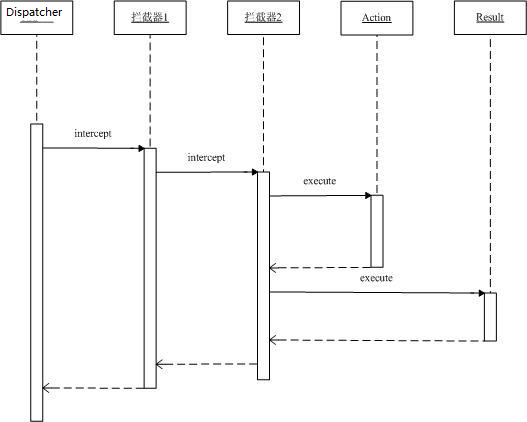
@InterceptorName

public ActionResult home() throws InterruptedException {

使用拦截器通常可以实现以下功能：

|  |  |
| --- | --- |
| 页面授权根据登录用户的权限 | 阻止或许可用户访问特定的页面 |
| 日志和审计记录 | 检查用户访问WEB应用的情况 |
| 页面压缩 | 压缩页面内容，加快下载速度 |
| 页面缓存 | 通过时间戳和分布式缓存实现对页面的缓存处理 |
| 。。。 | 。。。 |

原理图：



### 缓存：

#### 页面cache

用法java代码

@Path("")

@GET

@OutputCache(duration= 120) // 页面缓存，设置拦截器，duration为缓存时间单位秒，默认60s

public ActionResult home() {

System.out.println("aaaaaaaaaaaaaaaaaaaaa");

beat.getClient().getCookies().set("name", "123456789");

String view = "home";

return ActionResult.view(view);

}

memcache配置文件 /opt/ hades /<命名空间>/pagecache\_memcache.xml

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<memcached keyPrefix="URL-">

<servers>

<add address="192.168.100.19" port="12001" />

</servers>

<socketPool initConn="10" minConn="10" maxConn="30" aliveCheck="true" readTimeOut="1000" failover="true" maintSleep="30" nagle="false" />

</memcached>

</configuration>

#### 处理原理

A客户端第一次访问，HADES处理后，缓存内容到cache,过期时间为 duration指定值，http状态码200

HTTP/1.1 200 OK

Content-Type: text/html;charset=UTF-8

Cache-Control: max-age=60

. A客户端在60s内回车请求，直接读取本地缓存，不请求服务器，没有网络流量

. A客户端在60s后刷新，如果缓存没失效，返回 304

request

GET http://localhost:8011/ HTTP/1.1

Host: localhost:8011

Connection: keep-alive

Cache-Control: max-age=0

User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.696.68 Safari/534.24

Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,\*/\*;q=0.5

Accept-Encoding: gzip,deflate,sdch

Accept-Language: zh-CN,zh;q=0.8

Accept-Charset: UTF-8,\*;q=0.5

Cookie: renjun=123456789

If-Modified-Since: Tue, 17 May 2011 01:46:45 GMT

response

HTTP/1.1 304 Not Modified

B客户请求web server，服务端逻辑，如果缓存中有直接返回缓存内容

HTTP/1.1 200 OK

Content-Type: text/html;charset=UTF-8

Cache-Control: max-age=60

cookie 不被缓存

cache key的默认值为url的全路径，包括schema, servername, port, uri,querystring, 如下：

http://bj.58.com/erlizhuang/zufang/?p=b3 key生成的代码如下：

private String getFullUrl(BeatContext beat){

    HttpServletRequest request = beat.getRequest();

    StringBuffer url = new StringBuffer();

    String scheme = request.getScheme();

    int port = request.getServerPort();

    if (port < 0)

        port = 80; // Work around java.net.URL bug

    url.append(scheme);

    url.append("://");

    url.append(request.getServerName());

    if ((scheme.equals("http") && (port != 80))

      || (scheme.equals("https") && (port != 443))) {

      url.append(':');

      url.append(port);

