atitit.软件与sql与数据库设计模式大总结attialx总结v5 qc28.docx

[1.1. 版本历史 4](#_Toc21865)

[2. 软件设计模式起源于建筑领域，而建筑工程中设计模式的数量（253个） 4](#_Toc2505)

[2.1. 设计模式解决什么问题：提升开发效率，提升可读性，提升扩展性，提升兼容性，提升xxx性 4](#_Toc32190)

[2.2. 注意产品与项目的适用性，但有些模式不太适应于有更换数据库的需求情况下 4](#_Toc28754)

[3. 具体的设计模式 5](#_Toc26672)

[3.1. Gof 23个经典设计模式(截止1994年） 5](#_Toc8441)

[3.2. 软件新发展的设计模式(1995-2016年）attilax总结总和大约100个.. 5](#_Toc10216)

[4. 软件新发展的设计模式(1995-2016年）attilax总和大约100个.. 5](#_Toc2027)

[4.1. Sql优先模式 sql是dsl，是4gl，代表更高的生产力 5](#_Toc12127)

[4.2. 1=1 设计模式 条件表达式归一化DBTAG 5](#_Toc12474)

[4.3. table mapping表映射法 5](#_Toc6901)

[4.4. Rest+sql传递参数模式 5](#_Toc20751)

[4.5. 后端弱化为web 版本的jdbc驱动，提供rest接口交互 6](#_Toc20816)

[4.6. Er模型 DBTAG 6](#_Toc9747)

[4.7. 数据流程图(DFD) DBTAG 6](#_Toc12197)

[4.8. 外模式（External Schema) DBTAG 6](#_Toc8720)

[4.9. 内模式 DBTAG 6](#_Toc26969)

[4.10. 数据驱动 DBTAG 6](#_Toc5137)

[4.11. Unique索引设计模式 DBTAG 6](#_Toc15774)

[4.12. Trigger模式 DBTAG 7](#_Toc396)

[4.13. 视图模式 视图是的代码更加简洁，可读性更好 DBTAG 7](#_Toc31474)

[4.14. Oo模式，对于数据库sql这样的语言来说，适用oo模式可以大力提升存储过程可读性 7](#_Toc27231)

[4.15.](#_Toc18352) **[1nf 2nf 3NF设计模式](#_Toc18352)** [DBTAG 7](#_Toc18352)

[4.16.](#_Toc12704) **[鲍依斯-科得范式（BCNF是的改进形式）](#_Toc12704)** [DBTAG 7](#_Toc12704)

[4.17. Api映射法取代 warp法。 7](#_Toc8213)

[4.18. async异步模式 DBTAG 7](#_Toc31255)

[4.19. AOP 7](#_Toc15709)

[4.20. Abstract 优对于 interface 7](#_Toc31577)

[4.21. Ast模式 7](#_Toc30443)

[4.22. Auto type convert自动进行类型转换 8](#_Toc15576)

[4.23. biz业务化模块 8](#_Toc30105)

[4.24. 适当适用存储过程，提升代码简洁与可读性 dbtag 8](#_Toc20075)

[4.25. Compiler as a Service 8](#_Toc4772)

[4.26.](#_Toc4633) **[CBD（核心Core+行为Behavior+驱动Driver）](#_Toc4633)**[架构模式 8](#_Toc4633)

[4.27. Chain mode 8](#_Toc18126)

[4.28. Confirm模式(解除页面跳转的问题) 8](#_Toc2342)

[4.29. closure闭包模式 8](#_Toc29606)

[4.30.](#_Toc26300) **[components组件化](#_Toc26300)** [8](#_Toc26300)

[4.31.](#_Toc24611) **[concurr并发模式(解除性能问题](#_Toc24611)** [8](#_Toc24611)

[4.32. DSL 8](#_Toc31909)

[4.33. xxp vs AJAX 8](#_Toc19179)

[4.34. DSLparser 8](#_Toc31462)

[4.35. eventDriver事件驱动模式 9](#_Toc18430)

[4.36. Code behind 9](#_Toc329)

[4.37. Cfg2ui (attilax 创建 9](#_Toc3236)

[4.38. Biz\_index业务索引模式(attilax 创建,解 9](#_Toc14863)

[4.39. dispater分发器模式 9](#_Toc9159)

[4.40. declare声明式编程模式. 9](#_Toc12134)

[4.41. Dsl模式 9](#_Toc5667)

[4.42. filter过滤器模式 9](#_Toc22869)

[4.43. Fp模式 9](#_Toc11504)

