Atitit 之道系列心得总结书籍目录

[1. 项目管理系列 1](#_Toc12187)

[2. 企业文化系列 2](#_Toc28936)

[3. Atititi 软件界面gui开发之道 attilax著 3](#_Toc16656)

[4. atitit 性能提升之道.docx 6](#_Toc27516)

[5. Atitit 数据库之道.docx 9](#_Toc22112)

[6. Atitit 软件通讯之道attilax著 11](#_Toc27181)

[7. 可视化与报表图表系列 12](#_Toc14478)

[8. 知识管理之道 attilax著 13](#_Toc23170)

[9. Atitit 项目与产品战略之道 attilax著 v2 r77 14](#_Toc3598)

[10. atitit 技术新特性与趋势管理之道 14](#_Toc9508)

[11. atitit .大数据之道.v2 rb2 16](#_Toc4320)

[12. 人工智能之道系列 17](#_Toc10278)

[13. 文档管理之道 19](#_Toc19094)

[14. atitit 编程语言之道attilax著.docx 19](#_Toc881)

[15. Atitit.提升稳定性之道 27](#_Toc11661)

# 项目管理系列

itit 项目管理之道 attilax著

1.1.     各用一句话概括项目管理知识体系十大知识领域： 2

2. 3

3. 3

4. 项目管理总体有五个过程:启动过程、计划过程、实施过程、执行过程、收尾过程等 3

5. 九大领域的知识：范围管理、时间管理、成本管理、质量管理、风险管理、人力资源管理、沟通管理、采购管理及系统管理的方法与工具 3

6. 范围管理需求管理 3

6.1. 项目启动会 3

6.2. 需求搜集 3

6.3. 项目计划 路线图 3

6.4. Wbs分解 3

6.5. Backlog 备忘录 3

6.6. 范围变更控制 3

7. 时间管理进度管理 4

7.1. Wbs 4

7.2. 外包管理 4

8. 外部依赖管理 （采购管理） 4

8.1. Cbb体系 4

9. IPMA、PMI、PMBOK与PMP之间的联系与区别 4

10. 项目其他比较重要的管理内容 6

10.1. 团队文化管理 6

10.2. 培训管理 6

10.3. 技术趋势管理 6

10.4. 源码管理与分支管理 6

10.5. 文档管理 6

10.6. 知识管理 6

10.7. 解决方案管理 6

# 企业文化系列

1.1. 人员管理之道，如何落实以人为本 2

1.2. 组织架构之道 2

2. Uke信仰与价值观 （重要） 2

2.1. 使命感 宗教信仰（非常重要，占据了70%的重量，以及宗教典籍 ，宗教往往包含了此文化的哲学思想与世界观） 2

2.2. 语言文字（文化载体，很重要, 30%的重量） 2

2.3. Uke的宗教体系 2

2.4. 白皮书 蓝皮书 2

2.5. Uke价值观 2

3. 习俗 假日文化等 3

3.1. uke婚丧嫁娶法规 3

3.2. Uke历法 3

3.3. uke则选吉日大法 3

3.4. 音乐、绘画、雕塑等 3

3.5. 企业法定假日 3

3.6. uke饮食文化 可以吸收捕鲸文化 3

3.7. Uke住宿与建筑文化 3

3.8. Uke服饰文化 3

3.9. Uke交通文化 3

3.10. 放生活动 3

3.11. 赏赐内外 3

3.12. 情怀文化，休闲文化，下午茶文化 3

3.13. 各种比赛文化 体育精神 3

3.14. 交友文化 3

3.15. Uke医疗之道 4

3.16. 周某大扫除活动。。 4

4. 荣典体系 4

4.1. 企业全球地理区划 4

5. Uke的教育理念 4

5.1. TED演讲 4

5.2. 每周分享会 4

5.3. 培训会 4

5.4. 25 各类比赛&项目申报 4

5.5. 4

5.6. 驱灾除邪、卜算占卦 4

5.7. 其他 4

5.8. 文化墙，内刊，牌匾，横幅 5

# Atititi 软件界面gui开发之道 attilax著

1. 概览 2

1.1. 编程语言的发展 asm>native>vm>script>dsl 2

1.2. Ui的细化html ,css ,js,分离了布局，外观，与行为。。更加的领域特定了。。细化 3

1.3. 界面ui技术三大分类 native，插件体系，dsl体系 3

1.4. Gui界面语言的趋势，dsl 系列 h5系列 3

2. 界面控件dom体系 4

2.1. 布局体系  容器类 webkit控件 表单form winform 4

2.2. 菜单控件 右键菜单，托盘菜单 工具条 4

2.3. 常用控件 文本框，按钮，标签等 4

2.4. 数据控件 表格 树形控件 4

2.5. 多媒体 4

