

Mtools1.0v 使用说明

2014/7/21
张广海

修改记录:

版本	内容	修改人	修改日期
1.0v	新增使用手册	张广海	2014/7/22

目录

1.Mtools 介绍	5
1.1 插件结构设计	5
2 . 插件中各包的用处	5
2.1 com.mtools.core.plugin	5
2.2 com.mtools.core.plugin.annotation	6
2.3 com.mtools.core.plugin.aop.aspect	6
2.4 com.mtools.core.plugin.auth	6
2.5 com.mtools.core.plugin.constant.....	6
2.6 com.mtools.core.plugin.db	6
2.7 com.mtools.core.plugin.ehcache	6
2.8 com.mtools.core.plugin.entity	6
2.9 com.mtools.core.plugin.entiy.vo.....	6
2.10 com.mtools.core.plugin.freemark	6
2.11 com.mtools.core.plugin.helper	6
2.12 com.mtools.core.plugin.lucene.....	7
2.13 com.mtools.core.plugin.mail.....	7
2.14 com.mtools.core.plugin.monitor.....	7
2.15 com.mtools.core.plugin.notify	7
2.16 com.mtools.core.plugin.optlog	7
2.17 com.mtools.core.plugin.properties	7
2.18 com.mtools.core.plugin.security	7
2.19 com.mtools.core.plugin.solr	7
2.20 com.mtools.core.plugin.spring.resolver	7
2.21 com.mtools.core.plugin.staticres	8
2.22 com.mtools.core.plugin.sys	8
2.23 com.mtools.core.plugin.task.....	8
2.24 com.mtools.core.plugin.web.filter	8
2.25 com.mtools.core.plugin.web.interceptor	8
2.26 com.mtools.core.plugin.web.jcaptcha	8

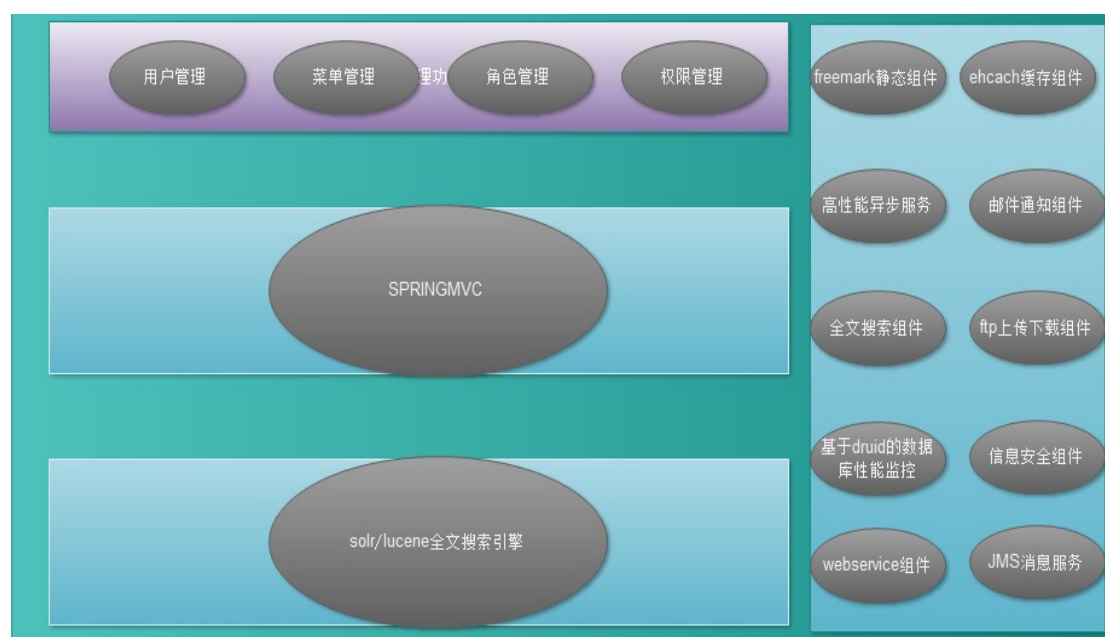
2.27	com.mtools.core.plugin.web.listener	8
2.28	com.mtools.core.plugin.web.servlet.init	8
2.29	com.mtools.core.plugin.netty.client	8
2.30	com.mtools.core.plugin.netty.server	8
3.	下列配置文件的用处	9
3.1	.src/main/resources.....	9
1、	common-mvc.xml.....	9
2、	mtools.params.properties	9
3、	mtools.default.properties.....	9
4、	log4j.properties.template	9
5、	mtools-monitor.xml.....	9
6、	mtools.beans.xml	9
7、	mtools.cache.xml.....	9
8、	alipay-druid-datasource.xml	9
9、	mtools.datasource01.xml	10
10、	mtools.datasource02.xml	10
11、	mtools.task.xml	10
3.2	conf 目录下的 template.....	10
1.	core-config.xml.template	10
2.	db.params.properties.template.....	10
3.	ehcache.xml.template	10
4.	ext.dic.template	10
5.	IKAnalyzer.cfg.xml	10
6.	log4j.properties.template.....	10
7.	schema.xml.template	11
8.	stopword.dic	11
9.	web-mvc.xml.template.....	11
10.	web-root.xml.template.....	11
11.	web.params.properties	11
12.	web.xml.template	11

4.使用教程	12
1 . 打开 eclipse，分别导入两个 maven 工程：mparent 和 mtools	12
2 . 新建 maven 工程（也就是将要开发的项目工程）	13
3 . 加入依赖工程 mtools	14
4 . 复制配置文件	15
5 . 根据项目需要修改配置	16
6 . 设置数据源参数	16
7.设置 web.params.properties 参数	17
8.配置 web.xml	17
5.关于作者	19

