# SpringMvc相关

## 注解

1.@Service用于标注业务层组件

2.@Controller用于标注控制层组件（如struts中的action）

3.@Repository用于标注数据访问组件，即DAO组件

4.@Component泛指组件，当组件不好归类的时候，我们可以使用这个注解进行标注。

5.@Autowired可以对成员变量、方法和构造函数进行标注，来完成自动装配的工作，这里 必须明确：@Autowired是根据类型进行自动装配的，如果需要按名称进行装配， 则需要配合@Qualifier使用

6.@Qualifier("XXX") 中的 XX是 Bean 的名称，所以 @Autowired 和 @Qualifier 结合 使用时，自动注入的策略就从 byType 转变成 byName 了。

@Autowired 可以对[成员变量](http://baike.baidu.com/view/684821.htm" \t "http://baike.baidu.com/view/_blank)、方法以及[构造函数](http://baike.baidu.com/view/411124.htm" \t "http://baike.baidu.com/view/_blank)进行注释，而 @Qualifier 的标注对 象是成员变量、方法入参、构造函数入参。

示例

配合autowired使用：

@Autowired

@Qualifier("userServiceImpl")

publicI UserService userService;

1. @SuppressWarnings 可以标注在类、字段、方法、参数、构造方法，以及局部变量上。

作用：

告诉编译器忽略指定的警告，不用在编译完成后出现警告信息.

使用：

@SuppressWarnings(“”)

@SuppressWarnings({})

@SuppressWarnings(value={})

1. @ModelAttribute 被@ModelAttribute注释的方法会在此controller每个方法执行前被执行，因此对于一个controller映射多个URL的用法来说，要谨慎使用。

## 文件上传

==============================================================

<bean id="multipartResolver"

class="org.springframework.web.multipart.commons.CommonsMultipartResolver"/>

================

...Controller...{

//创建一个通用的多部分解析器.

CommonsMultipartResolver commonsMultipartResolver =

new CommonsMultipartResolver(request.getSession().getServletContext());

//设置编码

commonsMultipartResolver.setDefaultEncoding(”utf-8″);

//判断 request 是否有文件上传,即多部分请求...

if (commonsMultipartResolver.isMultipart(request)){

//转换成多部分request

MultipartHttpServletRequest multipartRequest =

commonsMultipartResolver.resolveMultipart(request);

// file 是指 文件上传标签的 name=值

// 根据 name 获取上传的文件...

MultipartFile file = multipartRequest.getFile("file");

//上传后记录的文件...

File imageFile = new File("fileName");

//上传...

file.transferTo(imageFile);

}

}

======

也可以在spring配置文件中配置

也可以在Spring的配置文件中添加 Xml代码 :

<bean id=”multipartResolver”

class=”org.springframework.web.multipart.commons.CommonsMultipartResolver”>

<property name=”defaultEncoding” value=”UTF-8″ />

</bean>

jsp code:

<form id=”fileUploadForm” method=”post” action=”XXX.do?method=fileUpload” enctype=”multipart/form-data”>

<tr>

<td>File:</td>

<td><input type=”file” name=”file”></td>

</tr>

<tr>

<td><input type=”hidden” name=”fileOwner” value=”1002″/></td>

<td><input type=”submit” value=”submit”></td>

</tr>

</form>

=======================================

spring mvc 上传文件（注解）

package net.hqit.action;

import java.io.File;

import java.io.IOException;

import javax.servlet.http.HttpServletRequest;

import org.springframework.stereotype.Controller;

import org.springframework.util.FileCopyUtils;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.multipart.MultipartFile;

@Controller

public class FileUploadController {

@RequestMapping(value="/upload.do",method=RequestMethod.POST)

public String upload(@RequestParam("imageFile") MultipartFile image,HttpServletRequest request) throws IOException

{

String path = request.getRealPath("/upload");

request.setAttribute("name",image.getOriginalFilename());

request.setAttribute("img","<img src='http://mn960mn.blog.163.com/blog/upload/"+image.getOriginalFilename()+"'></img>");

FileCopyUtils.copy(image.getBytes(),new File(path+"/"+image.getOriginalFilename()));

return "ok";

}

}

spring-servlet.xml:

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:p="http://www.springframework.org/schema/p"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<context:component-scan base-package="net.hqit.action"/>

<bean class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHandlerAdapter" />