2.6. 其他 托盘图标，文件与文件夹选择 对话框 4

2.7. Icon图标 font icon 4

2.8. H5 体系 4

2.9. 报表与图表 （柱状图，饼图，线图趋势图，金字塔，地图，架构图）等 4

3. Gui风格体系 4

3.1. Page体系 h5 5

3.2. Windows体系 5

3.3. stage舞台 场景Scene 体系 5

4. 界面自绘 像素体系 5

4.1. 2d paint GDI+绘图 5

4.2. H5 canvas 5

4.3. Cocos2d 5

5. 事件处理与界面逻辑script 6

5.1. Gui线程 6

5.2. 拖放 6

5.3. Js 6

6. 特效与动画 6

6.1. 过渡、动画和变换 6

7. 架构体系 6

7.1. Bs cs 桌面 web 移动 6

7.2. 离线Web应用程序 6

8. Gui工具与框架与类库 6

8.1. Dw cs ajax fetch vue jquery 6

8.2. 双向绑定 6

8.3. Swing javafx wpf winform qt h5 6

8.4. 客户端mvc 6

9. 其他 7

9.1. Webkit渲染， 7

9.2. 国际化 7

9.3. 自定义外观　　样式表 子类化 css 7

10. Plugin体系 插件 7

11. 三维图形 7

11.1.  使用OpenGL绘图 three.js 7

11.2. 使用帧缓存对象生成叠加 7

12. 参考资料 8

1.概览

1.1.编程语言的发展 asm>native>vm>script>dsl

从机器语言，汇编语言到本地native语言(c c++) 到vm语言(java c#) 再到脚本语言(js php python等) 再到dsl（h5 sql 图像处理halcon matlab）

语言层次越高，可读性一般越好，可移植性越好，不过性能也越差了。。当然如果使用同样的类库的话，只是写点胶水代码的话，差别到是相对来说不大了。

机器语言， 可以说彻底淘汰，唯一的优点就是性能，其他基本全是缺点。。

汇编语言，貌似也基本淘汰，比起机器语言，提升了很大的可读性。。

native语言(c c++) 相对汇编语言可读性又大幅提升，性能方面不如汇编，但比vm语言要高。。。缺点是开发效率，以及可移植性仍然需要提升。。很多图像库貌似都是使用此开发。。依赖于性能的应用大有所为。

vm语言(java c#) 改进了native语言的问题，增加一个vm层隔离开了os。目前的业界高层应用开发主力

脚本语言 带来更高的生产力。但目前ide的问题，大型复杂企业级开发还难当重任，目前貌似在轻复杂度代码领域比较大发展。

Dsl语言 ，领域特定语言。。比如H5做界面。。Sql做数据库查询。Matlab halcon使用的语言，用来做图像处理。。

更高的开发效率，特别对于特定领域，图像处理，界面，数据库查询等大有优势。。正则表达式，用来文本搜索等。

Dsl语言本身性能很差，但是它如果是调用类库的，类库使用底层语言书写的，所以对性能不影响。

1.2.Ui的细化html ,css ,js,分离了布局，外观，与行为。。更加的领域特定了。。细化

1.3.界面ui技术三大分类 native，插件体系，dsl体系

Native的就不推荐了，swing winform 安卓 ios native。。

插件体系也没落了，flash Silverlight Applet 等。。

Dsl体系正主流。。H5 wpf（xaml）。。但是推荐公有标准化的h5..不推荐wpf了，wpf就是ms的h5。。Java体系基本没有标准化的dsl，只有一些builder框架有一些私有的h5.。。

微信小程序就是腾讯的h5，也是属于一种私有化dsl ，私有化h5

1.4.Gui界面语言的趋势，dsl 系列 h5系列

界面是个很专门的领域，需要领域特点语言来做。。

Dsl系列是目前最好的趋势了。。Dsl里面最好的额就是h5了，跨平台，通用。。

2.UI趋势 cli>gui>nui/cui

3.界面控件dom体系

3.1.布局体系  容器类 webkit控件 表单form winform

3.2.菜单控件 右键菜单，托盘菜单 工具条

3.3.常用控件 文本框，按钮，标签等

3.4.数据控件 表格 树形控件

3.5.多媒体

3.6.其他 托盘图标，文件与文件夹选择 对话框

3.7.Icon图标 font icon

3.8.H5 体系

3.9.报表与图表 （柱状图，饼图，线图趋势图，金字塔，地图，架构图）等

4.Gui风格体系

4.1.Page体系 h5

4.2.Windows体系

4.3.stage舞台 场景Scene 体系

展示stage舞台，stage舞台是一个类似于Swing中的JWindow的顶级容器，代表一个窗口。它用于容纳场景Scene，场景Scene是一个类似于Swing的JFrame的容器