1.Mtools 介绍

Mmtools 是一个开源的，插件式的，多功能的 Java 开发工具框架，mtools 的目的是简化 spring 项目的配置，但不重复造轮子，充分利用现有的一些框架技术,适用于 Java se 项目以及 Java ee 项目，该插件整合了 springMVC 框架，以及 ehcache、freemarker、lucene/solr 等开源框架，简单简洁，各种项目中应用到的技术，都在该插件中融合，不需要做过多复杂的配置，项目中已经对各种配置进行了整合，仅仅需要新的工程中，引入 core-config.xml 配置文件即可，当然，可以自由配置各自项目中需要的个性化配置，对于初学或者初用 ehcache、freemarker、lucenu、solr 等技术的开发人员来说，插件中已经提供了现成的使用方法，只要依葫芦画瓢即可，插件中还提供了常用的工具包，如安全方面的加解密，签名验签的工具类，字符串处理，字节处理，日期处理等等日常开发中应用到的基本工具包

1.1 插件结构设计



2 . 插件中各包的用处

2.1 com.mtools.core.plugin

插件基类，除web包下面的插件以外，其他的插件都可以继承该基类，以便具备一些基本的能力

2.2 com.mtools.core.plugin.annotation

该包下是一些自定义的注解，如：免登录注解@AuthLogin 免权限验证注解@AuthAccess 以及获取当前登录用户对象注解：@CurrentUser

2.3 com.mtools.core.plugin.aop.aspect

切面编程例子

2.4 com.mtools.core.plugin.auth

登录授权验证、角色管理等

2.5 com.mtools.core.plugin.constant

插件用到的常量

2.6 com.mtools.core.plugin.db

数据库工具

2.7 com.mtools.core.plugin.ehcache

ehcache缓存工具

2.8 com.mtools.core.plugin.entity

插件用到的实体类

2.9 com.mtools.core.plugin.entiy.vo

用于返回页面显示的视图类

2.10 com.mtools.core.plugin.freemark

页面静态化工具类，用于生成静态html

2.11 com.mtools.core.plugin.helper

常用工具箱，如日期处理，xml处理，json处理，加解密处理等等

2.12 com.mtools.core.plugin.lucene

lucene搜索工具类，用于建立索引，全文搜索

2.13 com.mtools.core.plugin.mail

邮件工具，用于发送邮件通知

2.14 com.mtools.core.plugin.monitor

系统监控插件，监控系统内存，线程数等等

2.15 com.mtools.core.plugin.notify

通知服务，如一般邮件通知，系统异常通知

2.16 com.mtools.core.plugin.optlog

系统日志插件，用于记录访问痕迹

2.17 com.mtools.core.plugin.properties

系统参数配置

2.18 com.mtools.core.plugin.security

系统安全插件，用于加解密

2.19 com.mtools.core.plugin.solr

solr搜索客户端插件，用于建立solr索引，全文查询

2.20 com.mtools.core.plugin.spring.resolver

使用springresolver的方式自定义注解

2.21 com.mtools.core.plugin.staticres

静态文件，html生成插件

2.22 com.mtools.core.plugin.sys

2.23 com.mtools.core.plugin.task

系统自动任务插件，用于执行某些定时任务

2.24 com.mtools.core.plugin.web.filter

系统过滤器，如异常过滤器，缓存过滤器

2.25 com.mtools.core.plugin.web.interceptor

系统拦截器，如session拦截器

2.26 com.mtools.core.plugin.web.jcaptcha

验证码插件

2.27 com.mtools.core.plugin.web.listener

系统监听器

2.28 com.mtools.core.plugin.web.servlet.init

系统初始化servlet，用于初始化基础信息，配置参数等

2.29 com.mtools.core.plugin.netty.client

高性能异步请求netty客户端

2.30 com.mtools.core.plugin.netty.server

高性能异步请求netty服务端

3. 下列配置文件的用处

3.1 .src/main/resources

1、common-mvc.xml

springMvc公共配置文件，在mvc配置文件中引入该公共配置，该配置已经包含了一些基础拦截器的配置，视图的配置，静态资源映射等等

2、mtools.params.properties

系统基础属性，如，线程池的参数值

3、mtools.default.properties

默认配置，在新的工程里面应当覆盖这些参数

4、log4j.properties.template

log4j配置文件模版，每一个应用应该有且只有一个log4j配置文件

5、mtools-monitor.xml

aop配置文件

6、mtools.beans.xml

该组件用到的基本bean配置

7、mtools.cache.xml

系统缓存配置文件，对ehcache的配置

8、alipay-druid-datasource.xml

数据库的配置，事务配置等，使用了druid数据连接池

9、mtools.datasource01.xml

数据库的配置，事务配置等

10、mtools.datasource02.xml

数据库的配置，事务配置等

11、mtools.task.xml

自动任务配置，无特殊情况，无需修改该配置文件

3.2 conf 目录下的 template

在建立新的工程时，把该目录的 **template** 复制到新工程的 **src/main/resources** 目录下，并把“**.template**”后缀去掉，然后修改相关参数即可使用