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="viewClass"

value="org.springframework.web.servlet.view.JstlView"/>

<property name="prefix" value="/WEB-INF/view/"/>

<property name="suffix" value=".jsp"/>

</bean>

<bean id="multipartResolver"

class="org.springframework.web.multipart.commons.CommonsMultipartResolver"

p:defaultEncoding="utf-8" />

</beans>

别忘了加入commons里的fileupload跟io两个JAR包，这样就可以了。

==============================================================

## springmvc.xml配置

### 1、拦截器

1. **<mvc:interceptors>**
2. **<mvc:interceptor>**
3. <!-- 需拦截的地址 -->
4. <!-- 一级目录 -->
5. **<mvc:mapping** path="/\*.do" **/>**
6. **<mvc:mapping** path="/\*.ajax" **/>**
7. **<mvc:mapping** path="/\*.htm" **/>**
9. <!-- 二级目录 -->
10. **<mvc:mapping** path="/\*/\*.do" **/>**
11. **<mvc:mapping** path="/\*/\*.ajax" **/>**
12. **<mvc:mapping** path="/\*/\*.htm" **/>**
13. <!-- 需排除拦截的地址 -->
14. **<mvc:exclude-mapping** path="/login.htm"**/>**
15. **<bean** class="com.anxin.msapweb.web.interceptor.SecurityInterceptor" **/>**
16. **</mvc:interceptor>**
17. **</mvc:interceptors>**

## SpringMvc 中数据库操作JdbcTemplate方法例子：

### query查询

**private** JdbcTemplate template;

String sql,RowMapper

#### 1：无参数query(sql,BeanPropertyRowMapper(Class<?>))

1. **public** List<AreaDto> getAreaDtos() {
2. String sql = "select \* from t\_area ";
3. **return** (List<AreaDto>) template.query(sql, **new** BeanPropertyRowMapper(AreaDto.**class**));
4. }

#### 2：无参数 query(sql,AreaRowMapper())

1. **public** List<AreaDto> getAreaDtos1() {
2. String sql = "select \* from t\_area ";
3. **return** (List<AreaDto>) template.query(sql, **new** AreaRowMapper());
4. }

#### 3:无参数 query(sql,BeanPropertyRowMapper(Class<?>))

1. **public** Map<String, Object> getBuilds() {
2. String sql = "select buildid,buildname from t\_building ";
3. List<BuildingDto> list = **this**.template.query(sql, **new** BeanPropertyRowMapper(BuildingDto.**class**));
4. Map<String, Object> map = **new** HashMap<String, Object>();
5. **for** (BuildingDto build : list) {
6. map.put(String.valueOf(build.getBuildid()), build.getBuildname());
7. }
8. **return** map;
9. }

#### 4:1个参数query(sql,Object[]{传值},BeanPropertyRowMapper(Class<?>))

1. **public** Map<String, Object> getFloors(String build) {
2. String sql = "select floorid ,floorname from t\_floor where build\_id=?";
3. List<FloorDto> list = **this**.template.query(sql, **new** Object[] { build }, **new** BeanPropertyRowMapper(FloorDto.**class**));
4. Map<String, Object> map = **new** LinkedHashMap<String, Object>();
5. **for** (FloorDto floorDto : list) {
6. System.out.println(floorDto.getFloorid());
7. map.put(String.valueOf(floorDto.getFloorid()), floorDto.getFloorname());
8. }
9. **return** map;
10. }

#### 5：2个参数 query(sql,Object[]{值1,值2},

**BeanPropertyRowMapper(Class<?>))**

query(sql,Object[],BeanPropertyRowMapper(Class<?>))

1. **public** Map<String, Object> getAreas(String build, String floor) {
2. String sql = "select areaid ,area\_name from t\_area where build\_id=? and floor\_id=?";
3. List<AreaDto> list = **this**.template.query(sql, **new** Object[] { build, floor }, **new** BeanPropertyRowMapper(AreaDto.**class**));
4. Map<String, Object> map = **new** LinkedHashMap<String, Object>();
5. **for** (AreaDto areaDto : list) {
6. System.out.println(areaDto.getAreaid());
7. map.put(String.valueOf(areaDto.getAreaid()), areaDto.getArea\_name());
8. }
9. **return** map;
10. }

#### 6：3个参数 query(sql,Object[]{值1,值2,值3},

**BeanPropertyRowMapper(Class<?>))**

1. **public** List<StoreDto> getStoreDtos(**int** buildid, **int** floorid, **int** areaid) {
2. String sql = "select \* from t\_store where build\_id=? and floor\_id=? and area\_id=?";
3. **return** (List<StoreDto>) template.query(sql, **new** Object[] { buildid, floorid, areaid }, **new** BeanPropertyRowMapper(
4. StoreDto.**class**));
5. }

#### 7：1个参数queryForObject(sql,new Object[]{传值}},

BeanPropertyRowMapper(Class<?>))

1. **public** AreaDto getAreaDto(**int** id) {
2. String sql = "select \* from t\_area where areaid=?";
3. **return** template.queryForObject(sql, **new** Object[] { id }, **new** BeanPropertyRowMapper(AreaDto.**class**));
4. }

8：返回String类型queryForObject(sqk,Objeng[],String.class)

1. **public** String storeName(**int** storeId) {
2. String sql = "select storename from t\_store where id=?";
3. **return** template.queryForObject(sql, **new** Object[] { storeId }, String.**class**);
4. }

9:1个参数queryForObject(sql,Object[]{},BeanPropertyRowMapper(Class<?>))