[4.44. Engeen+script引擎+脚本模式(托管) 9](#_Toc4155)

[4.45.](#_Toc27383) **[frmwk框架化](#_Toc27383)** [9](#_Toc27383)

[4.46. 事件驱动 9](#_Toc12310)

[4.47. 事件分发 9](#_Toc30514)

[4.48. generic泛型模式 9](#_Toc23336)

[4.49. -------------------h----n 9](#_Toc19801)

[4.50. Html5 ui 9](#_Toc32445)

[4.51. Hybrid化 9](#_Toc27673)

[4.52.](#_Toc4891) **[immediately](#_Toc4891)**[即时更新（热部署）快速迭代 持续交付 9](#_Toc4891)

[4.53. Ioc/di 10](#_Toc13767)

[4.54. Lazy load延迟初始化 cfg 10](#_Toc6108)

[4.55. Linq 10](#_Toc8130)

[4.56. meta programe(anno,.net attr) 10](#_Toc6220)

[4.57. microcore微内核 10](#_Toc28142)

[4.58. msg消息化 10](#_Toc8060)

[4.59. Mvc mvp mvvm 10](#_Toc25316)

[4.60. ----o-------t 10](#_Toc554)

[4.61. Orm 11](#_Toc21197)

[4.62. PageController 跟FrontController设计模式 11](#_Toc21681)

[4.63. Pipe 11](#_Toc13837)

[4.64. Pipe 管道模式、流式，链式 11](#_Toc21908)

[4.65. platform 11](#_Toc536)

[4.66. plugin插件模式 11](#_Toc6513)

[4.67. Reflection 11](#_Toc558)

[4.68. refresh刷新模式(解决缓存and静态的变量的值及时修改跟个读取问题) 11](#_Toc27469)

[4.69. refresh刷新按钮模式...解决若干页面不料条件..and 分页.. 11](#_Toc30426)

[4.70. propCopy数据转换器..propCopy 11](#_Toc11429)

[4.71. pool池模式 11](#_Toc14938)

[4.72. Retry(attilax 创建,提升稳定性) 11](#_Toc6826)

[4.73. Stream 12](#_Toc22879)

[4.74. Self contain自包含，容器化隔离。。 12](#_Toc19073)

[4.75. seri串口模式(attilax 创建,解除接口麻烦的) 12](#_Toc19147)

[4.76. spa ajax化 12](#_Toc31886)

[4.77. Try catch 12](#_Toc1617)

[4.78. 决缓存不及时的问题) 12](#_Toc10001)

[4.79. Theardlocal模式 12](#_Toc31014)

[4.80. trigger触发器模式(attilax 创建) 12](#_Toc23192)

[4.81. SOA 12](#_Toc4385)

[4.82. single单例文件夹模式(解决模块复制问题,提取问题) 12](#_Toc30485)

[4.83. Timer 模式...解决同步问题两个控件 12](#_Toc6655)

[4.84. Theardlocal模式 13](#_Toc7835)

[4.85. ----u----z 13](#_Toc17832)

[4.86. Uapi dbtag 13](#_Toc31503)

[4.87. Vm模式 13](#_Toc16288)

[4.88. visual 13](#_Toc22095)

[4.89. webService/HttpService 13](#_Toc27919)

[4.90. workflow工作流 13](#_Toc16538)

[4.91. Winform模式 native 13](#_Toc17926)

[4.92. Webform模式 13](#_Toc8006)

[4.93. Web page设计模式 13](#_Toc23020)

[4.94. Weak/Soft references 13](#_Toc3581)

[4.95. Promise设计模式 13](#_Toc24621)

[4.96. Yield模式 13](#_Toc3873)

[4.97. 管道模式，链式模式 13](#_Toc32685)

[4.98. Plugin设计模式 13](#_Toc11264)

[5. 反模式 13](#_Toc2702)

[5.1. Reset web 13](#_Toc27651)

[6. 参考资料 13](#_Toc6458)

软件的发展日新月异。。设计模式也是一直在发展。Gof 23个经典设计模式发表与1994年，也只记录了最常用的设计模式23个，但是，这个最常用的标准数量与内容因人而异和具体项目而不同，在当时1994年，设计模式的数量加上不常用的，按照20 80法则，应该总数也在100个左右。。

那么1995年以后，当然一直也有新的设计模式出现，只是没有经典书籍来总结了

软件新发展的设计模式(1995-2016年）attilax大概统计了下自身用到的设计模式大约80个

具体80个新总结的设计模式如下表所示。。

Mvc是不是设计模式，当然是设计模式来。。Struts，jsf，h5这类才是框架，才是实现mvc模式的代码实现。。..