1. core-config.xml.template

总配置文件，该配置文件引入了该组件所有需要的配置，当新的工程引入该组件时，直接在新工程的配置文件中引入该配置文件即可，此时不需要在引入下面的配置文件

2. db.params.properties.template

数据库配置，新工程中，修改相应的数据库用户名密码地址数据库类型即可

3. ehcache.xml.template

新项目中需要用到的缓存配置，仿照配置中的 **testCache** 配置，复制并修改缓存名即可完成添加新的缓存配置

4. ext.dic.template

IKAnalyzer 分词器的拓展词汇字典，可以在该文件中添加新的词汇

5. IKAnalyzer.cfg.xml

IKAnalyzer 分词器的核心配置，直接复制过去即可（）

6. log4j.properties.template

log4j 日志模版，如果只是简单的打印日志，直接去掉“**.template**”即可使用

7. schema.xml.template

solr collection 中的 conf 目录下的 schema,此模版加入了 IKAnalyzer 分词

8. stopwords.dic

IKAnalyzer 分词器的默认停止词典，不必修改，直接复制过去即可

9. web-mvc.xml.template

spring mvc 配置，在 web.xml 中将引入该配置文件

如下图：

```
<servlet>
  <servlet-name>spring</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:web-mvc.xml</param-value>
  </init-param>
</servlet>
```

10. web-root.xml.template

web 工程中引入配置文件的入口，在 web.xml 中配置

如下图：

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema"
3 <!-- 该配置文件复制到web工程的WEB-INF目录下 -->
4   <display-name>mweb</display-name>
5   <!-- Spring配置文件开始 -->
6   <context-param>
7     <param-name>contextConfigLocation</param-name>
8     <param-value>
9       classpath*:web-root.xml
10    </param-value>
11  </context-param>
12  <listener>
13    <listener-class>org.springframework.web.cont
14  </listener>
```

11. web.params.properties

该属性配置文件，包含一些如：静态资源目录的配置，缓存地址的配置，solr 地址的配置等等，如果需要使用到相关功能，请去掉注释，并设置相应的值

12. web.xml.template

web.xml 配置文件，已经配置了 spring 的相关配置，以及缓存，压缩的配置等，但如果需要使缓存和压缩响应生效，须根据实际情况，修改里面的 uri

如下图：