5.界面自绘 像素体系

5.1.2d paint GDI+绘图

5.2.H5 canvas

5.3.Cocos2d

6.事件处理与界面逻辑script

6.1.Gui线程

6.2.拖放

6.3.Js

7.特效与动画

7.1.过渡、动画和变换

8.架构体系

8.1.Bs cs 桌面 web 移动

8.2.离线Web应用程序

9.Gui工具与框架与类库

9.1.Dw cs ajax fetch vue jquery

9.2.双向绑定

9.3.Swing javafx wpf winform qt h5

9.4.客户端mvc

10.其他

10.1.Webkit渲染，

10.2.国际化

 使用Unicode

　　18.2 让应用程序感知翻译

　　18.3 动态切换语言

　　18.4 翻译应用程序

10.3.自定义外观　　样式表 子类化 css

11.Plugin体系 插件

12.三维图形

12.1. 使用OpenGL绘图 three.js

12.2.使用帧缓存对象生成叠加

13.参考资料

《HTML5权威指南【非常全面详实的网页设计参考书】》(（美）弗里曼 著)【简介\_书评\_在线阅读】 - 当当图书.html

Atitit gui界面ui技术发展史与未来趋势

《C++ GUI Qt 4编程（第二版）(Trolltech的Qt培训教材，生动、全面、深刻地阐明了Qt程序的设计理念，轻松创建跨平台的解决方案。)》(（加）布兰切特)【简介\_书评\_在线阅读】 - 当当图书.html