1. **public** StoreDto getStoreById(**int** storeInt) {
2. String sql = "select \* from t\_store where id=?";
3. **return** template.queryForObject(sql, **new** Object[] { storeInt }, **new** BeanPropertyRowMapper(StoreDto.**class**));
4. }

10:

1. **public** **int** getCountStore() {
2. String sql = "select count(id) from t\_store";
3. **return** **this**.template.queryForInt(sql);
4. }

11:

1. **public** **void** deleteBuildById(**int** buildid) {
2. String sql = "delete from t\_store where id=?";
3. **this**.template.update(sql, buildid);
4. }

说明：

JdbcTemplate提供的方法很多如：queryForXXX，可以将查询结果以int、long、Object、List、Map来返回，这里有几个需要注意的：

queryForList方法

此方法返回List数据，但是注意List中的数据却是Map形式，类似：[{AREAID=1, AREA\_NAME=1楼报名咨询区}, {AREAID=2, AREA\_NAME=2楼教学区}, {AREAID=3, AREA\_NAME=3楼课外辅导区}]，其中字段名为key，字段值为value

如果需要查询数据库返回一个List<T>类型的对象该如何实现？

1. **public** List<AreaDto> getAreaDtos() {
2. String sql = "select \* from t\_area ";
3. System.out.println(template.queryForList(sql));
4. **return** (List<AreaDto>) template.query(sql, **new** BeanPropertyRowMapper(AreaDto.**class**));
5. }

# JAVA常用工具类

## StringUtils

### isNotEmpty 和isNotBlank的区别

1. public static boolean isEmpty(String str)

判断某字符串是否为空，为空的标准是str==null或str.length()==0

下面是StringUtils判断是否为空的示例：

StringUtils.isEmpty(null) = true

StringUtils.isEmpty("") = true

StringUtils.isEmpty(" ") = false //注意在StringUtils中空格作非空处理

StringUtils.isEmpty(" ") = false

StringUtils.isEmpty("bob") = false

StringUtils.isEmpty(" bob ") = false

2. public static boolean isNotEmpty(String str)

判断某字符串是否非空，等于!isEmpty(String str)

下面是示例：

StringUtils.isNotEmpty(null) = false

StringUtils.isNotEmpty("") = false

StringUtils.isNotEmpty(" ") = true

StringUtils.isNotEmpty(" ") = true

StringUtils.isNotEmpty("bob") = true

StringUtils.isNotEmpty(" bob ") = true

3. public static boolean isBlank(String str)

判断某字符串是否为空或长度为0或由空白符(whitespace)构成

下面是示例：

StringUtils.isBlank(null) = true

StringUtils.isBlank("") = true

StringUtils.isBlank(" ") = true

StringUtils.isBlank(" ") = true

StringUtils.isBlank("\t \n \f \r") = true //对于制表符、换行符、换页符和回车符StringUtils.isBlank()均识为空白符

StringUtils.isBlank("\b") = false //"\b"为单词边界符

StringUtils.isBlank("bob") = false

StringUtils.isBlank(" bob ") = false

4. public static boolean isNotBlank(String str)

判断某字符串是否不为空且长度不为0且不由空白符(whitespace)构成，等于!isBlank(String str)

下面是示例：

StringUtils.isNotBlank(null) = false

StringUtils.isNotBlank("") = false

StringUtils.isNotBlank(" ") = false

StringUtils.isNotBlank(" ") = false

StringUtils.isNotBlank("\t \n \f \r") = false

StringUtils.isNotBlank("\b") = true

StringUtils.isNotBlank("bob") = true

StringUtils.isNotBlank(" bob ") = true

# log4j日志管理

日志记录器（Logger）的行为是分等级的。如下表所示：  
分为OFF、FATAL、ERROR、WARN、INFO、DEBUG、TRACE、ALL或者您定义的级别。Log4j建议只使用四个级别，优先级从高到低分别是ERROR、WARN、INFO、DEBUG。通过在这里定义的级别，您可以控制到应用程序中相应级别的日志信息的开关。比如在这里定义了INFO级别，则应用程序中所有DEBUG级别的日志信息将不被打印出来。程序会打印高于或等于所设置级别的日志，设置的日志等级越高，打印出来的日志就越少。如果设置级别为INFO，则优先级高于等于INFO级别（如：INFO、WARN、ERROR）的日志信息将可以被输出,小于该级别的如DEBUG将不会被输出。

### log4j.properties配置的方式

#### 第一步：导入jar包，加入log4j-1.2.8.jar到lib下

#### 第二步：在CLASSPATH下建立log4j.properties.

(放在src下的话就不用配置 否则得去web.xml里面配置一个Listener.