```

<!--页面缓存-->
<filter>
    <filter-name>helperApiCacheFilter</filter-name>
    <filter-class>com.mtools.core.plugin.web.filter.SimplePageCachingExtFilter</filter-class>
    <init-param>
        <param-name>cacheName</param-name>
        <param-value>helperApiWebCache</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>helperApiCacheFilter</filter-name>
    <url-pattern>/helper/searchhqli</url-pattern>
</filter-mapping>
<filter-mapping>
    <filter-name>helperApiCacheFilter</filter-name>
    <url-pattern>/helper/viewqf</url-pattern>
</filter-mapping>
<!--
    响应页面进行压缩
-->
<filter>
    <filter-name>CompressingFilter</filter-name>
    <filter-class>com.planetj.servlet.filter.compression.CompressingFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>CompressingFilter</filter-name>
    <url-pattern>/helper/supbanklist/*</url-pattern>
</filter-mapping>

```

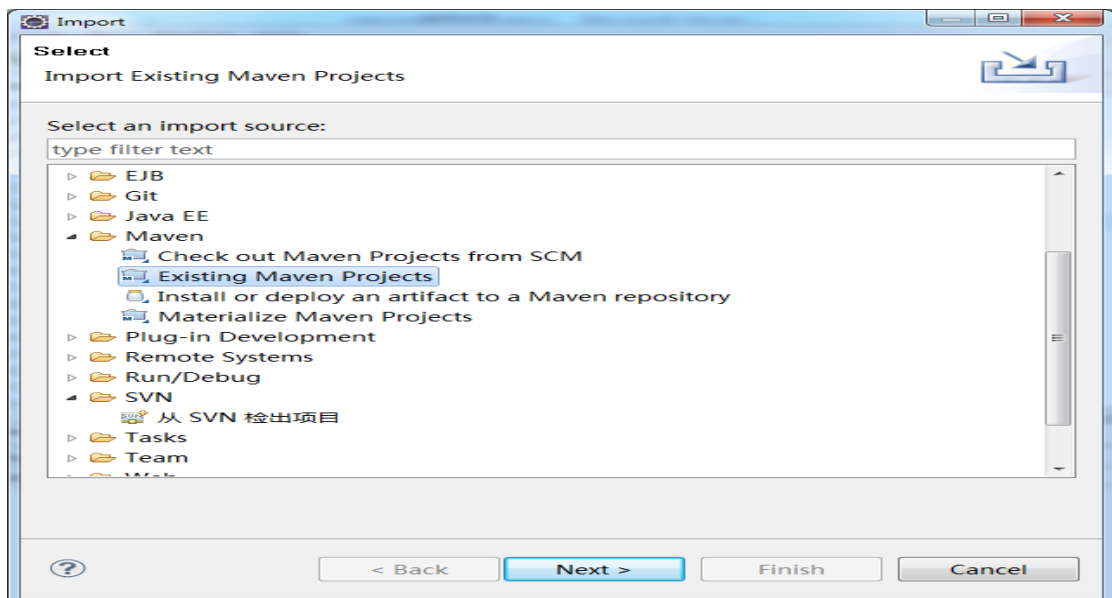
要过滤的uri

要对响应信息进行压缩的uri

4.使用教程

1. 打开 eclipse , 分别导入两个 maven 工程 : mparent 和 mtools

