

MgFinal 使用手册

版本： 1.0

作者： 梅刚

日期： 2016-05-13

QQ： 1092017732

<http://git.oschina.net/mgang/mgfinal>

目录

第一章.快速入门.....	4
1.1.创建项目.....	4
1.2.加入 mgfinal 依赖 jar 文件.....	7
1.3.配置 web.xml.....	8
1.4.加入需要的配置文件.....	8
1.4.1.新建 source floder 源码文件夹 config.....	8
1.4.2.将配置文件 mgfinal.propertis 放入.....	8
.....	8
1.4.3.加入 jdbc.properties 配置文件.....	9
1.4.4.加入 log4j.properties 配置文件.....	9
1.4.5.加入 mybatis 核心配置文件.....	10
1.5.创建 mg_user 表.....	12
1.6.加入测试代码.....	12
1.6.1.建立测试包.....	12
1.6.2.建立测试类.....	12
1.6.2.1.index.jsp.....	13
1.6.2.2.IndexAction.java.....	13
1.6.2.3.加入页面.....	13
1.7.启动项目.....	14
1.8.输入访问路径.....	14
第二章.控制跳转.....	15
2.1.实现原理.....	15
2.2.功能支持.....	15
2.2.1.增强 request.....	15
2.2.2.增强参数封装.....	15
2.2.3.增强 response.....	16
2.3.实例代码.....	16
2.4.附上 demo 代码.....	17
第三章.依赖注入.....	19
3.1.实现原理.....	19
3.2.实现方式.....	19
3.3.提供 2 中容器.....	19
3.4.实例代码.....	20
第四章.mybatis 的 ORM 拓展.....	21
4.1.集成初衷.....	21
4.2.功能支持.....	21
4.3.功能试用.....	22
4.3.1.基于动态 sql 的 CRUD 方法的使用.....	22
4.3.2.事务支持调用.....	23
4.3.3.数据源支持.....	23
4.3.4.基于 mybatis-ext 实现的对象的 CRUD 方法.....	25
4.3.5.pagehelper 分页.....	26

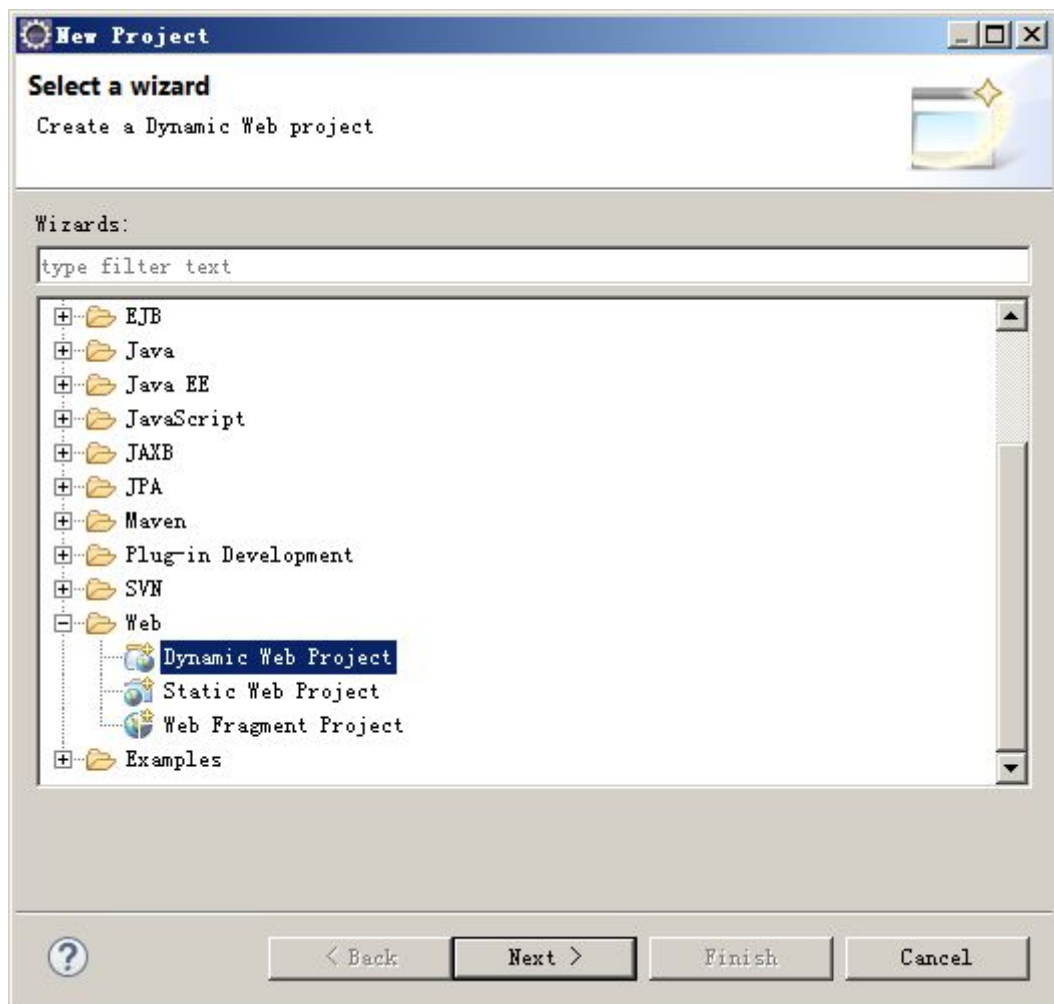
4. 3. 6. 基于 mybatis-redis 实现 Mybatis 的查询二级缓存.....	27
---	----