《Java Swing图形界面开发与案例详解》(王鹏)【简介\_书评\_在线阅读】 - 当当图书.html

《C# WinForm 实践开发教程 (软件职业技术学院“十一五”规划教材)》(钱哨)【简介\_书评\_在线阅读】 - 当当图书.html

《JavaFX本质论》(（美）安德森)【简介\_书评\_在线阅读】 - 当当图书.html

# atitit 性能提升之道.docx

1. 性能的几个指标主要是响应时间（Response time） 2．吞吐量（Throughput） 3

2. 性能提升的5个原则与性能提升模型 3

2.1. 分类优先级，与分区域，减少要操作或提取的内容与范围 3

2.2. div分而治之 聚沙成塔 分布式 并发 3

2.3. 大力提升单体处理能力 3

2.4. 异步 提升快组件与慢组件的同步模式，防止快速组件等待慢速组件 4

2.5. Cache法 空间换时间大法 “凡是看得见的,都是暂时的,只有看不见的,才是永远的 4

2.6. Hybrid 混合综合模式 4

3. 性能问题的工具概览 4

3.1. 问题程序的全局资源占用统计 Os资源占用统计比如win的资源管理器，与linux的top等工具 4

3.2. 性能检测工具 Profile工具 4

3.3. 自制profile工具 5

4. 常用性能检测性能问题的检测方法 5

4.1. 人工检查法 5

4.2. 常见性能故障排查表，根据性能异常情况，按图索骥，解决方法 5

4.3. 性能检测流程，防止遗漏重要的流程节点 5

4.4. 对常用涉及性能的api，使用团队内部api类库包装代替通用api，固化性能算法 5

5. 常见性能问题的解决方案 6

5.1. -高频功能与步骤的优化 6

5.2. Ui性能问题 6

5.3. 通讯性能问题 6

5.4. 数据库性能问题 6

5.5. Atitit 提升图像处理的性能方法总结attilax 6

5.6. 使用分布式，微服务msa等架构模式提升性能 6

6. 常用的性能问题标准化检测流程最佳实践bp总结 6

7. 编程语言的性能提升（以java项目为例子） 8

7.1. 性能检测Profile工具jvisualvm.exe JProfiler8 8

7.2. 常见的性能问题分类总结 9

7.3. 使用了性能较差的api或者类库，解决方法：使用更快的api代替 9

7.4. 操作过多，使用事务模式 批量集中提交处理 9

7.5. 是否只使用了一个cpu，解决办法使用多线程异步模型 10

7.6. 多线程线程数设置不合理，导致cpu大量耗费在切换线程上。 10

7.7. 使用事件驱动模式，本质上属于同步倍频操作 异步 10

7.8. 缓存频繁使用的对象变量等。 使用对象池 cache等， 10

7.9. 同步锁频繁冲突，使用无锁结构 10

7.10. 项目编译性能变慢，改动一行代码可能需要一分钟编译，Eclipse ide增量编译是开启的 10

7.11. 使用了Spring框架后项目启动运行性能很慢的解决，大型项目可能需要好几分钟，十分影响进度 10

7.12. 改动一个东西，需要频繁启动web服务器比如tomcat，解决方案完全热部署 11

7.13. 其他 参考互联网xxx语言优化 11

8. 数据库的性能提升 11

8.1. 统计数据库的表记录数量，以及磁盘文件体积，评估是由于大数据量还是sql性能 数据库连接数量过多引起的。 11

8.2. 数据库的profile工具。找到性能较差的sql 11

8.3. Explain 当找到具体的sql语句了，使用Explain 来查看sql的具体过程，一般是索引没有引用到。 12

8.4. 索引原理 常见的类似于书籍目录，hashmap原理，散列表 dic字典 13

8.5. 如何设置正确的索引 14

8.6. 设置了索引，但是使用expian却发现，没有使用到索引 14

8.7. 报表等复杂统计sql很慢，解决方案，业务索引 15

8.8. 适当的使用UDF自定义函数， 触发器与视图 等功能 15

8.9. 适当使用数据库提供的定时任务等功能 15

8.10. 使用存储过程大力减少数据网络往返，在多表往返操作情况下 15

8.11. 移植其他数据库的性能高的特性功能 15

8.12. 优先操作 HIGH\_PRIORITY 滞后操作LOW\_PRIORITY 延时插入 INSERT DELAYED 15

8.13. 临时表模式与 强制使用临时表 SQL\_BUFFER\_RESULT 16

8.14. 设置全文索引代替like查询 16

8.15. 使用solr 等第三方全文索引代替数据库的全文索引，降低数据库的资源占用 17

8.16. 使用数据库连接池和线程池 17

8.17. 使用cache redis等 17

8.18. 分区 分区是个很重要的提升性能操作 17

8.18.1. 分区的原理 17

8.18.2. 分区类型 17

8.18.3. 什么时候需要分区，基本一个是单表记录数量，达到几百万到一千万，或者数据表文件超过1g左右时 18

8.19. 分库分表 18

9. 多机情况下可以使用分布式，微服务msa等架构模式提升性能 18

10. 性能优化的一些误区 反模式与需要注意的地方 18

10.1. 过度设计 vs 事后优化策略 18

10.2. 可读性 vs 性能 vs可维护性 19

10.3. 性能指标响应时间 vs吞吐量 的矛盾 19

11. 常见的提升性能方面的书籍总结 19

11.1. 1. 并发（多线程,task) 1 20

11.2. 2. 《计算机组成与体系结构：性能设计（原书第8版）》(（美）斯托林斯...)【简介\_书评\_在线阅读】 - .html 1 20

11.3. 3. 《Java性能优化权威指南【Java性能优化圣经！Java之父重磅推荐！ 】》(Charlie...)【简介\_书评\_在线阅读】 - .html 2 20

11.4. 4. 《性能之巅：洞悉系统、企业与云计算》(Brendan...)【简介\_书评\_在线阅读】 - .html 4 20

11.5. 5. 《Java性能权威指南》([美]...)【简介\_书评\_在线阅读】 - .html 4 20

11.6. 6. 《高性能JavaScript》(Nicholas...)【简介\_书评\_在线阅读】 - .html 5 20

11.7. 7. 《JavaScript性能优化：度量、监控与可视化（首部系统化阐述JavaScript性能优化的经典著作，Amazon全五星好评）》([美]...)【简介\_书评\_在线阅读】 - .html 5 20

11.8. 8. 《高性能网站建设指南：前端工程师技能精髓》(Steve...)【简介\_书评\_在线阅读】 - .html 6 20

11.9. 9. 《Web性能权威指南》([加]Ilya...)【简介\_书评\_在线阅读】 - .html 6 21

11.10. 21

12. 参考资料 22

# Atitit 数据库之道.docx

1. 数据存储的历史与发展历程 4

1.1. 卡片时代 4

1.2. Rom存储时代 4

1.3. 扇区存储 4

1.4. 文件存储时代 4

1.5. 数据库时代 4

2. 初始概念 5

2.1. 数据库的类型，网状，层次，树形数据库，kv数据库。Oodb 多媒体数据库 5

2.2. Er模型 Sql 聚合运算 5

2.3. 索引btree，hash 全文哦 5

2.4. 并发控制与lock 5

2.5. Acid数据库完整性 关系模型 事务管理 5

2.6. ：1NF、2NF、3NF和BCNF  5

3. 存储引擎 5

3.1. 存储引擎 5

3.2. 数据库表存储与表空间 6

3.3. 分区 6

3.4. 分库分表 6

4. 索引引擎 7

4.1. 1. 索引的分类1 7

4.2. 1.1. 按照存储结构划分btree,hash,bitmap,fulltext1 7

4.3. 1.2. 索引的类型  按查找方式分，两种，分块索引 vs编号索引1 7

4.4. 1.3. 顺序索引  vs 散列索引2 7

4.5. 1.4. 按索引与数据的查找顺序可分为 正排与倒排索引2 7

4.6. 1.5. 单列索引与多列索引 复合索引2 7

4.7. 1.6. 分区索引和全局索引 2 7

4.8. 1.7.  Trie树一般指字典树 又称单词查找树，Trie树2 7

4.9. 1.8. 稠密索引 vs 稀疏索引3 7

4.10. 1.9. 多级索引 vs 单击索引3 7

4.11. 1.10. 索引模式extent和blockmap3 7

4.12. 2. 索引建立，更新的流程使用触发更新索引的事件4 7

4.13. 3. ISAM算法 索引顺序存取方法”（Indexed Sequential Access Method） 索引常用的存储结构 B树文件 叫做“，缩写为。4 8