注意, 需要以 maven 工程的形式导入, 如下图:



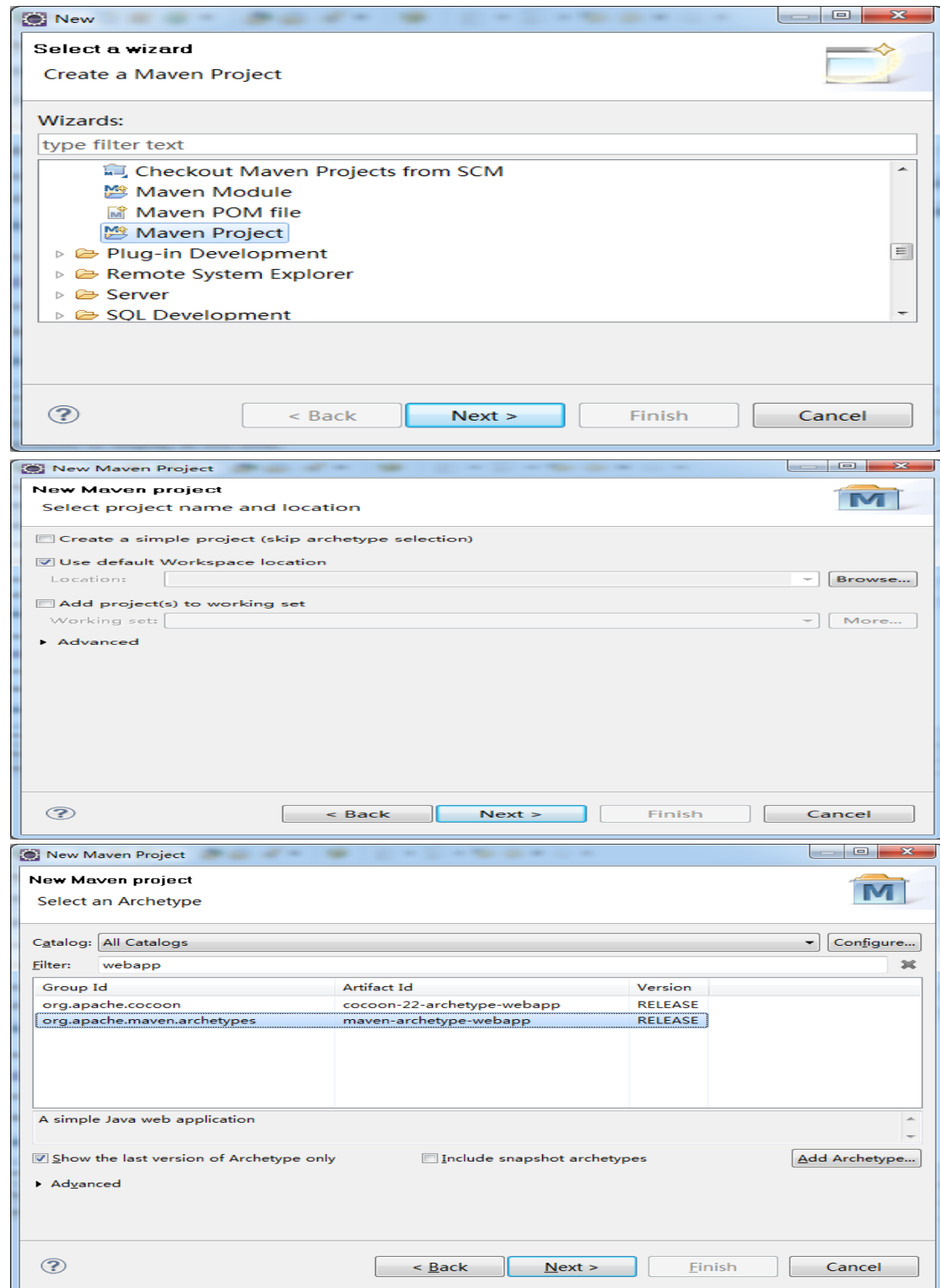
温馨提示:

初次导入这两个工程, 可能会由于本地资源仓库缺少一些包, 或者maven第一次下载太多依赖jar时, 导致其中一些jar没能完成下载成功, 而导致工程报错。此时, 从pom配置文件中找出缺少的jar包, 然后到本地资源仓库中把该jar的目录删除, 这个时候再次update dependence, 那么maven会重新下载jar, 一般再次下

载的时候便可以成功

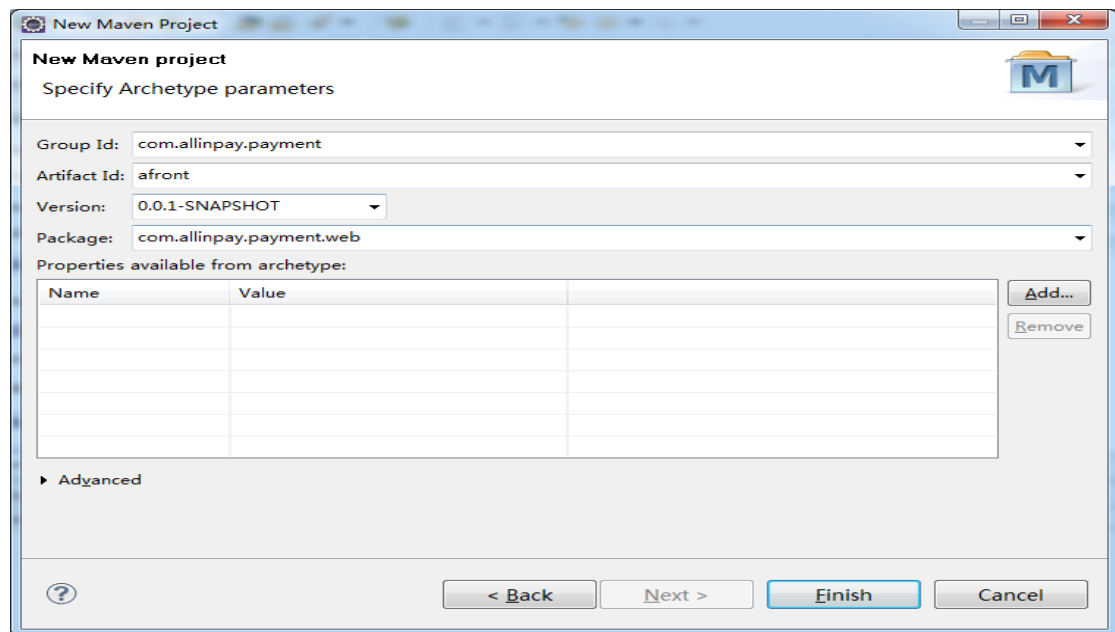
2. 新建 maven 工程（也就是将要开发的项目工程）

如图：



此处新建一个web工程，选择group id为：org.apache.maven.archetypes
Artifactid 为 maven-archetype-webapp

然后下一步

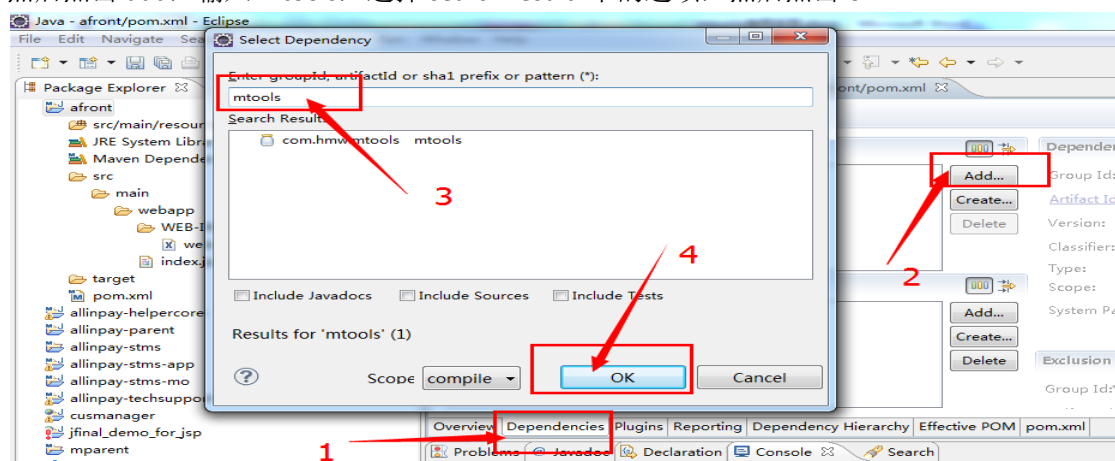


此处填写新工程的groupid artifactid 等信息，具体名字自己定义，然后finish
此时，eclipse会建立一个标准的maven webapp工程，如下图：

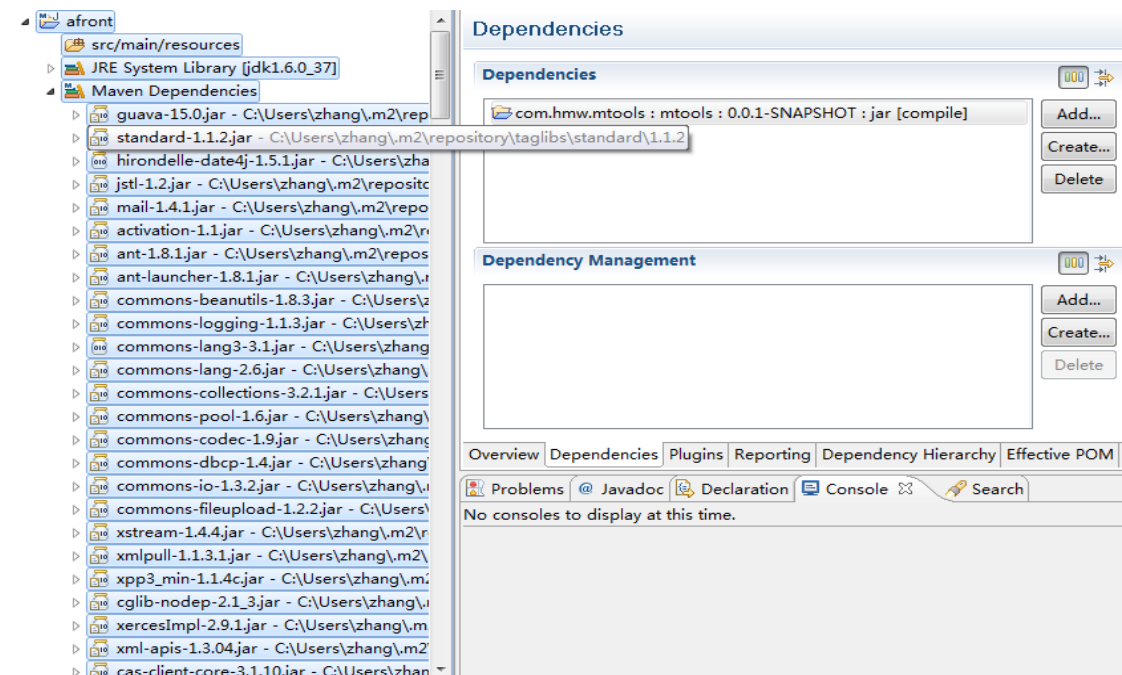


3. 加入依赖工程 mtools

以 maven pom.xml 的打开方式，打开 afront 工程下的 pom.xml 文件，然后点击 Dependencies，
然后点击 add，输入 mtools，选择 search result 下的选项，然后点击 ok

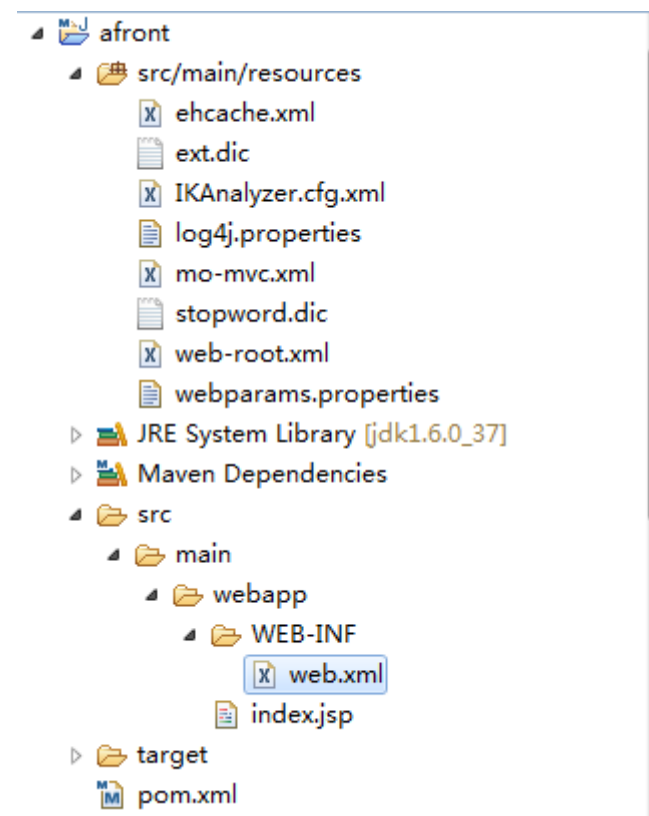


此时会发现，afront工程的maven dependence中把mtools所依赖的jar都关联进来了，包括mtools本身



4 . 复制配置文件

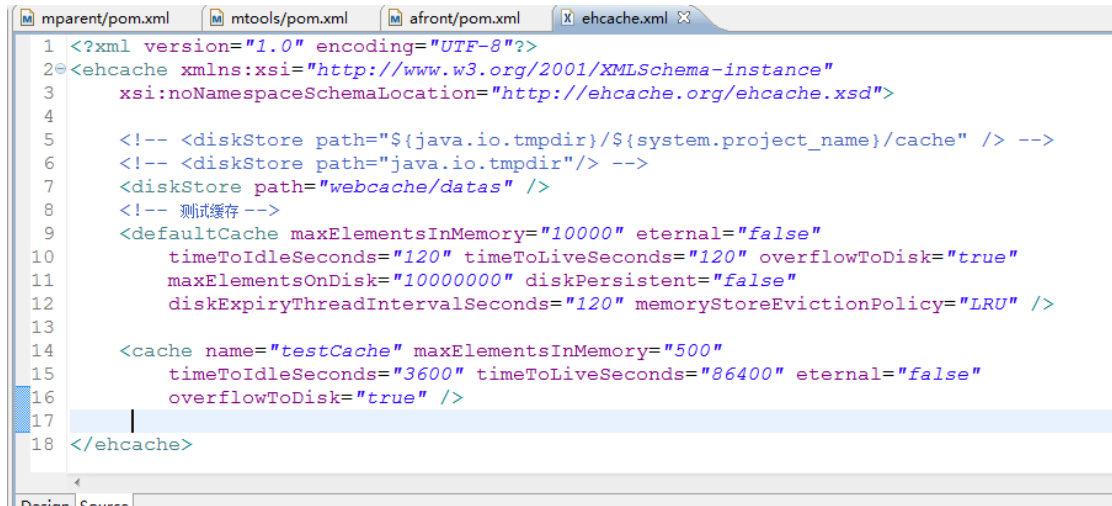