第一章.快速入门

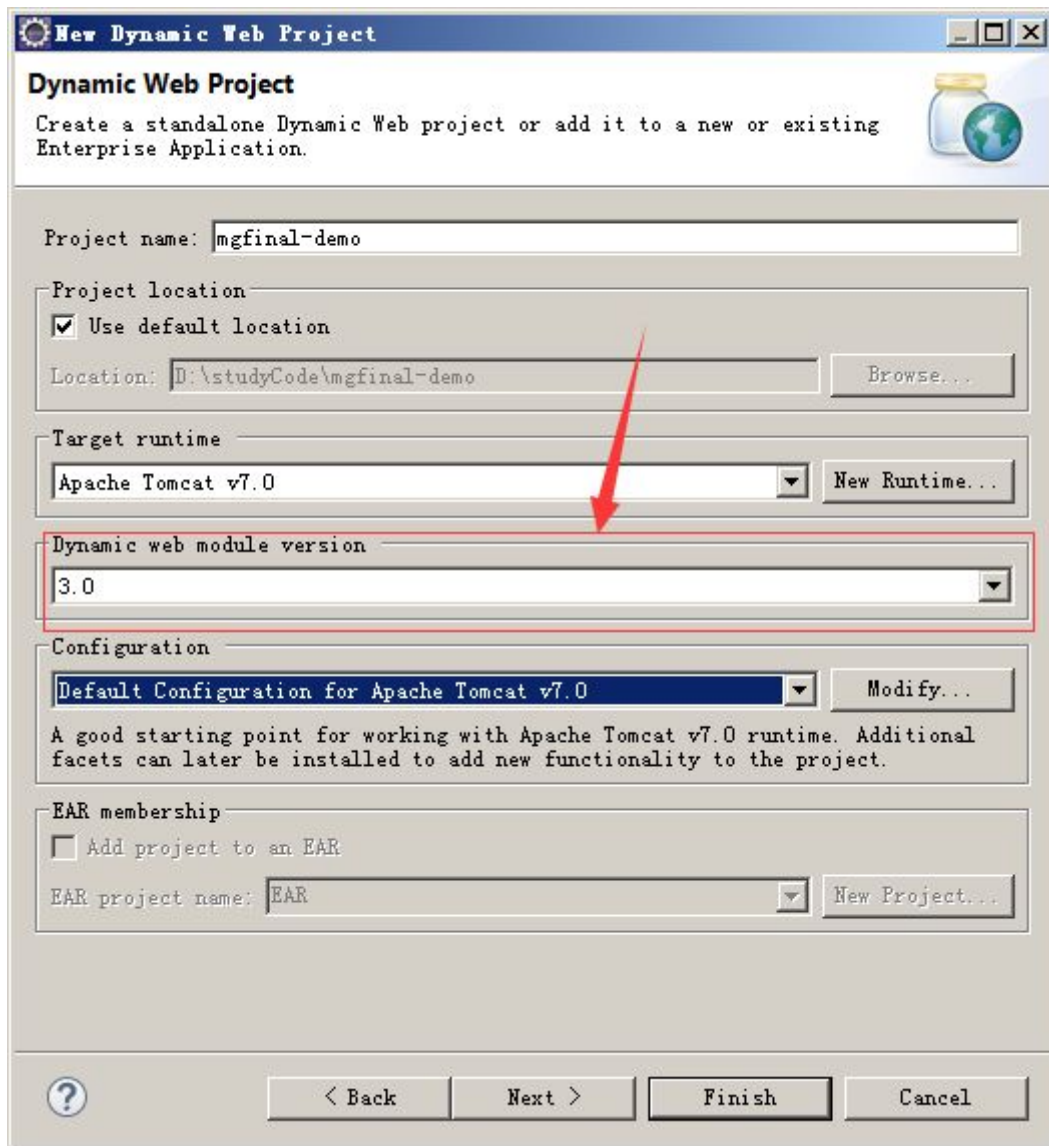
1.1.创建项目

现如今多是使用 eclipse 来作为 java 的 IDE 环境。这里就用 eclipse 最新版 marks 来做演示。

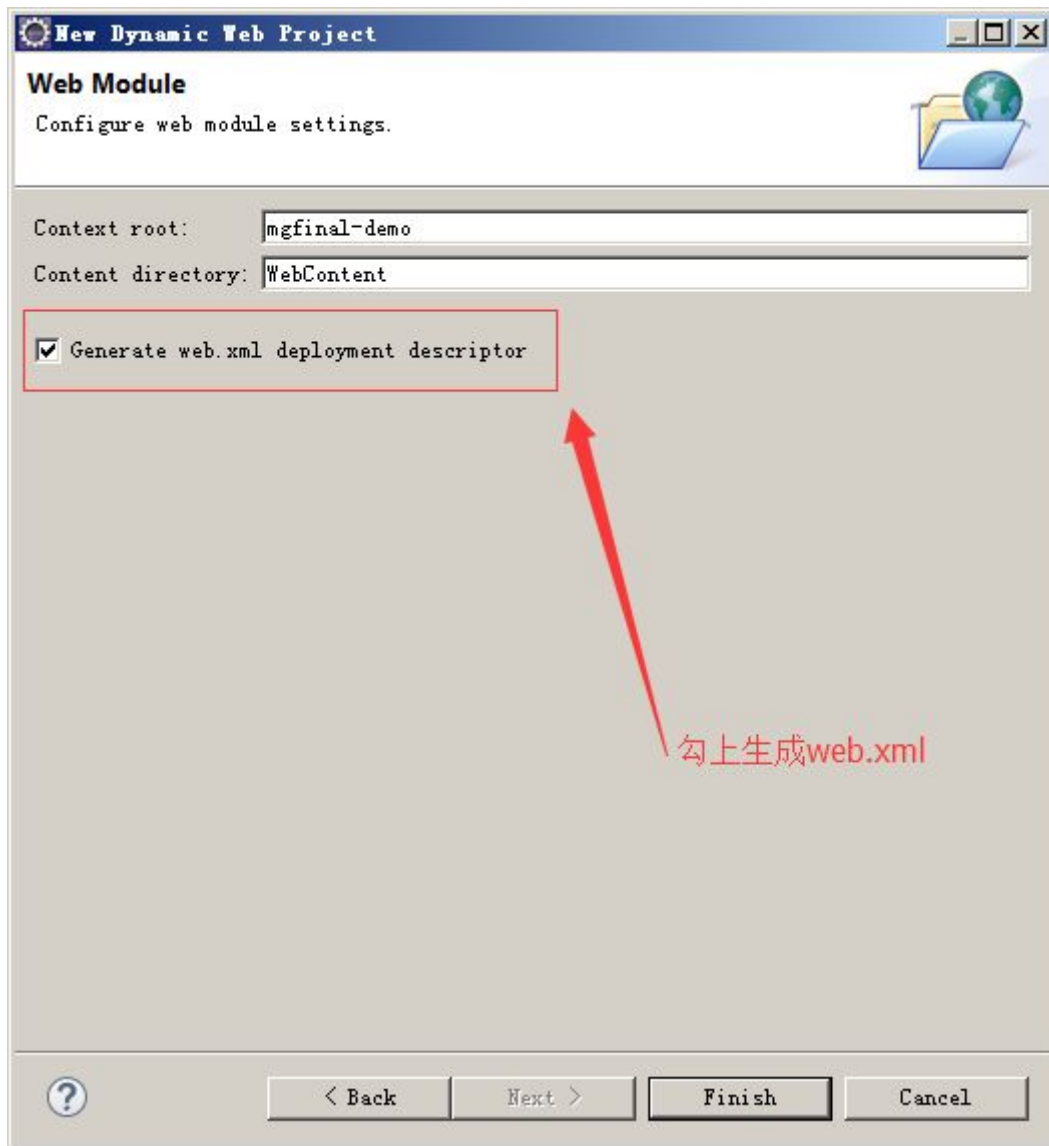
(1) 新建 Dynamic Web Project 动态 web 工程。



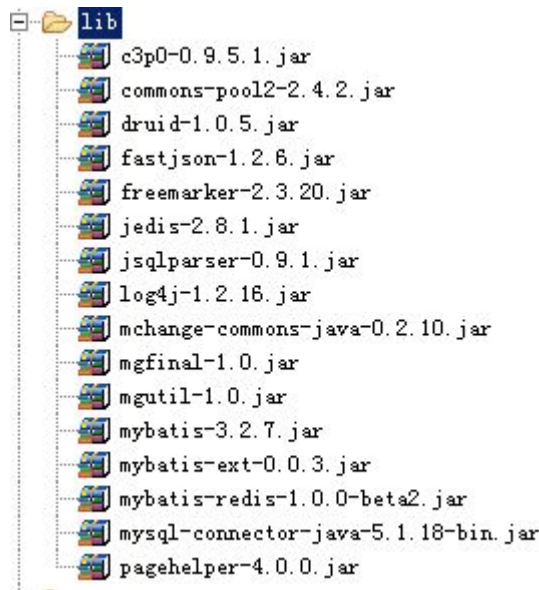
(2)注意使用 web module version 3.0, 因为 mgfinal 使用到 Servlet3.0.



(3) 勾选生成 web.xml



1.2.加入 mgfinal 依赖 jar 文件



其中有些功能未使用到，可以不用加入，这里加入的是所有可能依赖的 jar。

mgfinal 依赖的 jar 文件，需要是添加。

- 1: mgutil-1.0.jar mgfinal 的通用工具集
- 2: freemarker-2.3.20.jar 支持 FreeMarker 视图类型
- 3: cos-26Dec2008.jar 支持文件上传功能
- 4: mysql-connector-java-5.1.20-bin.jar 支持 mysql 数据库
- 5: druid-1.0.5.jar 支持 Druid 数据库连接池
- 6: c3p0-0.9.5.1.jar、mchange-commons-java-0.2.10.jar 数据库连接池 c3p0
- 7: junit-4.10.jar 单元测试
- 8: fastjson-1.2.6.jar json 操作工具 fastjson
- 9: log4j-1.2.16.jar 支持 log4j 日志，需要配置类路径下 log4j.properties
- 10: mybatis-3.2.7.jar mybatis 主文件支持
- 11: mybatis-ext-0.0.3.jar 支持 mybatis 基于对象 crud 操作
- 12: mybatis-pagehelper [jsqlparser-0.9.1.jar,pagehelper-4.0.0.jar 用来做 mybatis 分页]
- 13: mybatis-redis[commons-pool2-2.4.2.jar,mybatis-redis-1.0.0.-beta2.jar, jedis-2.8.1.jar 作为 mybatis 查询二级缓存]

