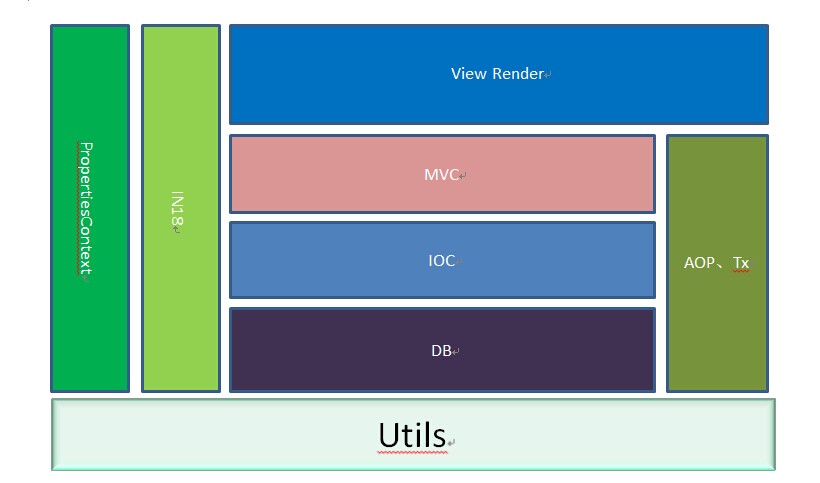
经过本人差不多半年的纠结，今天终于把JBeer的0.1版本完成了。Jbeer具备MVC，IOC,AOP,ORM,IN18,PropertiesContext，简单声明式事务以及自带Datasource功能。开发Jbeer的目的是为了总结J2EE框架原理，从而在一些细节上加入一些自己的想法，最终目的是个人的一个总结。当初第一次看到JFinal源码的时候，知道了MVC框架内部的原理原来是那样的，第一次参与了Smart4j（原来smart），了解了轻量级的框架是那样子的。于是便萌发了自己也做一个这样的东西来总结一下。JBeer并不是站在代替JFinal，smart4j或者ssh框架去实现的，而是一次总结的过程。之前只是站在框架的外围看它，当你站到框架内部去看它的时候，你会发现你才真正的理解了它。废话不多说，先出一个JBEER的“一寸免冠照片”来给大家瞅瞅。



看了JBeer的仪容之后，那么下面大概介绍一下各个功能吧。

在介绍各个功能之前，还是先废话一下。JBEER是追求极少的配置文件，这是借鉴了JFinal的风格。需要使用JBEER，只需要将项目依赖jbeer的jar包，然后在web.xml中配置如下信息即可

<context-param>

<param-name>basePackageName</param-name>

<param-value>org.jbeer.sample</param-value>

</context-param>

<listener>

<listener-class>com.jbeer.framework.startup.JBeerWebContextListener</listener-class>

</listener>

<servlet>

<servlet-name>jbeerDispatcherServlet</servlet-name>

<servlet-class>com.jbeer.framework.web.JBeerDispatcherServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>jbeerDispatcherServlet</servlet-name>

<url-pattern>\*.htm</url-pattern>

</servlet-mapping>

然后实现一个Configurate接口，需要注意的是，实现Configurate接口的类必须在上面web.xml里面配置的basePackageName的包内或者子包内。下面给出了一个实例：

**public** **class** AppConfig **implements** Configurate {

**public** **void** configurateContext(JBeerConfig config) {

config.setApplicationEncode("UTF-8");

}

**public** **void** configurateAop(AopConfig config) {

}

**public** **void** configurateDB(DBConfig config) {

}

**public** **void** configurateWeb(WebConfig config) {

config.setViewPrefix("/WEB-INF/pages");

config.setViewSuffix(".jsp");

}

**public** **void** configureateIOC(IOCConfig config) {

}

**public** **void** configurateIN18(IN18Config config) {

config.setBaseName("in18\_message");

}

**public** **void** configurateProperties(PropertiesConfig config) {

config.setPropertiesPath("\*\_test.properties");

config.setPropertiesPath("conf/\*.properties");

}

}

这样，你的项目就可以使用jbeer来进行开发了。

1. MVC模块

该模块借鉴了Spring mvc的风格，采用注解的方式来配置请求与处理action的关系。实例如下：

@Controller(urlPattern="/first")

**public** **class** FirstController {

**private** User user;

@Action(urlPatterns="index.htm",requestType=RequestType.*ANY*)