    }

    url.append(request.getRequestURI());

    String queryString = request.getQueryString();

    if(queryString != null)

        url.append('?').append(queryString);

    return url.toString();

 }

# HADES扩展功能

### 数据绑定对象

Action方法中可以使用对象作为参数，MVC框架可以将Request中的数据绑定，示例代码如下

@SqlSafe

@Path("new")

@POST

@Async

public ActionResult create(Account account) {

// 绑定数据错误

if (beat.getBindResults().hasError()){

beat.getModel().add (account);

return ActionResult.View("account/createForm");

}

this.accountService.put(account.assignId(), account);

return ActionResult.Redirect("./" + account.getId());

}

由代码可以看到系统的Action方法create中是一个有参方法Account 其中Account 为系统中的实体对象，包含一些get和set方法，当该方法对请求进行处理时，HADES会先对这些请求数据进行处理，把参数名与Account对象中的属性名相同的参数自动绑定给Account对象，开发人员可以直接通过对account 实体操作获得参数值。

#### URL路径绑定方法参数

HADES支持路径绑定方法参数，即路径中的每个值都可以作为参数值在Controller中直接获得，也可以使得前端的路径更加简单明了。

@Path中可以使用正则表达式，其中{paramname}或{paramname:expression},paramname对应方法中的参数名

@Path("{id}")

@GET

@SqlSafe(false)

public ActionResult getView(Long id) {

Account account = this.accountService.get(id);

### 数据校验

HADES支持JSR-303 标准所自带的一些约束声明，并对数据进行校验处理。

@NotNull

@Size(min=1, max=25)

private String name;

@NotNull

@Size(min=6, max=25)

private String password;

@Min(10)

@Max(100)

private int age;

# 开发规范

HADES秉承约束大于约定的原则，在开发过程中尽量不要打破HADES所提出的各项约束和规范。

## ****配置****

不可以在webapps下部署任何配置文件，上线前请自行检查

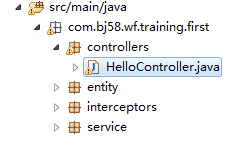
在web.xml中不可以配置其他参数

## **Controller**

开发人员在编写Controller时需要遵守以下规范，只有遵守开发规范的Controller才会被加载到分发器中。

所有的Controller在com.bj58.\*.controllers下

如：



所有的Controller必须是Controller结尾 。

所有的Controller必须继承于MVCController 。

MVCController是所有自定义Controller的基类，这样做的好处在于以后扩展新的通用功能时所有Controller在不改变代码的情况下就可以享受到在基类中新增的功能，目前在MVCController中我们绑定了HADES内置的日志实例，开发者如果想在Controller中记录日志，直接使用log.XXXX()就可以进行日志记录，并将日志输出到/opt/ hades /${namespace}/logs/下。

所有的Action必须是public的，只有public的Action才能够被加载并处理请求。

所有的Action必须返回[ActionResult](http://10.58.120.19/ActionResult)

遵守以上开发规则后，可以实现可用的Controller，Controller是Web项目最重要的部分，决定调用后端哪些服务处理请求，并对结果进行处理，然后决定跳转的页面。