注意：以上是全部依赖，未使用到的功能，可以不用加入该 jar。

如数据源，只需要选择一种。

1.3.配置 web.xml

配置启动 mgfinal ioc 容器

```
<!-- mgfinal ioc 启动 -->
<listener>
  <listener-class>com.mgfinal.core.ioc.context.IocListener</listener-class>
</listener>
```

1.4.加入需要的配置文件

1.4.1.新建 source floder 源码文件夹 config

1.4.2.将配置文件 mgfinal.propertis 放入

```
#mgwork 框架配置文件
#网页存放文件夹前缀
mgfinal.webfolder.prefix = /WEB-INF/pages
#网页文件后缀
mgfinal.web.page.stuffix = .html
#默认视图
mgfinal.web.view.type = freemarker
#ioc 扫描路径配置, 默认 src 下
mgfinal.ioc.scan.package = com
#mgioc 容器类型 map,redis, 默认是 map
mgfinal.ioc.type = map
#redis 服务主机
mgfinal.ioc.redis.host = localhost
#redis 服务端口
mgfinal.ioc.redis.port = 6379
```

其中有些配置可能没有用到, 但是是会有用的。这是 1.0 版本比较全的配置文件。

1.4.3.加入 jdbc.properties 配置文件

```
#mysql#  
username=root  
password=  
driver=com.mysql.jdbc.Driver  
url=jdbc:mysql://127.0.0.1:3306/test?autoReconnect=true&useUnicode=true&characterEncoding=UTF8
```

需要根据自己的数据库，做相应调整。

1.4.4.加入 log4j.properties 配置文件

主要是用来做日志输出的。包括配置数据 mybatis 执行的 sql 语句。

```
log4j.rootLogger=DEBUG, Console  
  
#Console  
log4j.appender.Console=org.apache.log4j.ConsoleAppender  
log4j.appender.Console.layout=org.apache.log4j.PatternLayout  
log4j.appender.Console.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss} - %m%n  
  
freemarker.cache = ERROR  
  
log4j.logger.java.sql.ResultSet= DEBUG  
log4j.logger.org.apache=ERROR  
log4j.logger.java.sql.Connection=INFO  
log4j.logger.java.sql.Statement=DEBUG  
log4j.logger.java.sql.PreparedStatement=DEBUG
```

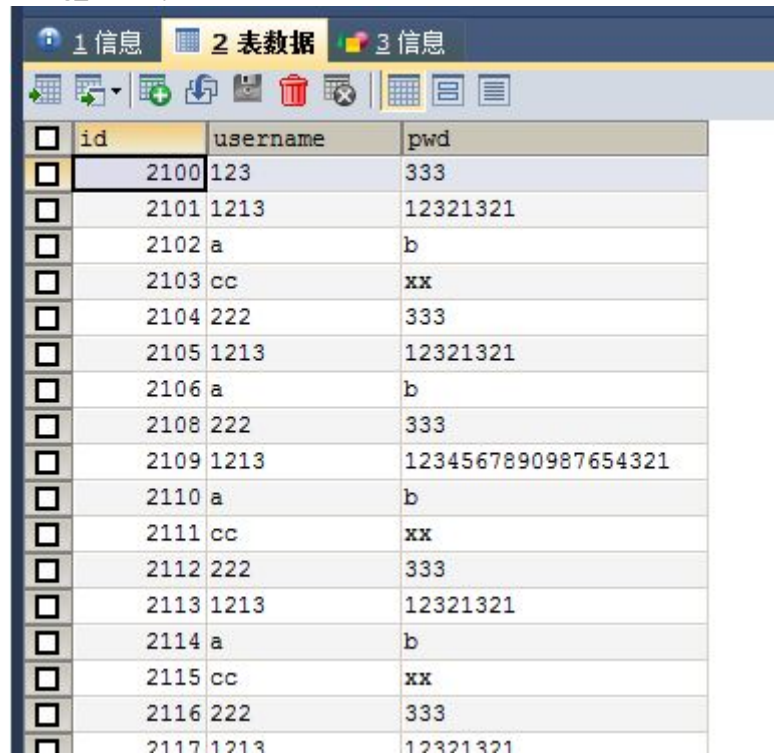
1.4.5.加入 mybatis 核心配置文件

名字是 mybatis.xml。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
  <properties resource="jdbc.properties"/>
  <!-- 分页插件 -->
  <plugins>
    <plugin interceptor="com.github.pagehelper.PageHelper">
      <property name="dialect" value="mysql"/>
      <property name="pageSizeZero" value="true"/>
      <property name="reasonable" value="true"/>
    </plugin>
  </plugins>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC" />
      <!-- mybatis 提供的数据源 -->
      <dataSource type="POOLED">
        <property name="driver" value="${driver}" />
        <property name="url" value="${url}" />
        <property name="username" value="${username}" />
        <property name="password" value="${password}" />
      </dataSource>
    </environment>
  </environments>
</configuration>
```


1.5.创建 mg_user 表

表结构如下：（mg_user.sql 执行可以获得）



<input type="checkbox"/>	id	username	pwd
<input type="checkbox"/>	2100	123	333
<input type="checkbox"/>	2101	1213	12321321
<input type="checkbox"/>	2102	a	b
<input type="checkbox"/>	2103	cc	xx
<input type="checkbox"/>	2104	222	333
<input type="checkbox"/>	2105	1213	12321321
<input type="checkbox"/>	2106	a	b
<input type="checkbox"/>	2108	222	333
<input type="checkbox"/>	2109	1213	1234567890987654321
<input type="checkbox"/>	2110	a	b
<input type="checkbox"/>	2111	cc	xx
<input type="checkbox"/>	2112	222	333
<input type="checkbox"/>	2113	1213	12321321
<input type="checkbox"/>	2114	a	b
<input type="checkbox"/>	2115	cc	xx
<input type="checkbox"/>	2116	222	333
<input type="checkbox"/>	2117	1213	12321321

1.6.加入测试代码

1.6.1.建立测试包