<context-parm>

<para-name>log4jConfigLocation</param-name>

<param-value>/WEB-INF/log4j.properties</param-value>

</context-param>

<listener>

<listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>

</listener>

)

log4j.properties内容如下：

log4j.rootCatefory=INFO,stdout,R

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=[QC] %p [%t] %C.%M(%L) | %m%n

log4j.appender.R=org.apache.log4j.DailyRollingFileAppender

log4j.appender.R.File=D:\\Tomcat 5.5\\logs\\qc.log

log4j.appender.R.layout=org.apache.log4j.PatternLayout

log4j.appender.R.layout.ConversionPattern=%d-[TS] %p %t %c - %m%n

log4j.logger.com.neusoft=DEBUG

log4j.logger.com.opensymphony.oscache=ERROR

log4j.logger.net.sf.navigator=ERROR

log4j.logger.org.apache.commons=ERROR

log4j.logger.org.apache.struts=WARN

log4j.logger.org.displaytag=ERROR

log4j.logger.org.springframework=DEBUG

log4j.logger.com.ibatis.db=WARN

log4j.logger.org.apache.velocity=FATAL

log4j.logger.com.canoo.webtest=WARN

log4j.logger.org.hibernate.ps.PreparedStatementCache=WARN

log4j.logger.org.hibernate=DEBUG

log4j.logger.org.logicalcobwebs=WARN

log4j.rootCategory=INFO, stdout , R

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=[QC] %p [%t] %C.%M(%L) | %m%n

log4j.appender.R=org.apache.log4j.DailyRollingFileAppender

log4j.appender.R.File=D:\\Tomcat 5.5\\logs\\qc.log

log4j.appender.R.layout=org.apache.log4j.PatternLayout

log4j.appender.R.layout.ConversionPattern=%d-[TS] %p %t %c - %m%n

log4j.logger.com.neusoft=DEBUG

log4j.logger.com.opensymphony.oscache=ERROR

log4j.logger.net.sf.navigator=ERROR

log4j.logger.org.apache.commons=ERROR

log4j.logger.org.apache.struts=WARN

log4j.logger.org.displaytag=ERROR

log4j.logger.org.springframework=DEBUG

log4j.logger.com.ibatis.db=WARN

log4j.logger.org.apache.velocity=FATAL

log4j.logger.com.canoo.webtest=WARN

log4j.logger.org.hibernate.ps.PreparedStatementCache=WARN

log4j.logger.org.hibernate=DEBUG

log4j.logger.org.logicalcobwebs=WARN

#### 第三步：相应的修改其中的属性

1、logger：log4j.rootCategory = INFO,stdout,R

此句为将等级为INFO的日志信息输出到stdout和R这两个目的地，stdout和R的定义在下面的代码，可以任意起名。等级可分为OFF、FATAL、ERROR、WARN、INFO、DEBUG、ALL ,如果配置OFF则打不出任何信息，如果配置为INFO这样只显示INFO、WARN、ERROR的log信息，而DEBUG信息不会被显示。

2、Appender：log4j.appender.stout=org.apache.log4j.ConsoleAppender

此句为定义名为studout的输出端是哪种类型，可以是：

org.apache.log4j.ConsoleAppender（控制台），

org.apache.log4j.FileAppender（文件），

org.apache.log4j.DailyRollingFileAppender（每天产生一个日志文件），

org.apache.log4j.RollingFileAppender（文件大小到达执行尺寸产生一个新的文件），

org.apache,log4j.WriterAppender（将日志信息以流格式发送到任意指定的地方）

3、Layout:log4j.appender.stdout.layout=org.apache.log4.PatternLayout

此句为定义名为stdout的输出端的layout是哪种类型的，可以是：

org.apache.log4.HTMLLayout（以HTML表格形式布局），

org.apache.log4j.PatternLayout（可以灵活的指定布局模式），

org.apache.log4j.SimpleLayout（包含日志信息的级别和信息字符串），

org.apache.log4j.TTCCLayout（包含日志生产的时间、线程、类别等等信息）

4、格式化日志信息：log4j.appender.stout.layout.ConversionPattern=[QC]%P[%t]%C.%M($L) | %m%n

如果使用pattern布局就要指定的打印信息的具体格式ConversionPattern,打印参数如下：

%m输出代码中指定的消息

%p输出优先级，即DEBUG、INFO、WARN、ERROR、FATAL

%r输出自应用启动到输出该log信息耗费的毫秒数

%c输出所属的类目，通常就是所在的类的全名

%t输出产生该资质时间的线程名

%n输出一个回车换行符，Windows平台为”rn”,Unix平台为”n”

%d输出日志时间点的日期或时间，默认格式为ISO8601，也可以在其后制定格式，比如：%d{yyyy MM dd HH:mm:ss,SSS},输出类似：2017年10月18日11:11:11,666

%l 输出日志事件的发生位置，包括类目名、发生的线程，以及代码中的行数。

[QC] 是log信息的开头，可以为任意字符，一般为项目简称。

输出的信息

[TS] DEBUG [main] AbstractBeanFactory.getBean(189) | Returning cached instance of singleton bean 'MyAutoProxy'