## **ActionResult**

所有的方法都必须返回ActionResult 如:View Velocity,Redirect,Stream,HttpStatus

开发者如果想自定义ActionResult可以实现ActionResult接口，并进行实现。

# HADES应用实作

## 准备工作：

### 安装JDK

Webx需要JDK 6.0以上的版本。请从这里下载并安装它：

http://www.oracle.com/technetwork/java/javase/downloads/index.html

### 安装和配置maven

Webx需要maven 2或更高版本。请从这里下载并安装它：http://maven.apache.org/。

请在maven配置中添加Webx所在的repository，以便取得Webx发布版、archetypes、以及相关插件。

### 安装集成开发环境

很难想像不用集成开发环境（IDE）来帮助开发Java应用会变成怎样。

如果你使用Eclipse（从这里下载：http://www.eclipse.org/），建议安装如下插件：

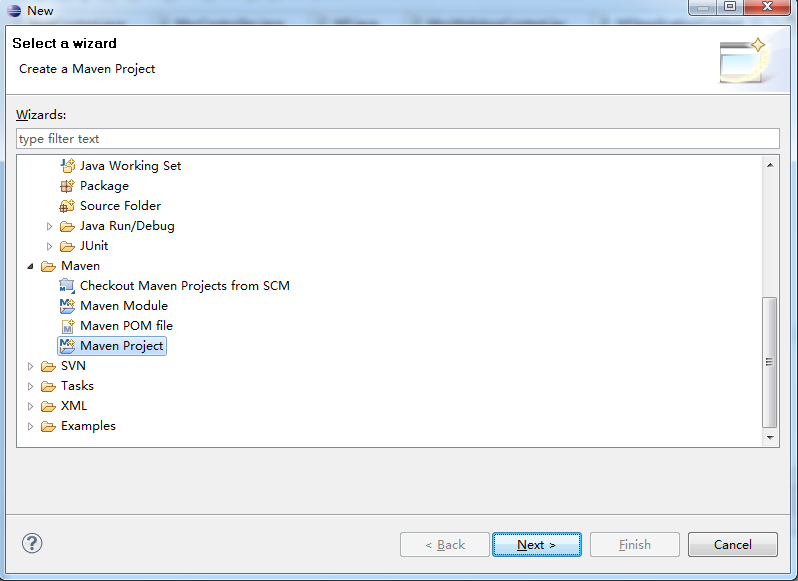
• Maven eclipse插件：http://m2eclipse.sonatype.org/