```
com.demo.action -- action 包  
com.demo.service -- service 包  
com.demo.dao -- dao 包  
com.demo.vo -- vo 包
```

1.6.2.建立测试类

1.6.2.1.index.jsp

首先因为在 web.xml 中配置的入口是 index.jsp，先在 webcontent 下加入 index.jsp 文件。
内容如下，作用让其跳转到 index.do/index 首页。

```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<jsp:forward page="index.do/index"></jsp:forward>
```

1.6.2.2.IndexAction.java

```
package com.demo.action;

import javax.servlet.annotation.WebServlet;

import com.mgfinal.core.mvc.core.MGWorkServlet;
@WebServlet("/index.do/*")
public class IndexAction extends MGWorkServlet{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public String index(){

        return "index";
    }

}
```

1.6.2.3.加入页面

在 web-info 下建立 pages 目录，并添加 index.html 页。

```
<!doctype html>
<meta charset="utf-8"/>
<title>welcome mgfinal</title>
```

欢迎来到 mgfinal!

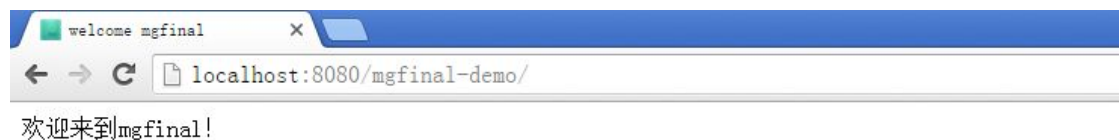
1.7.启动项目

```
信息: At least one JAR was scanned for TLDs yet contained no TLDs. Enable debug logging for this logger fo
2016-05-14 02:17:46 - mgfinal ioc init success.
2016-05-14 02:17:46 - mgfinal ioc type is map
2016-05-14 02:17:46 - mgfinal ioc --> {"ioc":[],"size":0}
五月 14, 2016 2:17:46 上午 org.apache.catalina.util.SessionIdGeneratorBase createSecureRandom
信息: Creation of SecureRandom instance for session ID generation using [SHA1PRNG] took [219] milliseconds
五月 14, 2016 2:17:46 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["http-bio-8080"]
五月 14, 2016 2:17:46 上午 org.apache.coyote.AbstractProtocol start
信息: Starting ProtocolHandler ["ajp-bio-8009"]
五月 14, 2016 2:17:46 上午 org.apache.catalina.startup.Catalina start
信息: Server startup in 2027 ms
```

启动成功。

1.8.输入访问路径

<http://localhost:8080/mgfinal-demo/>



第二章.控制跳转

2.1.实现原理

基于 Servlet3.0 的注解方式，来实现配置请求。然后通过反射获取/a/b/c.do/*来匹配 Action，/*表示进入该 Action 中的方法名，并实现跳转。

2.2.功能支持

支持表单封装 mgf2Object，mgf2Json，mgf2Map 方法。

支持跳转模板渲染，renderJsp，renderFreemarker，renderJson 等。默认模板，支持直接返回字符串来跳转。

支持 ajax 请求处理 ajaxJsonSuccess,ajaxJsonError。

对 request，response 通用方法的一些增强。

2.2.1.增强 request

方法	备注
void setAttr(String key,Object v)	实现 req.setAttribute，设置参数。
Object getAttr(String key)	实现 req.getAttribute，获取参数。
void setSessionAttr(String key,Object v)	实现 session.setAttribute,设置会话参数。
Object getSessionAttr(String key)	实现 session.getAttribute，获取会话参数。
String getPara(String key)	实现 req.getParameter，请求参数获取。
Integer getParaToInt(String key)	实现请求参数获取转 int 型。
Float getParaToFloat(String key)	实现请求参数获取转 float 型。
Double getParaToDouble(String key)	实现请求参数获取转 double 型。

2.2.2.增强参数封装

方法	备注
Object mgf2Object(Class c)	实现将请求参数封装为对象 c。
JSONObject mgf2Json()	事项将请求参数封装为 json 对象。
Map<String,String> mgf2Map()	事项将请求参数封装为 map
注意: <u>//checkbox</u> ， <u>mutli select</u> 的属性会转成逗号间隔的字符串	

2.2.3.增强 response

方法	备注
void ajaxJsonSuccess(Object obj)	具体返回: {'state':200,'data':obj,'msg':' 操作成功'};
void ajaxJsonError(Object obj)	具体返回: {'state':0,'data':obj,'msg':'操作失败'};
void renderJson(Object obj)	为请求响应 json 数据
void renderJsp(String view)	渲染模板视图, jsp.
void renderFreemarker(String tpl)	渲染模板视图, freemarker.
注意: render 方法和默认的 return “index”,是会有默认前缀和后缀的支持的。配置在 mgfinal.properties 中的: #网页存放文件夹前缀 mgfinal.webfolder.prefix = /WEB-INF/pages #网页文件后缀 mgfinal.web.page.stuffix = .html #默认视图 mgfinal.web.view.type = freemarker	

2.3.实例代码