5、 log4j.appender.R.File=D:\\Tomcat 5.5\\logs\\qc.log

此句为定义名为R的输出端的文件名为D:\\Tomcat 5.5\\logs\\qc.log 可自行修改

6、log4j.logger.com. neusoft =DEBUG

指定com.neusoft包下的所有类的等级为DEBUG

可以吧com.neusoft改为自己项目所有的包名

7、 log4j.logger.com.opensymphony.oscache=ERROR

log4j.logger.net.sf.navigator=ERROR

这两句是把这两个包下出现的错误的等级设为ERROR，如果项目中没有配置EHCache，则不需要这两句。

8、 log4j.logger.org.apache.commons=ERROR

log4j.logger.org.apache.struts=WARN

这两句是struts的包。

9、 log4j.logger.org.displaytag=ERROR

这句是displaytag的包。（QC问题列表页面所用）

10、log4j.logger.org.springframework=DEBUG

此句为Spring的包。

11、log4j.logger.org.hibernate.ps.PreparedStatementCache=WARN

log4j.logger.org.hibernate=DEBUG

此两句是hibernate的包。

#### 第四步：在要输出日志的类中加入相关语句：

定于属性：protected final Log log = LogFactory.getLog(getClass());

在相应的方法中：if(log.isDebugEnabled) {

log.debug(“System ...”);

}

### DailyRollingFileAppender:

DailyRollingFileAppender是日志记录软件包Log4J中的一个Appender，它能够按一定的频度滚动日志记录文件。

我们可以按下面的方式配置DailyRollingFileAppender：

log4j.rootCategory=INFO,file  
log4j.appender.file=org.apache.log4j.DailyRollingFileAppender  
log4j.appender.file.DatePattern='.'yyyy-MM-dd  
log4j.appender.file.File=run.log  
log4j.appender.file.Append=true  
log4j.appender.file.Threshold=INFO  
log4j.appender.file.layout=org.apache.log4j.PatternLayout  
log4j.appender.file.layout.ConversionPattern=%c %x - %m%n

在DailyRollingFileAppender中可以指定monthly(每月)、 weekly(每周)、daily(每天)、half-daily(每半天)、hourly(每小时)和minutely(每分钟)六个频度，这是通过为 DatePattern选项赋予不同的值来完成的。DatePattern选项的有效值为：

'.'yyyy-MM,对应monthly(每月)

'.'yyyy-ww,对应weekly(每周)

'.'yyyy-MM-dd,对应daily(每天)

'.'yyyy-MM-dd-a,对应half-daily(每半天)

'.'yyyy-MM-dd-HH,对应hourly(每小时)

'.'yyyy-MM-dd-HH-mm,对应minutely(每分钟)

DatePattern中不用处理的文字要放到单引号(')中，如上面的(.)。如果您对此有疑问可以查阅SimpleDateFormat的文档。DailyRollingFileAppender中使用这个类来处理DatePattern。

DatePattern格式化之后的文本作为文件名字的后缀。DailyRollingFileAppender不支持格式化之后的文本作为文件名字的前缀。

DailyRollingFileAppender在每一个日志事件(LoggingEvent)附加(append)之前检查是否需要附加。也就是说如果在一个滚动区间中没有任何日志记录，那么这个区间的日志记录文件就不会形成。

### **RollingFileAppender**

下面是RollingFileAppender的一个Log4j配置样例：

log4j.appender.R=org.apache.log4j.RollingFileAppender

log4j.appender.R.Threshold=DEBUG

log4j.appender.R.File=test.log

log4j.appender.R.layout=org.apache.log4j.PatternLayout

log4j.appender.R.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} [%5p] - %c -%F(%L) -%m%n

log4j.appender.R.MaxFileSize=20MB

log4j.appender.R.MaxBackupIndex=10

RollingFileAppender使用MaxFileSize设置一个日志文件的最大大小，当产生多个日志时，会在日志名称后面加上".1"、".2"、……这样的后缀，我们可以看到RollingFileAppender有个属性MaxBackupIndex，这个属性通过限制日志文件名后缀".n"中的n大小来限制日志数量，比如上面MaxBackupIndex=10，其实最大日志数量为11。我们知道这个有这个限制是很必要的，当我们的程序在服务器上运行时，随着时间的迁移，日志会越来越多，如果对日志数量没有限制，日志大小会越来越大，最后甚至占满整个硬盘。

# 常用开发框架

## Spring MVC & Spring Boot

Spring MVC应该最常用的。 Spring Web MVC框架（ 简称Spring MVC）是一个丰富的模型视图控制Web框架。使用 DispatcherServlet发布请求处理，他使用可配置的处理程序映射，视图解析和主题解决方案。Spring MVC 用户似乎都商量好的，大约40%的开发都使用Spring。

Spring Boot让开发团队在创建基于JVM的应用及方案更加有效率。至于Web框架，Spring Boot 为 Spring MVC 提供自动设置功能，可以适用于大多数程序。如果你想快速开始Web应用开发，推荐使用。

## Grails

　　知名Java虚拟机Web应用程序开源框架。Grails是一个开源、全栈JVM Web开发框架。吸取Groovy语言精华，对于Java虚拟机来说是一个敏捷和动态语言，为开发者提供一个高效开发经验。

Spring

Spring Framework 是一个开源的Java/Java EE全功能栈（full-stack）的应用程序框架，以Apache许可证形式发布，也有.NET平台上的移植版本。该框架基于 Expert One-on-One Java EE Design and Development（ISBN 0-7645-4385-7）一书中的代码.