• Subversion eclipse插件：http://subclipse.tigris.org/

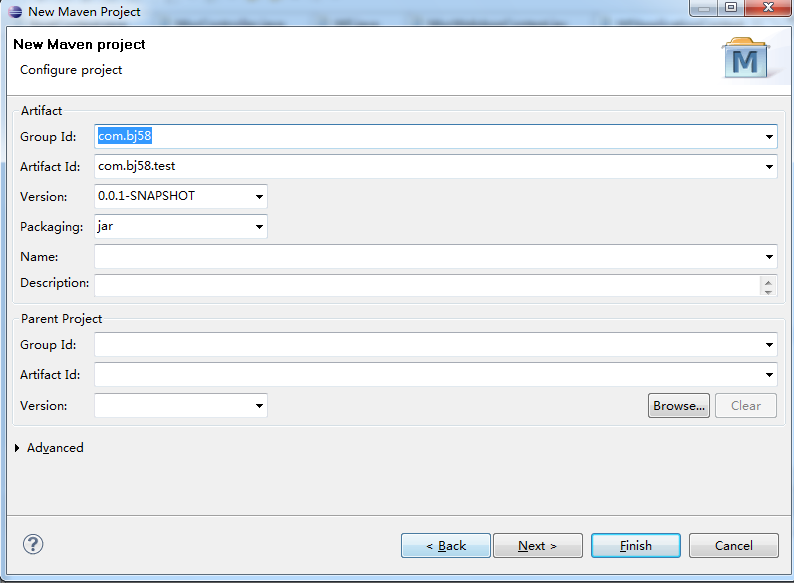
## 构建应用：

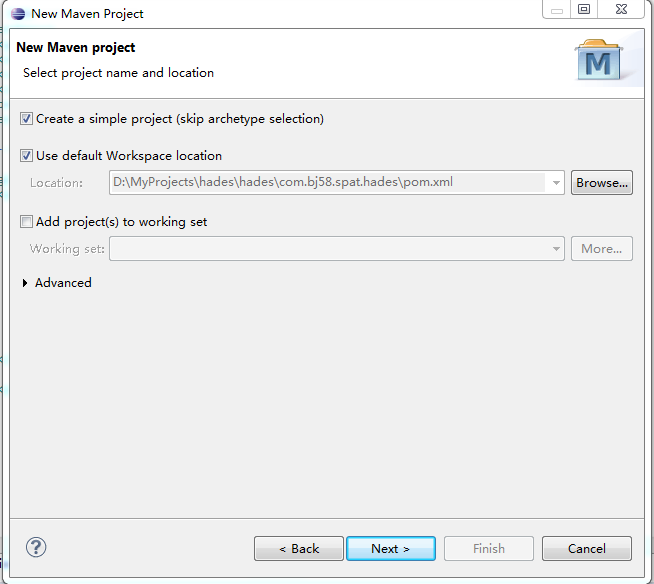
### 创建Maven项目

利用eclipse的Maven插件创建Maven项目

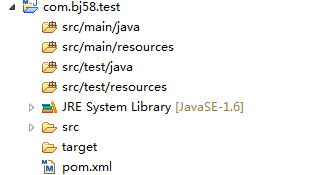


### 填写基本信息





这是标准的maven项目结构



### 添加引用

在pom.xml中添加对HADES的引用。

<dependencies>

<dependency>

<groupId>com.bj58.spat</groupId>

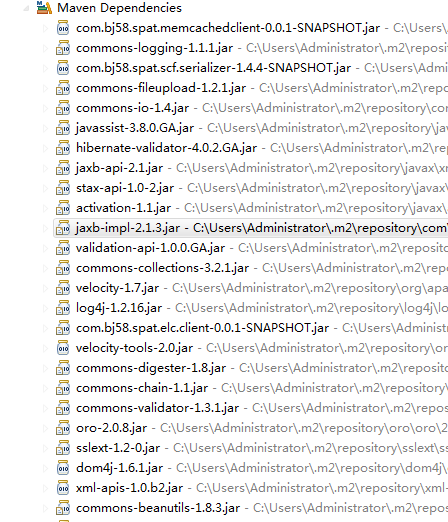
<artifactId>com.bj58.spat.hades</artifactId>

<version>1.0.0</version>

</dependency>

</dependencies>

可以看到项目的引用增加了以下的Maven依赖，表明依赖生效。



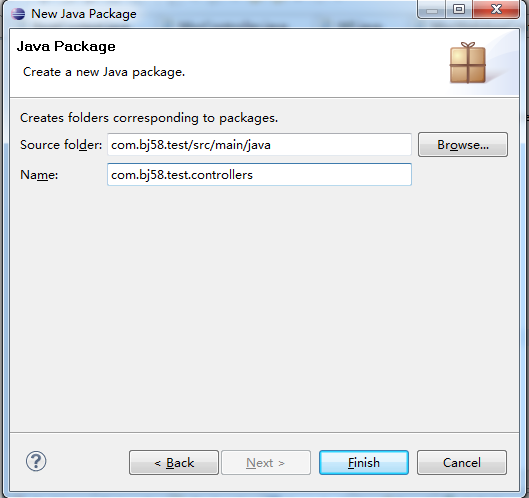
### 创建namespace



首先要创建 META-INF文件夹 并在其中建立namespace.properties文件，在namespace.properties中设置namespace=first，标识自己的项目名称和配置文件目录等信息。