```
@WebServlet("/index.do/*")  
public class IndexAction extends MGWorkServlet
```

这个是 IndexAction 的类头,使用 Servlet3.0 的配置注解,@WebServlet 来完成 servlet 的配置。所以 mgfinal 的 Action 实际上就是 servlet 的,其速度是最接近 servlet 的,也可以说是最快的响应。(大家都了解,struts2 使用拦截器来实现 Action,这样的速率是非常慢的,从响应上来说。)

```
public String index(){  
    return "index";  
}
```

以上是 mgfinal 中 Action 的一个方法,类似 strut2 中的。其中 HttpServletRequest 和 HttpServletResponse 对象都能继承得到,并拥有上述增加的方法功能。可以完成表单参数封装,插入 request,session 等值,后跳转到指定页面。
其中 return “index”;就是默认的跳转,如果 mgfinal.properties 中配置的是 /WEB-INF/pages,后缀.html 和模板是 freemarker 的话,就会跳转到 /WEB-INF/pages/index.html 的 freemarker 视图页。

2.4.附上 demo 代码

```
package mg.test;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import javax.servlet.annotation.WebServlet;
import org.apache.tomcat.util.security.MD5Encoder;
import com.alibaba.fastjson.JSONObject;
import mg.mvc.core.MGWorkServlet;
@WebServlet("/test.mg/*")
public class TestServlet extends MGWorkServlet{
    private static final long serialVersionUID = 1L;
    public String index(){
        return "index";
    }
    /**
     * test1 方法
     * @return 跳转到 test1.html
     */
    public String test1(){
        return "test1";
    }
    /**
     * test2 方法
     * @return 跳转到 test2.html
     */
    public String test2(){
        return "test2";
    }
    public String wrapfrm(){
        this.setSessionAttr("time", new Date().toLocaleString());
        User u = (User) mgf2Object(User.class);
        this.setAttr("user", u);
        return "freemarker/demo1";
    }
}
```

```

public void testjsp(){
    this.setSessionAttr("time", new Date().toLocaleString());
    User u = new User();
    u.setAge(12);
    u.setBirthday(new Date());
    u.setHobby("长街, 跳水");
    u.setPassword(MD5Encoder.encode("123456".getBytes()));
    u.setUsername("xiaogang");
    this.setAttr("user", u);
    renderJsp("jsp/demo");
}

public void testfreemarker(){
    this.setSessionAttr("time", new Date().toLocaleString());
    List<User> list = new ArrayList<User>();
    for(int i=0;i<10;i++){
        User u = new User();
        u.setAge(12+i);
        u.setBirthday(new Date());
        u.setHobby("长街, 跳水");
        u.setPassword(Math.random()+"");
        u.setUsername("xiaogang"+i);
        list.add(u);
    }
    this.setAttr("ulist", list);
    this.renderFreemarker("freemarker/demo");
}

public void testAjax(){
    JSONObject json = this.mgf2Json();
    this.ajaxJsonSuccess(json);
}
}

```

第三章.依赖注入

3.1.实现原理

依赖注入其实做的事很简单，就是在项目启动时，将一些常用的对象单例到一个管理容器中，然后等到要用的时候，注入到对象的属性中使用即可。

这样做，能够省掉 java 虚拟机频繁地创建对象，释放对象内存的操作，优化性能。

还有，mgfinal 采用的是运行时注入的方式，这样就支持多并发操作了。所谓运行时注入，就是当一个类中的属性要被使用时，就会通过容器来注入。这样做的缺点在于，会有频繁的注入操作，但是因为容器都是基于内存的，速度也是很快的，可以忽略不计。

3.2.实现方式

基于注解 @ToBean 和 @UseBean 这 2 个注解。

@ToBean 是类注解，使用该注解的类，会在项目启动时，初始化到容器中，项目关闭时，释放掉容器资源。

@UseBean 是属性注解，使用该注解的类，会在对象实例化的时候根据对象类型，从容器中注入到属性，不用自己 new 了。

说明：对于 mgfinal 来说，action 是交给 tomcat 管理的，service,dao 等是 ioc 容器管理的。所以当 action 被创建时，会递归注入其使用到的 service,dao 等属性。

3.3.提供 2 中容器

一种传统的 Map 容器，一种 Redis 容器。利用 redis 高效的读写性能来做容器，可以提供 mgfinal 的 bean 容器的性能，从而达到更大的体积。

默认 map 容器，通过 mgfinal.properties 配置文件中：

#mgfinal 容器类型 map,redis,默认是 map

mgfinal.ioc.type = [map](#)

切换到 redis 容器：

首先修改 mgfinal.ioc.type = redis，然后配置 redis 服务连接，默认是 0 号数据库，后面的 mybatis redis 查询缓存使用 1 号数据库，然后开启 redis 服务。

#redis 服务主机

mgfinal.ioc.redis.host = [localhost](#)

#redis 服务端口

mgfinal.ioc.redis.port = 6379

3.4.实例代码

```
@WebServlet("/test.do/*")
public class TestAction extends MGWorkServlet{
    @UseBean
    private DemoService demoService;
```