复制mtools工程下conf目录中的template文件到新工程afront的src/main/resources目录下，把”.template”后缀去掉，另外，把web.xml覆盖到WEB-INF下面的web.xml如下图：



5. 根据项目需要修改配置

5.1 新增项目需要的 cache

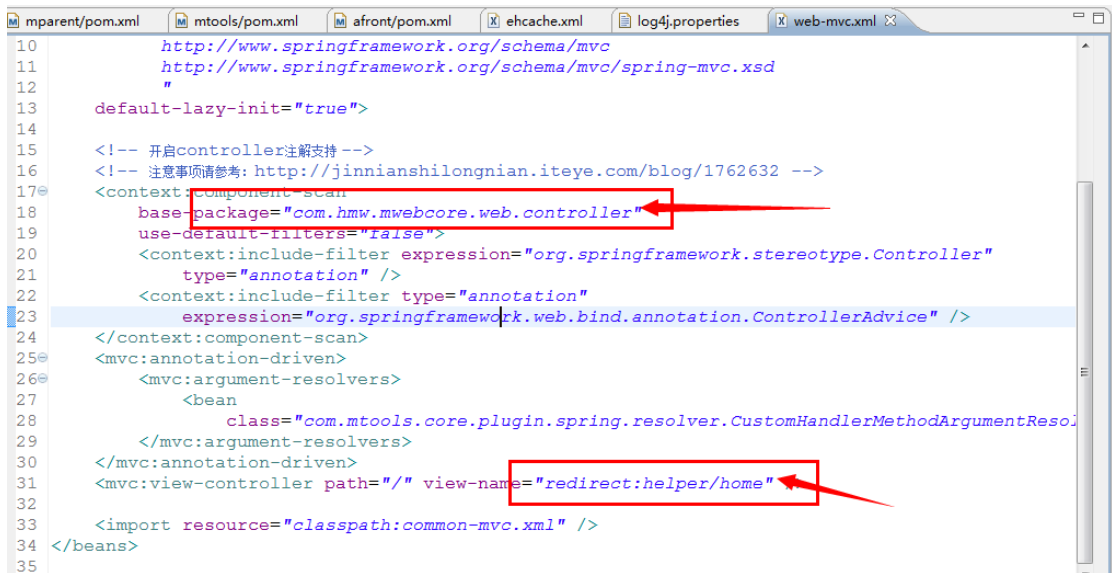
参考已有的 testCache 配置，复制，修改名称即可



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3       xsi:noNamespaceSchemaLocation="http://ehcache.org/ehcache.xsd">
4
5     <!-- <diskStore path="${java.io.tmpdir}/${system.project_name}/cache" /> -->
6     <!-- <diskStore path="java.io.tmpdir"/> -->
7     <diskStore path="webcache/datas" />
8     <!-- 测试缓存 -->
9     <defaultCache maxElementsInMemory="10000" eternal="false"
10        timeToIdleSeconds="120" timeToLiveSeconds="120" overflowToDisk="true"
11        maxElementsOnDisk="10000000" diskPersistent="false"
12        diskExpiryThreadIntervalSeconds="120" memoryStoreEvictionPolicy="LRU" />
13
14     <cache name="testCache" maxElementsInMemory="500"
15        timeToIdleSeconds="3600" timeToLiveSeconds="86400" eternal="false"
16        overflowToDisk="true" />
17
18 </ehcache>
```