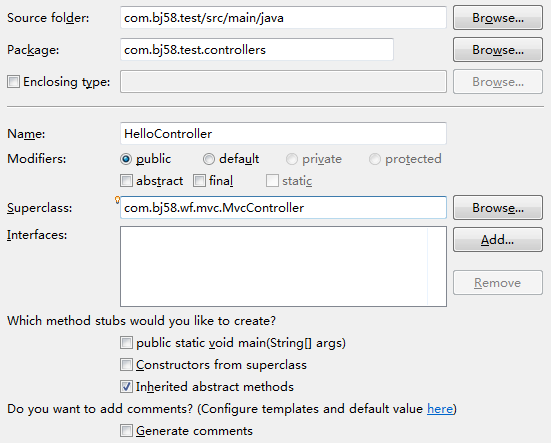
### 创建包

创建包名为com.bj58.test.controllers的包用来存放controller。



### 创建类

创建HelloController类，该类继承自MvcController



并在该类添加以下内容

**package** com.bj58.test.controllers;

**import** com.bj58.spat.hades.ActionResult;

**import** com.bj58.spat.hades.MvcController;

**import** com.bj58.spat.hades.annotation.Path;

**public** **class** HelloController **extends** MvcController{

@Path("/index")

**public** ActionResult hello(){

beat.getModel().add("username","58");

**return** ActionResult.*view*("hello");

}

}

在项目的src目录下创建如下目录结构，用来放置资源和模板文件。



views中用来放置velocity模板文件，resource用来放置项目的其他资源文件。

在views目录下创建hello.html内容：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta content="zh-cn" http-equiv="Content-Language" />

<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />

<title>hello</title>

</head>

<body>

<p>hello, $!username

</p>

</body>

</html>

考虑兼容性需要在WEB-INF下建立 web.xml 内容为

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

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:web=*"http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*

id=*"WebApp\_ID"* version=*"2.5"*>

</web-app>

### 部署应用

jetty是eclipse内置的服务容器，在开发过程中建议使用jetty作为测试容器。

在pom.xml文件中添加以下内容：

<build>

<finalName>com.bj58.test</finalName>

<plugins>

<plugin>

<!-- http://wiki.eclipse.org/Jetty/Feature/Jetty\_Maven\_Plugin -->

<groupId>org.mortbay.jetty</groupId>

<artifactId>jetty-maven-plugin</artifactId>

<configuration>

<stopPort>9966</stopPort>

<stopKey>foo</stopKey>

<scanIntervalSeconds>0</scanIntervalSeconds>

<connectors>

<connector implementation=*"org.eclipse.jetty.server.nio.SelectChannelConnector"*>