## JFinal 快速开发平台 EOVA

基于JFinal开发，首创JFinal 快速开发平台，JFinal最佳实践！ 大家好我是Java的表弟Eova，中文名:伊哇 :) 我可以快速完成开发任务，信手拈来 我会10大控件,4大组件,N种业务模型 常见的增删改查、条件查询、数据校验、前后置业务处理、批量操作、权限管理....

EOVA

## JavaEE应用基础平台 AOS

AOS平台简介 AOS应用基础平台基于JavaEE技术体系，以“标准功能可复用、通用模块可配置、行业需求快速开发、异构系统无缝集成”为目标，为软件开发团队提供高 效可控、随需应变、快速实现业务需求的全栈式技术解决方案。帮助企业落实IT策略、屏蔽技术壁垒.

AOS

## Spring Scala

Spring Scala 项目的目的是为了简化在 Scala 应用中使用 Spring 框架。我们相信很多 Spring 用户想尝试 Scala，但并不像放弃他们在 Spring 框架上的积累，这个项目就是为这些人准备的。 目前该项目还在进一步完善开发中。

## SpringSide

SpringSide是以Spring Framework为核心的，Pragmatic风格的JavaEE应用参考示例，是JavaEE世界中的主流技术选型，最佳实践的总结与演示。 Quickstart: 一个迷你的Todo-List应用，演示CRUD Web+Restful API。 Showcase: 五花八门的JavaEE技术大派对。

SpringSide

## Java应用框架 Nutz

首先，它是开源的，商用也是完全免费的。详尽的文档，众多线上项目 不吹嘘概念,代码严谨可靠，近千个testcase,坚如磐石。您使用的nutz.jar,也是各位提交者在各种生产环境上使用的 nutz.jar ^\_^ 一个 jar 包就包含了 Mvc,Ioc,Dao,Aop，不强制依赖第三方 .

## 基于Flex和J2EE的信息管理系统基础框架 Pomer

Pomer是一 个开源的企业信息管理系统基础框架，它提供了一系列企业信息管理系统的基础功能，包括企业组织架构管理，角色权限管理，菜单管理，数据权限控制等功能。 Pomer提供了简洁高效的Flex应用程序开发框架，丰富实用的Flex控件以及常见的Flex应用模式.

Pomer

## 企业应用开发平台 Spring Roo

Spring Roo 是针对 Java™ 技术的一个轻量级生产力工具，可快速便捷地开发基于 Spring 的应用程序。使用 Spring Roo 创建的应用程序遵循 Spring 最佳实践，且基于 JPA、Bean Validation (JSR-303) 和 Dependency Injection (JSR-330) 等标准。

Spring Roo

## JBoss Seam

JBoss Seam是“Java EE 5.0的一个轻量级的框架”。它在Java EE 5.0框架的上层，为所有的在企业Web应用中的组件提供了一个统一的、易于理解的编程模型。它同样使基于状态的应用和业务流程驱动的应用的开发易如反 掌。

JBoss Seam

## Web框架 Jdon Framework

Jdon Framework（简称JF）是一套适合开发中小型J2EE/JavaEE应用系统的轻量Web框架(Lightweight Java Web Framework)。是国人独立开发的中国人自己的框架产品，2005年入选全球SUN公司网站java.net正式企业应用目录。经过多年发展和众多 用户的使用及其完善.

Jdon Framework

## J2EE应用开发框架 TinyFramework

值得拥有的企业级j2ee应用开发框架套件，专业团队开放，完整的生态体系，活跃的社区氛围，无限的水平扩展能力，7\*24不间断运维能力。 我心目中理想的开源框架 她应该是小的、简单的，满足Simple Is Beautiful 她应该是成长性好的，随着不断的扩展.

## J2EE应用开发框架 Beetle

甲壳虫，一个功能强大而简单易用的J2EE应用开发框架。它涵盖了J2EE体系结构的表示层、业务层和持久层，为构建一个可靠、高性能、可扩展、灵活缩放的高质量企业应用系统提供了一套理想的解决方案。

Beetle

## 商业】J2EE快速开发平台 F1JEE

F1JEE-中国自主知识产权J2EE快速开发平台（非开源） F1JEE业务基础软件平台基于业界工业标准J2EE实现，包括集成应用平台、开发体系两个部分。从技术角度分析，F1JEE业务基础软件平台为复杂应用 软件系统的开发提供了一个基本框架，并有与之相应的、方便易用...