```
@ToBean
public class DemoService extends BaseService{
    @UseBean
    private DemoDao demoDao;
}
```

```
@ToBean
public class DemoDao extends BaseDaoImpl<Demo>
```

第四章.mybatis 的 ORM 拓展

4.1.集成初衷

本来说自己实现一个 orm 框架集成到 mgfinal 中的，但是这是一个大的工程。然后，看到当今的 mybatis 强大的生命力，就决定集成 mybatis 作为 mgfinal 的 orm 层了。

4.2.功能支持

基于 mybatis 的 orm 层，为了提高开发速度，提供了很多常用方法。

支持如下：

- ①基于动态 sql 的 CRUD 方法。
- ②对事务的支持。
- ③对 druid,c3p0 数据源的支持。
- ④基于 mybatis-ext 实现的对象的 CRUD 方法。
- ⑤基于 pagehelper 实现的分页方法。
- ⑥基于 mybatis-redis 实现 Mybatis 的查询二级缓存。

方法	备注
1.基于动态 sql 的 CRUD 方法	
T selectOne(String id,Object p)	查询一个对象 @param id 带 namespace 的 sql 的 id @param p sql 的参数 @return 结果
List<T> selectList(String id,Object p)	查询对象集合
List<Map<String,Object>> selectListMap(String id,Object p)	查询 map 集合
Map<String,Object> selectMap(String id,Object p)	查询 map 对象
Integer selectForInt(String id,Object p)	查询 int 列数据
String selectForString(String id,Object p)	查询 string 列数据
Object query(String id,Object p)	通用查询方法
int ddl(String id,Object p)	通用 DDL 方法
int insert(String id,Object p)	保存数据方法
int delete(String id,Object p)	删除数据方法
int update(String id,Object p)	更新数据方法

2.对事务的支持	
void start()	开启事务
void ddlTx(String id,Object p)	带事务 DDL 操作
void end()	提交事务
3.基于 mybatis-ext 实现的对象的 CRUD 方法	
void save(T obj,Class clazz)	保存对象
void saveList(List<T> obj,Class clazz)	保存 list 对象
void update(T obj,Class clazz)	更新对象
void delete(T obj,Class clazz)	删除对象
T one(String key,Object val,Class clazz)	通过 where key = vlaue 来查询对象
T one(T condition,Class clazz)	通过对象属性 and 条件来查询对象
List<T> list(T condition,Class clazz)	通过对象条件来查询对象 list
List<T> page(Class clazz,int pageNo,int size,T condition,String ...columns)	对象分页方法
int count(T condition,Class clazz)	通过对象条件查询记录数
4.基于 pagehelper 实现的分页方法	
PageInfo<T> selectPage(String id,Object p,HttpServletRequest req)	查询分页方法

4.3.功能试用

首先得配置 mybatis，加入 demo 的 mapper。

让 dao 层集成 BaseDaoImpl 类，就可以获得对 mybatis 的增强了。

4.3.1.基于动态 sql 的 CRUD 方法的使用

DemoService 的方法

```
public Demo show(){
    String id = "1";
    return (Demo)this.demoDao.selectOne("com.demo.vo.Demo.showId",id);
}
```

```
<mapper namespace="com.demo.vo.Demo">
    <select id="showId" resultType="com.demo.vo.Demo">
        select * from mg_user where id=#{id}
    </select>
</mapper>
```

4.3.2.事务支持调用

```
public void add2Demo() {
    Map<String, Object> p = new HashMap<String, Object>();
    p.put("pwd", new Date().toLocaleString());
    p.put("username", UUID.randomUUID().toString().substring(0,
10));
    //开启事务
    this.demoDao.start();
    //操作 1
    this.demoDao.ddlTx("com.demo.vo.Demo.addDemo", p);
    //操作 2
    this.demoDao.ddlTx("com.demo.vo.Demo.addDemo", p);
    //提交事务
    this.demoDao.end();
}
```

```
<insert id="addDemo">
    insert into mg_user(username,pwd) values(#{username},#{pwd})
</insert>
```

4.3.3.数据源支持

C3P0:

- ①添加 jar 包支持 c3p0-0.9.5.1.jar 和 mchange-commons-java-0.2.10.jar
- ②mybatis.xml 配置文件中，配置数据源如下

```
<dataSource type="com.mgfinal.core.mybatis.ds.C3P0DataSourceFactory">
    <property name="driverClass" value="${driver}" />
    <property name="jdbcUrl" value="${url}" />
    <property name="user" value="${username}" />
    <property name="password" value="${password}" />
    <property name="idleConnectionTestPeriod" value="60" />
    <property name="maxPoolSize" value="20" />
    <property name="maxIdleTime" value="600" />
    <property name="preferredTestQuery" value="SELECT 1" />
</dataSource>
```

Druid

- ①添加 jar 包支持 druid-1.0.5.jar
- ②mybatis.xml 配置文件中，配置数据源如下