5.2 修改 web-mvc.xml

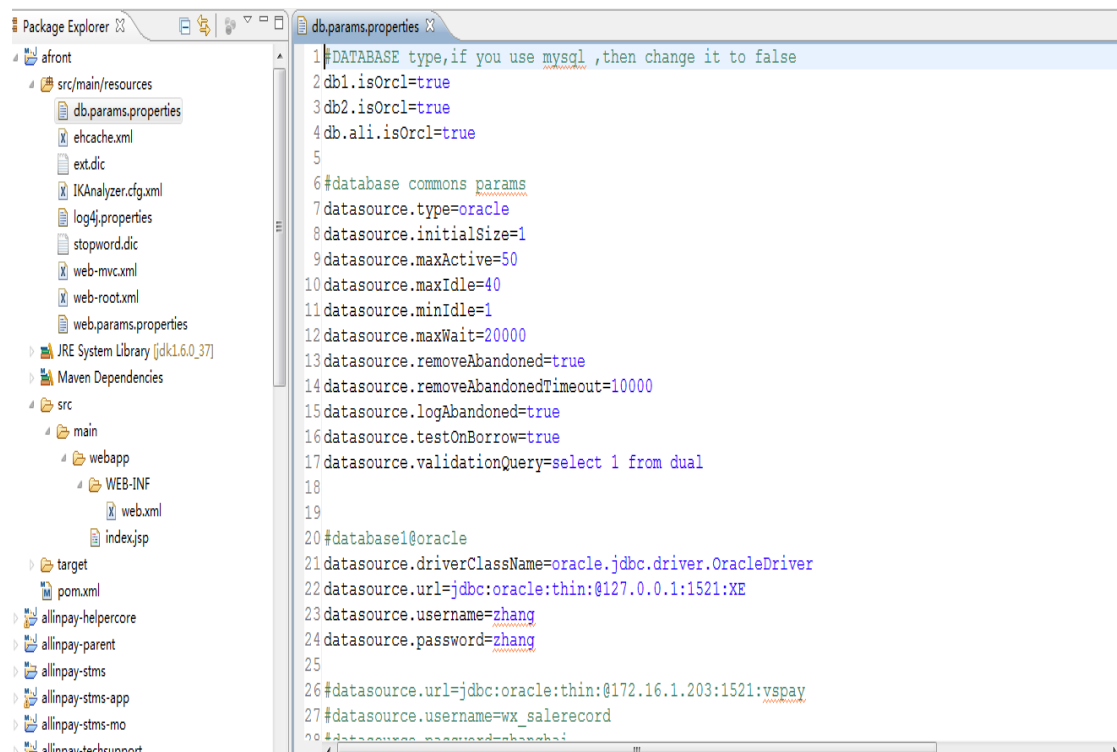
配置 web 工程 controller 包路路径，以及/跳转的视图



```
10 http://www.springframework.org/schema/mvc
11 http://www.springframework.org/schema/mvc/spring-mvc.xsd
12 "
13 default-lazy-init="true">
14
15 <!-- 开启controller注解支持 -->
16 <!-- 注意事项请参考: http://jinnianshilongnian.iteye.com/blog/1762632 -->
17 <context:component-scan
18     base-package="com.hmw.mwebcore.web.controller"
19     use-default-filters="false">
20     <context:include-filter expression="org.springframework.stereotype.Controller"
21         type="annotation" />
22     <context:include-filter type="annotation"
23         expression="org.springframework.web.bind.annotation.ControllerAdvice" />
24 </context:component-scan>
25 <mvc:annotation-driven>
26     <mvc:argument-resolvers>
27         <bean
28             class="com.mtools.core.plugin.spring.resolver.CustomHandlerMethodArgumentResol
29         </mvc:argument-resolvers>
30 </mvc:annotation-driven>
31 <mvc:view-controller path="/" view-name="redirect:helper/home"
32 </import resource="classpath:common-mvc.xml" />
33 </beans>
34
35
```