## j2ee应用快速开发手架 thinker

thinker(像一个思想者一样去写代码) 介绍 工作中的积累，搭建的j2ee应用快速开发手架，重点在代码重构，一些常用功能的演示，工具，插件等的整理.

## 敏捷开发框架 ajf

ajf是自主研发类似ROR的新一代java快速开发平台 借鉴了struts webwork spring hibernate ibatis等开源组件优点 实现了mvc,orm,sqlmap,cache,pool,jdbcutil,commutil等组件 2004年6月份开发测试维护至今,简洁高效稳健,将向分布式领域积极拓展 发扬山寨精神.

## Warp Framework

Warp framework 是基于Google Guice的轻量级Web开发框架 Warp框架充分利用了JDK5.0的Annotation和泛型机制，并且基于Google Guice这个IoC框架，提供了full-stack的Web开发设施，他主要包含了四个部分： warp-persist框架：封装Hibernate和JPA，提供事务管理.

## J2EE框架 Jxva

Jxva Framework是一个开放源代码的、基于热插拨功能扩展的、超轻量级的、快速开发的、不依赖任何第三方框架的JEE框架及企业资源整合集成 平台。利用 Web方式极大的方便了用户对于系统功能的使用；通过合理的框架组织及泛型与注解的采用.

Jxva

## 企业应用快速开发平台 jeefuseMDA

jeefuseMDA 是一个基于MDA的javaEE企业应用快速开发平台。 它整合一些流行的开源框架如Ibatis、 Hibernate、 Spring Framework、Spring Security、struts2、ehcache、lucene、solr等，构建为稳定通用的基础开发框架。定义或导入uml模型,并可从UML模 型.

## J2EE OpenCore

Open Core is a middleware abstraction layer allowing subsystems and applications to seamlessly run directly on top of Java Virtual Machine or inside of various J2EE application servers.

J2EE OpenCore

## 基于 POJO 的Java框架 ROMA Framework

Roma是一个基于 POJO 的Java框架，是唯一的“meta”（元）框架。使用Roma，你可以完全从你使用的技术中解脱出来。它是如何工作的？Roma提供称作 “Aspects”的非常普通的行为接口。Aspects封装了最普通的用例。通过Aspects使用工具和框架.

更多ROMA Framework信息

## 动态模块框架 Impala

Impala是一个Java企业应用程序开发的动态模块框架，它基于Spring框架。集中于简易性和生产力，Impala通过使用Spring和相关技术，从根本上改变了应用程序的开发形式。

## openFrame

openFrame 是一个开源的 J2EE 应用框架。

openFrame

## J2EE快速开发框架 JDFrame

JDFrame 采用基于J2EE的B/S/S三层体系架构、基于MVC设计模式、集成业界常用标签库、集成权限体系管理功能、集成待办事宜、系统公告等功能； 为达到最佳性能开发环境建议采用Jdk1.6及以上版本,采用主流中间件及数据库，支持任何J2EE 1.3规范的中间件.

## Java集成框架 iBeans

iBeans， 一个新的集成框架，类似于Spring templates的概念。iBeans与Tcat Server一起发行，不过它也可以部署在Apache Tomcat和Mule ESB上。 iBeans的设计目的是提供一个简单的、一致的、可重用的方式运用各种协议进行通讯。许多框架都把注意力集中到了底层.

## J2EE开发框架 JFoxSOAF

JFoxSOAF 是一个可靠的体系架构，用以更简单、更快速的开发 J2EE 应用程序。

JFoxSOAF

## 基于J2EE架构的中间件平台 Netone

NETONE中间件平台是一套完整且全面的、基于J2EE架构的中间件平台。它能帮助IT公司或者大型企业的IT部门快速搭建复杂且全面的企业级应用软件平台。基于NETONE中间件平台的软件开发是可视化且无编码的，可极大的降低开发人员的技术门槛.

## 企业级快速开发平台 JXADF

基 于OSGi的企业级快速开发平台——JXADF 1、基于OSGi、J2EE 2、插件集市（http://osgi.jxtech.net）中有丰富的插件。 3、前端代码（界面、CSS、JS、图片等）、后端代码（JAVA业务逻辑、SQL脚本【只需要提供一种数据库脚本】。

JXADF

## J2EE 框架 FleXive

FleXive 是一个基于EJB3标准并且整合了JSF组件库，灵活且扩展性高的java EE 5的程序包（library）。它重点关注于企业级web应用，并全面支持通过jsf来显示和操纵企业规模级的数据存储和查询。 你可以在现有的java EE应用中添加FleXive.