4.14. 4. 索引文件的合并问题5 8

4.15. 分布式存储的索引 8

5. sql与Sql引擎 8

5.1. 常用sql select update delete 8

5.2. 条件where sql 8

5.3. 数据库翻页 limit offset系列 8

5.4. 多表join 8

5.5. 排序 8

6. 报表与统计sql 8

6.1. 分组groupby 8

6.2. 聚合统计函数 8

7. 标准库函数 9

7.1. 加密解密与二进制转换函数 9

7.2. 数学系列 9

7.3. 字符串系列 9

7.4. 日期运算系列 9

7.5. 控制流程函数 条件判断函数； 9

7.6. 5 搜索函数 9

7.7. 6 加密函数 9

7.8. 7 信息函数 9

7.9. 8 其他函数 9

7.10. 9 聚合函数 9

7.11. 系统信息函数； 9

7.12. 格式化函数； 9

8. 数据库报表系统 9

9. 数据库Sql编程，存储过程sp，视图触发器 10

9.1. Sp存储过程 10

9.2. Udf自定义函数 10

9.3. Trigger触发器 10

9.4. View视图 10

9.5. 游标 10

10. 存储过程异常处理 10

11. Event事件与计划任务 10

12. 表结构Uml模型与表结构文档 10

12.1. View>>er chart 自动生成er图表 10

12.2. 表文档view>detail info 即可看到注释 10

12.3. 表结构与注释，可以使用sql ddl 10

13. 元数据系列 10

14. Sql标准化 11

14.1. 1. Sql标准化的历史3 12

14.2. 1.1. Sql92标准4 12

14.3. 1.2. Sql99标准4 12

14.4. 1.3. SQL:2003为例，它包括以下9个部分 5 12

14.5. 1.4. Sql2006标准6 12

14.6. 1.5. Sql2008标准7 12

14.7. 1.6. SQL:2011 7 12

15. Sql性能优化与explain 12

16. 数据库的安装运行与维护部署系列 12

16.1. 导出数据与结构 12

16.2. 导出结构 12

16.3. 导入数据 12

16.4. 备份与恢复 12

16.5. 数据库表统计（体积占用与行数） 12

16.6. 服务器监控与历史日志 13

17. Sql解析与美化 13

17.1. 源码高亮 13

17.2. Sql解析 13

18. 数据库访问接口  jdbc odbc ado oledb 13

19. Oracle mysql mssql sqlserver数据库 13

20. 数据库的gui管理工具 14

21. 数据库新特性 14

22. 数据同步与结构同步 14

22.1. 项目备份与补丁压缩 14

22.2. 结构同步 14

22.3. 数据同步 14

23. 常用的数据库算法 14

23.1. Join算法 15

23.2. Groupby算法 15

23.3. 聚合union算法 15

24. 数据库表分析 15

25. 数据库事务 15

25.1.    1. 事务隔离级别的作用 1 15

25.2.     2. 在的隔离级别 2 15

25.3. 3. 常见数据库的默认管理级别 15

26. Daas data as a service 15

26.1. Sql as rest 参数，dsl参数 16

26.2. Rest +jdbc 16

27. 内存数据结构与nosql db 16

27.1. Map List Set、 16

27.2. Redis kvdb 16

28. 参考书籍 16

# Atitit 软件通讯之道attilax著

Atitit 软件通讯之道attilax著

1. Atitit 通讯与网络发展史 attilax总结 1

1.1. 1.1. 漂流瓶 1 2

1.2. 1.2. 人工信号机 1 2

1.3. 1.3. 1844年5月24日 电报网 1 2

1.4. 1.4. 电话网络 2 2

1.5. 1.6. 互联网 与pc 3

1.6. 1.7. 短信 2 3

1.7. 1.8. 对讲机 2 3

1.8. 手机 3

2. Atitit 通讯终端 手机 pc 汽车 移动通信终端 watch手表 3

2.1. 可穿戴通讯设备 glass系列 3

2.2. Pad平板系列 3

2.3. Tv电视系列 3

3. Atitit.前后端数据通讯最快速方式还是服务端xxp技术.docx 3

4. Atitit 跨平台通讯解决方案 3

4.1. 1.1. 基于http+rest，可读性高，开发简便 1 4

4.2. 1.1.1. HTTP协议的主要特点可概括如下： 1 4

4.3. 1.2. Rest 2 4

4.4. 1.3. 基于hessian二进制数据+http 2 4

4.5. 1.4. 基于socket，更高性能 ava mina netty 2 4