## 版本历史

Frm atitit.软件设计模式大总结attialx总结v3 pa5

凡是打了dbtag标签的设计模式都是适合于数据库以及sql编程的设计模式。。

# 软件设计模式起源于建筑领域，而建筑工程中设计模式的数量（253个）

## 设计模式解决什么问题：提升开发效率，提升可读性，提升扩展性，提升兼容性，提升xxx性

注意：提升可读性往往与性能成反比，一般情况下，尽可能可读性优先。。面向人类编程优先于面向机器编程嘛。。

## 注意产品与项目的适用性，但有些模式不太适应于有更换数据库的需求情况下

一般来说，本设计模式很适合于项目模式。只有在存在更换数据库的需求下，有些模式才不那么适合。

# 具体的设计模式

## Gof 23个经典设计模式(截止1994年）

Gof只总结了最常用的23个经典模式。当然还有很多次常用的设计模式没有总结进去，还有数据库与sql方面的设计模式也是么有考虑的，他们貌似只考虑了编程语言中的通用性的设计模式

## 软件新发展的设计模式(1995-2016年）attilax总结总和大约100个..

# 软件新发展的设计模式(1995-2016年）attilax总和大约100个..

## Sql优先模式 sql是dsl，是4gl，代表更高的生产力

Sql的可读性，大大优先于同样功能的默认java c# 等3gl语言。。当然3gl也可以提供linq stream等函数式api提升很大部分的可读性。。

目测貌似简单sql可以达到sql可读性的90%..

但对于复杂的sql（有join，有分组聚合），相对来说 可读性貌似只有sql的50%了。。

## 1=1 设计模式 条件表达式归一化DBTAG

## table mapping表映射法

每个数据库的表格都是现实生活中的一张纸质表格

## Rest+sql传递参数模式

直接通过rest传递sql语句。。

可以大力提升可读性与开发效率。但是会对安全性造成一定的影响。。如果内部系统使用此模式可以无需关心安全性，大力提升了开发效率

外部系统或许可以增强安全性。。或者只传递一个sqlid，这样就很好的安全性了。

## 后端弱化为web 版本的jdbc驱动，提供rest接口交互

此时的业务逻辑基本可以完全使用前段语言js来熟悉，带来很大的灵活性与开发效率。。

但可能会带来一点的安全隐患。。注意使用场合。。

## Er模型 DBTAG

## 数据流程图(DFD) DBTAG

## 外模式（External Schema) DBTAG

外模式也称子模式(Subschema)或用户模式，它是数据库用户（包括应用程序员和最终用户）能看见和使用的局部数据的逻辑结构和特征描述，

子模式

DBMS 提供子模式描述语言（子模式DDL）来严格地定义子模式。

## 内模式 DBTAG

内模式又称存储模式，对应于物理级，它是数据库中全体数据的内部表示或底层描述

内模式由内模式描述语言来描述、定义所有内部记录类型、索引和文件的组织方式，以及数据控制方面的细节，它是数据库的存储观

## 数据驱动 DBTAG

采用数据驱动而非[硬编码](http://baike.baidu.com/view/2024903.htm" \t "http://baike.baidu.com/_blank)的方式，许多策略变更和维护都会方便得多，大大增强系统的灵活性和扩展性。