<port>80</port>

<maxIdleTime>60000</maxIdleTime>

</connector>

</connectors>

<webAppConfig>

<contextPath>/hello</contextPath>

</webAppConfig>

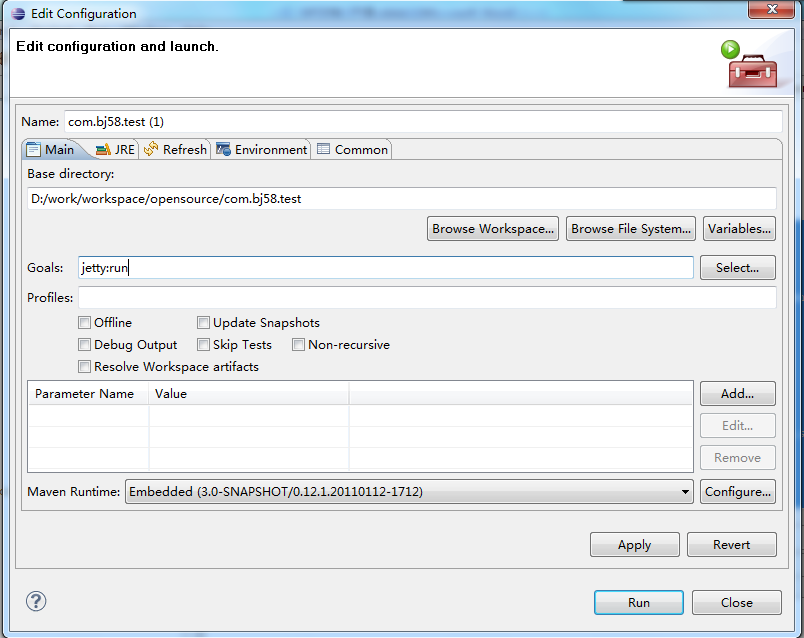
</configuration>

</plugin>

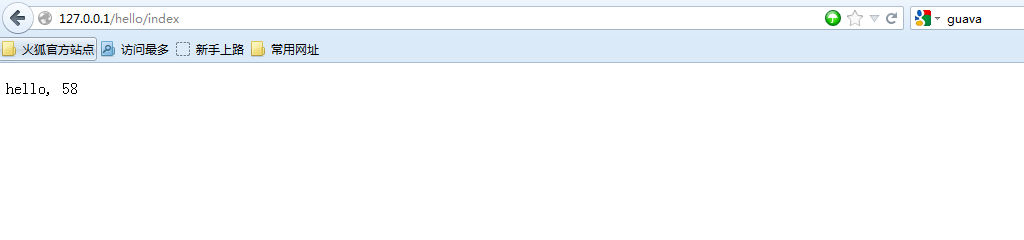
</plugins>

</build>

右键单击项目🡪Debug As🡪Maven Build弹出界面，在Goals中输入:jetty:run



在浏览器中输入:127.0.0.1/hello/index，如果观察到如下结果，则证明服务正常。



如果是在生产环境发布，可以使用maven 的package 打包成war包，然后部署在Tomcat下。

# 附录

## HADES与Spring比较

|  |  |  |
| --- | --- | --- |
|  | **Spring MVC** | **HADES.MVC** |
| 配置文件 | Web.xml, applicationcontext.xml | 无（可选） |
| Controller | @Controller | 约定 |
| url 匹配 | @RequestMapping | @Path |
| 运行监控 | 无 | 运行时间，健康状况 |
| 复杂度 | 复杂 | 简单 |
| 可扩展性 |  | 易 |
| 拦截器 | 复杂 | 简单，强大 |
| 出错处理 | 无 | 简单，强大 |

## 可选组件

### memcache客户端

58同城内部使用的memcache客户端通过异步访问、和二进制存储大大提高了系统性能。

### 数据访问层

58同城通过dal组件对数据库进行访问。

DAL组件属于简单的ORM组件，能够实现对数据库的面向对象封装。

### 支持HA数据库连接池

58同城通过支持HA的数据库连接池，实现数据库宕机时的业务处理。

等等。。。 我们会陆续开源内部使用的优秀组件。

## 参考文档

《Spring开发文档》

## 术语与缩写技术

MVC模式：MVC架构是"Model-View-Controller"的缩写，中文翻译为"模型-视图-控制器"。MVC[应用程序](http://baike.baidu.com/view/330120.htm)总是由这三个部分组成。Event(事件)导致Controller改变Model或View，或者同时改变两者。只要Controller改变了Model的数据或者属性，所有依赖的View都会自动更新。类似的，只要Controller改变了View，View会从潜在的Model中获取数据来刷新自己。

Interceptor：拦截器，是HADES里一项基本概念，其主要作用是对符合某一条件的http请求进行过滤处理，interceptor可以分为前置和后置两种，分别作用于请求到达处理前和响应离开前。

Controller：控制器，是HADES中起应用业务处理的主控制器，其接收用户的输入并调用模型和视图去完成用户的需求。所以当单击Web页面中的超链接并发送数据时，Controller本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后再确定用哪个视图来显示返回的数据。

BeatContext：即应用上下文，是每一个请求在HADES中的所有信息的集合，其中包含了原始的请求信息和响应流，BeatContext中的 Model 由Controller传递给View可以用作数据展现，BeatContext中的BindResults 存储绑定和校验结果，BeatContext中的Client存储客户端信息，如Cookie、url路径等，BeatContext中的Server 存储服务器端信息Session，环境变量等。

Model：即数据模型，在HADES中的每一个数据实体和具体处理业务数据的操作都可以视为Model。模型表示企业数据和业务规则，模型拥有最多的处理任务，被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据。由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。