5. 通讯安全性 4

5.1. Atitit Rsa 系统解决方案，解决非https系统的通讯安全性.docx 4

5.2. https 4

6. Dsl通讯接口 4

6.1. Atitit 基于dsl的Rest 接口通讯解决方案.docx 4

# 可视化与报表图表系列

Atiitt 可视化 报表 图表之道 attilax著

1.  信息可视化 1

2. Gui可视化 2

2.1. atitit 知识的可视化.docx 2

3. 可视化的可视化= 图形化 +表格化 2

4. Html可视化 与可视化插件 2

5. 可视化大力提升用户体验== 2

6. 报表系统（三大图表，金字塔，组织结构图等,仪表板，地图） 2

6.1. Dataset 数据集合 4

7. 可视化的具体实现（canvas，svg ，dom） 4

8. 报表图表框架 4

9. 常见的可视化书籍 4

9.1. 《可视化数据》目录 4

9.2. 数据可视化之美(通过专家的眼光洞察数据) 6

10. Other 7

10.1. Atitit 可视化 名字的可视化 日期的可视化.docx 7

10.2. atitit.可视化编程的历史与发展.txt 7

10.3. atitit.可视化编程规则引擎与工作流的区别.txt 7

11. 参考资料 7

# 知识管理之道 attilax著

atitit 知识管理之道 attilax著 .docx

1. 第一章 读书即是获取知识，获取资料，获取信息的一种方式 3

2. 第二章 获取资料信息的目的，有什么作用，有什么用途？？ 3

2.1. 扩大视野，辅助决策，提供多个解决方案 3

2.2. 实现从无到有的跨越 3

2.3. 找准方向，不在迷茫，方向比努力更重要 3

2.4. 提升幸福感 与成就感 3

2.5. 提升个人与组织的软资产，提升资产安全度。。 3

2.6. 获得额外加持，增强提升个人与组织学历 能力 4

2.7. 大力减少成本，减少无谓浪费。 4

2.8. 其他 了解趋势 4

3. 第三章 读书与获取资料的原则总结 4

3.1. 简单易读原则 4

3.2. 二八原则 轻松原则 4

3.3. 广博优先于精深原则（80%广博+20%精深） 5

3.4. 软资产优先于硬资产原则 5

3.5. 通用原则 普世原则 5

3.6. 时效性原则，长期原则 务必坚持长期适用的 5

3.7. 抓大放小 聚焦宏观与大方向原则 细节是魔鬼 5

4. 要读哪些书？？获取哪些资料与信息？ 5

4.1. 建立较为完善的个人与组织知识体系树 5

4.2. 各个专业领域（管理 市场 技术等） 5

4.3. 宗教与文学与文化哲学精神领域：： 5

4.4. 行政领域：行政就是实行政策，实现想法。。这个很常见的活动了。。行政就是想法的实现与行动 5

4.5. 历史，要预知未来就要关注历史，成为预测的大巫师 6

4.6. 基础知识 衣食住行一类的 旅游 户外 打猎 食材表 建筑 等 6

4.7. 医学 6

4.8. 建立较为完善的知识层次书籍 文章表 6

5. 理解与吸收知识 6

5.1. 建立自己的知识管理，知识图谱体系 6

5.2. 体系化，分门别类 ，方便查找 6

5.3. 专题化 ，把某一类汇集起来，另外一个维度的方便查询 7

5.4. 网络点化，索引化 方便按图索骥 7

5.5. 提取要点 摘要方法 7

6. 索引与目录 8

6.1. 分类目录（树形索引 聚集索引） 8

6.2. 页码目录 (hash索引 ） 8

6.3. 关键词索引 （全文索引） 8

6.4. Map索引 （图索引 二维索引） 8

6.5. 索引目录优化 8

7. 知识的存储 8

8. 应用知识与资料 8

8.1. 各种总结 完善 8

9. 知识的储存与备份 8

10. 从哪里获取 8

11. 参考资料 8

# Atitit 项目与产品战略之道 attilax著 v2 r77

# atitit 技术新特性与趋势管理之道

**Atitit 趋势管理之道 attilax著**

**1. 何处获取趋势 3**

**1.1. 技术雷达 3**

**1.2. Google trend 3**

**1.3. 信息技术研究和顾问公司Gartner，于近日发布了《2017十大技术趋势》报告，作为每年最有影响力的报告之一 3**

**1.4. 各种ted会议 3**

**1.5. 行会产业规划 3**

**1.6. 政府产业规划 比如软件和信息技术服务业发展规划（2016-2020年）》（ 3**

**1.7. 未来学 先知拜访 3**

**1.8. 各种前沿信息获取 3**

**2. 历史趋势与热点 3**

**2.1. name 3**

**2.2. bbs 3**

**2.3. 门户 4**

**2.4. 4**

**2.5. 、搜索、 4**

**2.6. 电商、团购 4**

**2.7. 博客 4**

**2.8. 微博 4**

**2.9. 微信 4**

**2.10. qq 4**

**2.11. 