## Unique索引设计模式 DBTAG

特别对于插入之前先检查记录是否存在，很适用。。比入注册用户名检查。。可以大大减少编程语言中的逻辑代码

## Trigger模式 DBTAG

可以在数据操作的时候提供一个检查，否则出发一个数据库返回码异常。。编程语言捕获这个返回码，转换为编程语言异常抛出

## 视图模式 视图是的代码更加简洁，可读性更好 DBTAG

## Oo模式，对于数据库sql这样的语言来说，适用oo模式可以大力提升存储过程可读性

## ****1nf 2nf 3NF设计模式**** DBTAG

## ****鲍依斯-科得范式（BCNF是的改进形式）**** DBTAG

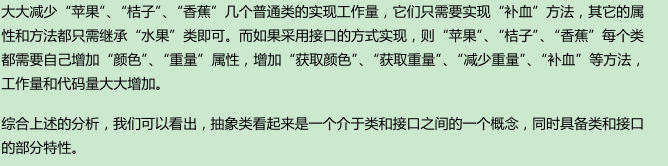
## Api映射法取代 warp法。

类似于hbuilder,将native api映射为我们的api，

## async异步模式 DBTAG

## AOP

## Abstract 优对于 interface



## Ast模式

## Auto type convert自动进行类型转换

Script

Jsp EL

Sql str>>num

## biz业务化模块

业务化的**业务模块，**框架，类库，模版，api

## 适当适用存储过程，提升代码简洁与可读性 dbtag

## Compiler as a Service

## ****CBD（核心Core+行为Behavior+驱动Driver）****架构模式

## Chain mode

## Confirm模式(解除页面跳转的问题)

## closure闭包模式

## ****components组件化****

## ****concurr并发模式(解除性能问题****

## DSL

## xxp vs AJAX

## DSLparser

## eventDriver事件驱动模式

## Code behind

## Cfg2ui (attilax 创建

## Biz\_index业务索引模式(attilax 创建,解

## dispater分发器模式

## declare声明式编程模式.

## Dsl模式

## filter过滤器模式

## Fp模式

## Engeen+script引擎+脚本模式(托管)

## ****frmwk框架化****

## 事件驱动

## 事件分发

## generic泛型模式

## -------------------h----n

## Html5 ui

## Hybrid化

## **immediately** 即时更新（热部署）快速迭代 持续交付

（script+engine）模式

使用原生开发，从招聘、开发、上线各个环节的效率都慢一倍以上，而且参与的人越多，沟通效率往往拖慢不止一倍。

配置也应该使用dev模式,立即更新..可以..不需要重启app

## Ioc/di

## Lazy load延迟初始化 cfg

## Linq

允许编写C#或者Visual Basic代码以查询数据库相同的方式操作内存数据。

## meta programe(anno,.net attr)

## microcore微内核

## msg消息化

这个图，是改用消息模式结果，用户注册功能模块，跟接收到消息处理各个任务直接，代码没有直接关联。如果需要增加一个接收注册消息，给其它人通知一下，只需要新增一个模块，订制注册消息，然后处理一个新流程即可。

目前各类开源项目里面，消息机制已经变得非常常见了。 但在wordpress这个blog框架，它的消息主题非常完善。几乎任何动作都有消息主题发送消息。 这样让开发者，只要订制相关主题，就可以，增加自己额外处理功能。 例如： 用户发一个帖子后。检测下用户是不是有广告信息，只要订制：发帖消息，然后，增加新功能，检测内容。发现不满足，直接屏蔽帖子。

有人估计要说，这类功能，现在很多框架有类似东西，例如，页面开始有个start事件，结束有个end事件。 确实，消息概念在很多框架里面确实有用。 但是，没有这么完毕的消息主题抛出。 **几乎做到整个系统任何操作，都能有消息，这样没有什么功能不能进行扩展了**。

## Mvc mvp mvvm

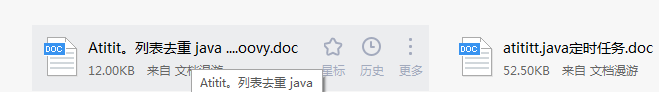
## ----o-------t

## Orm

## PageController 跟FrontController设计模式

## Pipe

## Pipe 管道模式、流式，链式



## platform

## plugin插件模式

## Reflection

## refresh刷新模式(解决缓存and静态的变量的值及时修改跟个读取问题)

## refresh刷新按钮模式...解决若干页面不料条件..and 分页..

## propCopy数据转换器..propCopy

## pool池模式

## Retry(attilax 创建,提升稳定性)

## Stream

## Self contain自包含，容器化隔离。。

iframe

## seri串口模式(attilax 创建,解除接口麻烦的)

## spa ajax化

## Try catch

## 决缓存不及时的问题)

## Theardlocal模式

## trigger触发器模式(attilax 创建)

## SOA

## single单例文件夹模式(解决模块复制问题,提取问题)

## Timer 模式...解决同步问题两个控件

控件的onchange事件不能生效..

## Theardlocal模式

## ----u----z

## Uapi dbtag

## Vm模式

## visual

## webService/HttpService

## workflow工作流

## Winform模式 native

## Webform模式

## Web page设计模式

## Weak/Soft references

## Promise设计模式

## Yield模式

## 管道模式，链式模式

## Plugin设计模式

# 反模式

## Reset web

# 参考资料

Atitit. 设计模式的总结Oa7.doc

atitit.软件设计模式大总结attialx总结v3 pa5.doc

atitit.设计模式的理解.txt

atitit.设计模式总结o5o

作者:: 绰号:老哇的爪子 （ 全名：：Attilax Akbar Al Rapanui 阿提拉克斯 阿克巴 阿尔 拉帕努伊 ）

汉字名：艾提拉（艾龙），   EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

Atiend