FleXive

## 应用开发框架 Nuxeo5

Nuxeo5 是一个用于ECM(Enterprise Content Management)应用程序开发的开源框架。基于组件和面向服务的架构使得它真正易于定制和扩展。Nuxeo5构建流行的Java5与Java EE5技术包括：Apache Jackrabbit ,JBoss应用服务器,JBoss Seam,jBPM,JBoss Rules,JSF,EJB3...

Nuxeo5

## EJB3 容器 EasyBeans

EasyBeans 是一个轻量级的实现了EJB 3.0 规范的容器。

EasyBeans

## J2EE应用程序框架 Verge

Verge框架是一个开源J2EE应用程序框架，完全采用Java开发只使用到标准的J2EE APIs.这个开源项目的目的为开发企业应用程序提供一个非常灵活和可扩展的框架.

## Java体系结构框架 Arch4J

Arch4J是一个Java体系结构框架，提供了一个服务基础结构来支持企业业务开发。包括一套分层的模块化服务提供者，可以进行数据访问，消息传递和日志，等等。

Arch4J

## spring-all

本 人比较烦使用Spring时加入一大推Spring的jar文件。 于是将Spring的所有jar包打包成一个文件。 以3.2.5为例分别为： spring-3.2.5.jar （bin only） spring-src-2.3.5.jar （source only） spring-with-src-3.2.5.jar（bin+source） .

更多spring-all信息

## J2EE 全栈框架 Ameba

Ameba 一个构建在J2EE标准之上的全栈框架，以后可能包含更多，从售前竞标到发布部署整条流水。简单、易用、务实、流程化。只为解决问题，而不创造技术。 Ameba采用了jersey、logbak（groovy配置）、AKKA、Ebean等等第三方类库，所以注定Ameba的身材很感人.

## 媒体对象服务组件 jMOS

jMOS 是利用Java来实现MOS(Media Object Servers)通信协议的开源组件。它使用快速的XML处理引擎(如：SAX,JAXP或StAX)来与新闻编辑室控制系统(NCS) 和多媒体对象服务器(MOS:如视频服务器，音频服务器等)相互协作。

## 应用开发框架 Naked Objects

Naked Objects是一种架构模式，也是一个以领域对象为中心角色的应用开发框架。Naked Object应用中的领域对象是用户界面的核心，同时也是开发活动的焦点。Naked Objects最近发布了3.0版，它支持Java 1.5、注入（injection）、一个可选UI、Hibernate对象存储.

Naked Objects

## 可测试的集成架构 Savara

SAVARA 是由JBoss/RedHat 开发的一个技术解决方案。 这个项目可以为企业和解决方案架构创建开发工具，独特之处就是基于一个新的技术—“可测性整合框架”。 Eclipse Update Site URL http://downloads.jboss.org/savara/eclipse/1.0.x.

Savara

## GreatTurbo Enterprise Server

GreatTurbo Enterprise Server 10.5（简称GTES10.5）是拓林思公司的旗舰产品企业级服务器操作系统的最新版本。它针对大型企业计算环境进行了特别优化，为企业提供了适合大型 商业数据库和大型商业应用软件运行的操作系统平台解决方案。GTES10.5支持24×7.

## Expresso

Expresso is the most popular, powerful, enterprise-strength J2EE architectural persistence framework.

## RAD 快速应用开发环境 EntireJ

EntireJ 是一个 RAD 快速应用开发环境，用于构建高度可伸缩的企业级应用，基于 RCP/RAP、JavaFX 平台以及 Android 和 iOS 移动平台。使用强大的声明式特性。

## Polyforms

Polyforms 项目的目标是剔除用于绑定数据访问对象和底层持久化API的重复代码。该框架采用的办法是自动实现开发人员定义的Repository接口上的服务方法。 接口中定义的方法默认通过命名规则连接到数据库操作上。例如，项目wiki上讨论了如何为User实体对象实现...

## Spring Roo Addon 套件 gvNIX

gvNIX 是 Spring Roo Addon Suite，主要基于企业知识和企业标准来构建 Java 应用。

## Play

Play使用Java 和 Scala创建应用。基于Akka，为资源提供优化。他还提供一个轻量级的、无边界的、对用户友好的开发者社区。

## Vaadin

Vaadin是一个专为当前Web应用设计的Java框架，界面美观，性能高，开发者和用户都会喜欢。基于Google Web Toolkit (GWT)，而且，Vaadin建的App和UI都使用Java编写。

## Dropwizard

Dropwizard是一个轻量级、的Java开发框架。他对开发友好、高性能、还有RESTful Web服务。

## Vert.x

Vert.x，轻量级、高性能JVM应用开发平台，移动、Web、企业程序都可用。他是一个服务器框架，提供基于事件的编程模型异步AP。在2014年得过奖 "Most Innovative Java Technology（最具创新性的Java技术奖）"

## Spark

受Sinatra启发而创建的web框架，在Java 8中有不小表现。他并不是致力于取代Sinatra.

等等

博客：http://my.oschina.net/leejayblog/blog 博客：http://abrahamleejay.360doc.com