开心网 4**

**2.12. 4**

**2.13. 4**

**2.14. 5**

**2.15. qrcode二维码 5**

**2.16. 互联网金融元年 5**

**2.17. 滴滴打车 5**

**2.18. 共享单车 5**

**2.19. 共享充电宝 5**

**2.20. 5**

**2.21. 团购 5**

**2.22. 外卖 5**

**2.23. 直播 6**

**2.24. 自媒体 6**

**2.25. 智能硬件 移动支付 手游 短视频 新零售 知识付费 区块链 6**

**2.26. 无人售货 新零售 6**

**2.27. 垂直电商 6**

**2.28. vr 6**

**2.29. 内容创业 7**

**2.30. ai 7**

**2.31. 陌生社交，一时间人人都在“约”。 7**

**2.32. 内容平台的崛起，比如爱奇艺、今日头条、喜马拉雅。 7**

**2.33. 智能手机android ios 7**

**2.34. netdisk 9**

**2.35. 云 9**

**2.36. 大数据 9**

**2.37. 在线视频 9**

**2.38. lbs 9**

**2.39. iot物联网 9**

**2.40. 无人机 9**

**2.41. 云计算 9**

**2.42. 3d打印 9**

**2.43. rss 9**

**2.44. 娃娃机 10**

**2.45. nfc 10**

**2.46. 手机支付/移动支付/移动电子商务/手机钱包/……这些词印象里是在2007年左右在国内被吵热 10**

**3. 常见趋势 10**

**3.1. 物联网趋势 11**

**3.2. 11**

**3.3. Ai趋势 11**

**3.4. 11**

**3.5. 共享经济趋势 11**

**3.6. 11**

**3.7. 参考资料 11**

**3.8. Atitit 未来趋势把控的书籍 attilax总结 11**

**4. 未来学 1 11**

**4.1. 著名的未来学家 1 11**

**4.2. 未来学书籍 2 11**

# atitit .大数据之道.v2 rb2

1. 大数据是什么 4v 大规模 非结构化 2

2. 大数据技术综述 巨量数据中 快速获得有价值信息的技术 3

3. 大数据的格式种类:普通数据 以及图片视频半结构化 非结构化数据 3

4. 大数据特点: 价值密度低 3

5. 大数据的应用: 3

7.1. 行为分析 3

8. 大数据优点 实时性 4

9. 关联技术 4

9.1. 刷票器/自动化web操作/浏览器引擎 4

9.2. 注册机/登录器/ 4

9.3. 发帖机/ 4

9.4. 语音识别技术 4

9.5. 手写识别技术 4

9.6. 验证码../手机验证码. 4

9.7. 水军好评差评 4

9.8. 云计算、移动互联网 4

9.9. 物联网. 5

9.10. 非结构化nosql数据库技术 5

9.11. 分布式文件系统 5

9.12. DDBS 分布式存储数据库 5

9.13. ETL工具 5

9.14. 采集器/分布式网络爬虫 5

9.15. 分布式编程语言,ERLANG并行处理计算 5

9.16. 机器智能化学习 5

10. 大数据的生产/来源 fromB到fromC 6

11. 大数据处理的流程(采集>>预处理>>导入>>统计分析>>数据挖掘) 6

12. 数据采集以及承载 6

13. 数据清洗、转换、集成ETL工具 6

14. 大数据的存储 数据库与云存储 文件存储 7

15. 统计分析 统计分析与模型预测 与结果呈现 7

16. 大数据的5个基础分析方法可视化分析，数据挖掘，预测 7

17. 数据挖掘 8

18. 大数据的传输 8

19. 应用领域 8

20. 数据分析师的入门进阶(20天时间) 9

21. 常用大数据技术 9

21.1. Erlang分布式大并发编程语言 9

21.2. 超强的并发性 10

21.3. 天生的分布式 10

21.4. 灵活多样的错误处理 10

21.5. 代码热替换 10

21.6. ● 软实时性- 11

22. NoSQL数据库 11

23. Ddms--Storm —— Twitter开发的ddms 11

23.1. 4.1. HBase 3 12

23.2. 4.2. Hypertable 3 12

23.3. 4.3. Hadoop -----Hive 3 12

23.4. 4.4. 基于MySQL的分布式数据库实践 3 12

23.5. 4.5. 使用MySQL federated 引擎构建 MySQL 分布式数据库访问层 3 12

23.6. 4.6. 用Amoeba构架MySQL分布式数据库环境 4 12

23.7. 4.7. Facebook开源的Cassandra 4 12

23.8. 4.8. 淘宝的Amoedb 4 12

24. 案例: 12

24.1. 谷歌流感趋势"的工具 12