```
<!-- druid 数据源 -->
<dataSource type="com.mgfinal.core.mybatis.ds.DruidDataSourceFactory">
    <!-- 基本属性 url、user、password -->
    <property name="driver" value="${driver}" />
    <property name="url" value="${url}" />
    <property name="username" value="${username}" />
    <property name="password" value="${password}" />
    <!-- 配置初始化大小、最小、最大 -->
    <property name="initialSize" value="1" />
    <property name="minIdle" value="1" />
    <property name="maxActive" value="20" />
    <!-- 配置获取连接等待超时的时间 -->
    <property name="maxWait" value="60000" />
    <!-- 配置间隔多久才进行一次检测，检测需要关闭的空闲连接，单位是毫秒 -->
    <property name="timeBetweenEvictionRunsMillis" value="60000" />
    <!-- 配置一个连接在池中最小生存的时间，单位是毫秒 -->
    <property name="minEvictableIdleTimeMillis" value="300000" />
    <property name="validationQuery" value="SELECT 'x'" />
    <property name="testWhileIdle" value="true" />
    <property name="testOnBorrow" value="false" />
    <property name="testOnReturn" value="false" />
    <!-- 打开 PSCache，并且指定每个连接上 PSCache 的大小 -->
    <property name="poolPreparedStatements" value="true" />
    <property name="maxPoolPreparedStatementPerConnectionSize" value="20" />
</dataSource>
```


4.3.4. 基于 mybatis-ext 实现的对象的 CRUD 方法

首先要加入 mybatis-ext 的 jar 包支持，然后配置实体对应表。

```
@TableName(name="mg_user")  
public class Demo implements Serializable
```

然后就可以使用 ext 的 curd 方法了。

```
public void save(){  
    Demo d = new Demo();  
    d.setPwd("123456");  
    d.setUsername("mybatis1234");  
    this.demoDao.save(d, Demo.class);  
}  
public void delete(){  
    Demo d = this.demoDao.one("id", 2107, Demo.class);  
    this.demoDao.delete(d, Demo.class);  
}  
public List<Demo> selectList(){  
    Demo d = new Demo();  
    d.setUsername("123");  
    return this.demoDao.list(d, Demo.class);  
}  
public void update(){  
    Demo d = this.demoDao.one("id", 2109, Demo.class);  
    d.setPwd("1234567890987654321");  
    this.demoDao.update(d, Demo.class);  
}  
public List<Demo> page(int pageNo, int size, Demo condition) {  
    return this.demoDao.page(Demo.class, pageNo, size, condition);  
}  
public int count() {  
    return this.demoDao.count(null, Demo.class);  
}
```

4.3.5.pagehelper 分页

加入 pagehelper 的 jar 包支持，然后在 mybatis.xml 配置文件中加入分页插件支持：

```
<!-- 分页插件 -->
<plugins>
  <plugin interceptor="com.github.pagehelper.PageHelper">
    <property name="dialect" value="mysql"/>
    <property name="pageSizeZero" value="true"/>
    <property name="reasonable" value="true"/>
  </plugin>
</plugins>
```

```
public PageInfo<Demo> selectPage(HttpServletRequest request) {
    return this.demoDao.selectPage("com.demo.vo.Demo.showAll", null, request);
}
```

```
<select id="showAll" resultType="com.demo.vo.Demo">
  select * from mg_user
</select>
```

下图是 demo 的截图：

id	姓名	密码
2151	cc	xx
2152	222	333
2153	1213	12321321
2154	a	b
2155	cc	xx
2156	222	333
2157	1213	12321321
2158	a	b
2159	cc	xx
2160	222	333

前一页	2	3	4	5	6	7	8	9	下一页
-----	-------------------	-------------------	-------------------	-------------------	----------	-------------------	-------------------	-------------------	-----

4.3.6. 基于 mybatis-redis 实现 Mybatis 的查询二级缓存

加入 Mybatis-redis 的 jar 包支持，然后开启 Mybatis 自身提供的缓存做为一级缓存，卡其 redis 服务，做为 mybatis 的查询 2 级缓存。

开启一级缓存，在 mybatis.xml 配置文件中加入配置：

```
<settings>
  <!-- 这个配置使全局的映射器启用或禁用缓存 -->
  <setting name="cacheEnabled" value="true" />
  <!-- 对于未知的 SQL 查询，允许返回不同的结果集以达到通用的效果 -->
  <setting name="multipleResultSetsEnabled" value="true" />
  <!-- 配置默认的执行器。SIMPLE 执行器没有什么特别之处。REUSE 执行器重用预处理语句。BATCH 执行器重用语句和批量更新 -->
  <setting name="defaultExecutorType" value="REUSE" />
  <!-- 全局启用或禁用延迟加载。当禁用时，所有关联对象都会即时加载。 -->
  <setting name="lazyLoadingEnabled" value="false" />
  <setting name="aggressiveLazyLoading" value="true" />
  <!-- <setting name="enhancementEnabled" value="true"/> -->
  <!-- 设置超时时间，它决定驱动等待一个数据库响应的的时间。 -->
  <setting name="defaultStatementTimeout" value="25000" />
</settings>
```

开启 redis 二级缓存：

在类路径下添加 redis.properties 配置文件，来连接 redis 数据库作为二级缓存容器。

```
#使用 redis 的第 2 个数据库来做 mybatis 查询缓存
host=localhost
port=6379
connectionTimeout=5000
soTimeout=5000
password=
#选择数据库，0 被 mgwork-ioc 占用
database=1
clientName=
```

注意，redis 的 0 数据库可能会被 mgfinal - ioc 占用。

然后再在需要使用缓存的 Mapper 中加上缓存开启配置：

```
<mapper namespace="com.demo.vo.Demo">
  <!-- redis 缓存 -->
  <cache type="org.mybatis.caches.redis.RedisCache" />
</mapper>
```

如此，redis 二级缓存就会生效，提升数据库查询效率了。