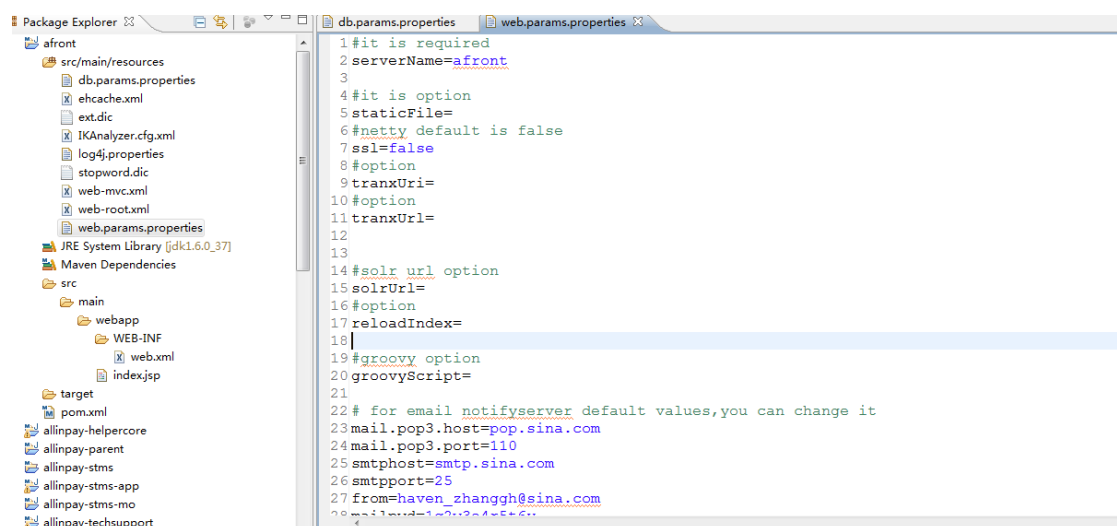
6. 设置数据源参数

根据自己的环境，配置自己的数据库用户名地址等



7.设置 web.params.properties 参数

如下图，注释为 option 的参数均为可选参数，默认为空值



8.配置 web.xml

a.根据项目需要，配置错误返回视图

```
db.params.properties  web.params.properties  web.xml  web.xml.template  RequestContextListener.class
85
86 <!-- 错误页面映射 -->
87 <error-page>
88     <error-code>404</error-code>
89     <location>/WEB-INF/jsp/error/error.jsp</location>
90 </error-page>
91 <error-page>
92     <error-code>400</error-code>
93     <location>/WEB-INF/jsp/error/error.jsp</location>
94 </error-page>
95 <error-page>
96     <error-code>500</error-code>
97     <location>/WEB-INF/jsp/error/error.jsp</location>
98 </error-page>
99 <error-page>
100     <error-code>503</error-code>
101     <location>/WEB-INF/jsp/error/error.jsp</location>
102 </error-page>
103 <error-page>
104     <exception-type>java.lang.Exception</exception-type>
105     <location>/WEB-INF/jsp/error/error.jsp</location>
106 </error-page>
107
108 <!-- 初始化servlet -->
109 <servlet>
110     <servlet-name>InitServlet</servlet-name>
```

b. 根据需要，配置 web 缓存

注意，此处的缓存名必须先在 ecache.xml 文件中配置

```
db.params.properties  web.params.properties  web.xml  web.xml.template  RequestContextListener.class
151 </session-config>
152 <!-- 页面缓存 -->
153 <filter>
154     <filter-name>helperApiCacheFilter</filter-name>
155     <filter-class>com.mtools.core.plugin.web.filter.SimplePageCachingExtFilter</filter-class>
156     <init-param>
157         <param-name>cacheName</param-name>
158         <param-value>helperApiWebCache</param-value>
159     </init-param>
160 </filter>
161 <filter-mapping>
162     <filter-name>helperApiCacheFilter</filter-name>
163     <url-pattern>/helper/searchhqlist</url-pattern>
164 </filter-mapping>
165 <filter-mapping>
166     <filter-name>helperApiCacheFilter</filter-name>
167     <url-pattern>/helper/viewqfile</url-pattern>
168 </filter-mapping>
169 <filter-mapping>
170     <filter-name>helperApiCacheFilter</filter-name>
171     <url-pattern>/helper/addhotqs</url-pattern>
172 </filter-mapping>
173 <filter-mapping>
174     <filter-name>helperApiCacheFilter</filter-name>
175     <url-pattern>/helper/basisdef/*</url-pattern>
176 </filter-mapping>
```

c. 配置服务端响应压缩过滤器

```
<!-- 响应页面进行压缩 -->
<filter>
    <filter-name>CompressingFilter</filter-name>
    <filter-class>com.planetj.servlet.filter.compression.CompressingFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>CompressingFilter</filter-name>
    <url-pattern>/helper/supbanklist/*</url-pattern>
</filter-mapping>
```

如上图是指过滤 uri 为: `/helper/supbanklist/` 下面的所有请求, 此时符合该过滤要求的响应都会以 GZIP 的压缩格式返回, 大大提升了客户端的响应速度

到目前为止, 整个项目的配置就完毕了

5.关于作者

Blog : <http://kanckzhang.iteye.com/>

Email : kanckzhang@163.com

Github: <https://github.com/zhanggh/mtools>