**public** ModelAndView index() **throws** IOException, ScanClassException{

PageModelAndView mav = ModelAndView.createModelAndView();

mav.setDataMap("user", userService.getName());

mav.setView("view");

return mav;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

}

上面就完成了一个简单的controller和action，当访问/first/index.htm的时候，则会触发index方法执行，然后返回视图和数据模型。上面有一个User字段，字段名是user，并且有setUser方法，当请求参数中包含user.name字段传递过来，则会自动封装成User对象的name字段，并且调用setUser方法复制给当前的controller。在Action注解的urlPatterns支持restful风格的请求路径，比如index\_${id}\_${name}.html,并且在对应的action的方法入参(@PathParam(“id”)int id, @PathParam(“name”)String name),这样当请求的时候将会自动将路径对应的占位符内容当作入参传入。具体细节，后面会在每个单独模块总介绍。

1. IOC模块

IOC通过Bean注解申明是一个IOC的bean实体，通过RefBean表明应用IOC容器中的一个实体。不多说，直接粘贴处实例代码。

@Bean

**public** **class** UserService {

@RefBean

**private** CodeService codeService;

**public** String getName(){

**return** codeService.hello()+" world";

}

}

RefBean注解是可以定义在方法的入参级别，当Controller的action方法入参注解了RefBean，那么这个参数将会从IOC容器中获取一个对应的实体，传入方法内。IOC提供了第三方接口，用于监控实体bean的初始化过程等功能。关于详细的设计思想将会在对应的模块中介绍。

1. AOP模块

也是通过注解的方式申明一个增强。

@AOP(classRegex="org.jbeer.sample.bean.service.\*Impl")

**public** **class** TestAspect {

@Before

**public** **void** before(InvokeHandler handler){

System.*out*.println(**this**.getClass().getName()+handler.getClass().getName()+"."+handler.getInvokeMethod().getName()+"before");

}

@After

**public** **void** after(InvokeHandler handler,Object ret){

System.*out*.println(**this**.getClass().getName()+handler.getClass().getName()+"."+handler.getInvokeMethod().getName()+",return:"+ret+"after");

}

@Excep

**public** **void** exception(InvokeHandler handler,Exception e){

System.*out*.println(**this**.getClass().getName()+handler.getClass().getName()+"."+handler.getInvokeMethod().getName()+",exception:"+e.getMessage());

}}

通过AOP注解表示这是一个增强，并且制定切面，切面可以制定到类级别和方法级别，如果两个属性都没配置，则增强所有IOC容器中的bean实体，切面的定义是大家熟悉的正则表达式。然后依次在方法中注解Before,After,Excep是什么意思，我想我不说应该都知道是什么意思。

1. ORM模块

JBEER采用链式的SQL拼装加上可自定义SQL的方式进行SQL执行，然后ORM转化，提供了导航式的数据库访问。不多说，还是来点代码有意思：

执行插入：

DB.*insert*(user).execute();

DB.*insert*(User.**class**).insert("name", "bieber").execute();

执行更新：

DB.update(user).where("id", OperationType.EQUAL, 23).execute();

执行删除：

DB.delete(User.class).where("id", OperationType.EQUAL, 23);

执行查询：

DB.select(User.class).page(1, 1).selectList();

上面只是展示了简单几个实例，在执行这几个语句之前，必须到对应的实体类里面进行通过Entity和Column进行注解。

1. 关于配置信息加载以及IN18部分

通过实现Configurate接口，告知Jbeer需要加载的配置资源地址以及IN18资源地址，那么JBEER将会将其加载到容器中。通过如下方式注入到Bean中：

@Properties(name="test")

private String test;

@Message(name="test",args={"bieber","tom"})

private String message;

通过Properties和Message注解边可引用制定的配置内容，这两个配置也支持方法级别，在Controller的Action方法入参上面可以配置相关引用信息。

1. 声明式事务

通过在bean的方法上面配置Tx注解来表示这个方法需要进行事务管理，默认的传播机制是PROPAGATION\_REQUIRED

1. 关于JBeer测试

当使用Jbeer的时候，需要对你的功能进行单元测试，而且测试过程中需要引用Jbeer中的IOC中实体bean，或者是配置信息或者是in18内容，或者数据库访问。很简单就可以将jbeer嵌入到测试中。只需要测试类实现JBeerTestHelper抽象类，实现相关方法，便可以通过注解引用jbeer中的资源。

以上是JBeer各个功能的一个预览，基本上支持了J2EE常规开发的一些需求，可能有些细节上还有不足，希望大家积极反馈意见和问题，从而使得JBeer更加完整。