24.2. 这个工具工作的原理 13

24.3. 大数据应用案例之：医疗行业 13

24.4. 大数据应用案例之：能源行业 13

24.5. 大数据应用案例之：通信行业 14

24.6. 大数据应用案例之：零售业 14

24.7. 麻省理工学院利用手机定位数据和交通数据建立租房规划。[5] [6] 15

25. 参考: 15

# 人工智能之道系列

1.   导读： 计算机视觉、机器学习、自然语言处理、机器人和语音识别是人工智能的五大核心技术，它们均会成为独立的子产业。 2

1.1. 五大分类 机器视觉，机器听觉，NLP自然语言处理机器文字处理 ，机器人自动化，机器学习 2

1.2. 修改历史 2

2. 目前比较好的解决 3

2.1. 文本分析和NLP：自然语言处理（NLP）使用和支持文本分析， 3

2.2. 机器人流程自动化：使用脚本及其他方法，实现人类操作自动化 3

2.3. 生物特征识别技术 3

2.4. NLP 3

2.5. 图片验证码识别，很成熟 3

2.6. 普通话语音识别 tts 语音转文本 文本转语音 3

2.7. 如指纹识别、 3

2.8. 印刷体识别 ocr 3

2.9. 车牌识别 3

2.10. 场景识别 3

2.11. 自动翻译 4

2.11.1. 自动工程 自动驾驶（OSO系统） 4

2.12. 大数据+机器学习ML 4

2.13. 聊天机器人 4

2.14. Qr 条码识别 4

2.15. 物联网 icir识别 4

3. 技术要点列表 结构化表格列表 4

3.1.1. 机器视觉 4

3.1.2. 图片识别 4

3.1.3. 视频内容分析 4

3.1.4. ocr印刷体识别 文字识别 4

3.1.5. 生物特征识别技术 5

3.1.6. 图像搜索 5

3.1.7. 机器人 5

3.1.8. 聊天机器人 5

3.1.9. 信息采集机器人 5

3.1.10. 信息发布机器人 5

3.1.11. 其他机器人 5

3.1.12. 自然语言处理 6

3.1.13. 自动翻译 6

3.1.14. 自动化摘要 6

3.1.15. 自动化索引 6

3.1.16. 自动化tag技术 6

3.1.17. 知识图谱 6

3.1.18. 语音识别 6

3.1.19. 普通话语音识别 tts 语音转文本 文本转语音 6

3.1.20. 语言合成 6

3.1.21. 机器学习、 6

4. 四大领域 人机对弈 7

4.1.1. 模式识别 7

4.1.2. 自动工程 7

4.1.3. 知识工程 7

4.1.4. 知识工程 7

4.2. 5.智能检索 8

4.3. 6.专家技术 8

4.4. 决策管理：引擎将规则和逻辑嵌入到人工智能系统 8

5. 需要解决的问题 8

5.1. 目标识别。。照片里面有哪些东西。。 8

5.2. 智能问答？？如何不再迷茫 8

6. 参考资料 8

# 文档管理之道

# atitit 编程语言之道attilax著.docx

1. 编程语言常见概念与理论 9

1.1. solid原则 9

1.2. 变量6属性 9

1.3. 三大流程 9

1.4. 异常控制流程 9

1.5. 流程图（Flow Chart 9

1.6. 多分支结构 9

1.7. 表达式（中缀前缀表达式 9

1.8. 运算符 9

1.9. 数据类型 9

1.10. Lambda 表达式 9

1.11. 方法引用 11

1.12. 生命周期 11

1.13. Cohesion 11

1.14. 耦合Coupling 11

1.15. 过程 11

1.16. 函数 11

1.17. 返回值 11

1.18. 循环(loop), 11

1.19. 递归(recursion), 11

1.20. 遍历(traversal), 11

1.21. 迭代(iterate). 12

1.22. 泛型 12

1.23. 环境变量 12

1.24. s-exp 12

1.25. 类型系统 12

2. 编程语言种类 and趋势 逻辑式语言..函数式语言...命令式语言 12

3. 语言实现 14

3.1. 词法分析 14

3.2. fsm状态机 14

3.3. 正则表达式 14

3.4. 词性标注 14

3.5. 语法分析 14

3.6. ast 14

3.7. 语义分析 14

3.8. 内部DSL实现模式 14

3.9. 外部DSL 14

3.10. 生成AST 14

3.11. BNF 15

3.12. 正则表达式表的词法分析器 15

3.13. 递归下降法语法解析器 15

3.14. 解释器 15

3.15. 表达式生成器 15

3.16. 嵌套函数 15

3.17. 方法级联 15

3.18. 状态机fsm 15

4. 函数式理论 章节 16

4.1. lambda 16

4.2. stream 16

4.3. fluce接口 16

4.4. 连锁接口 16

4.5. pipe 16

4.6. map & reduce 16

4.7. pipeline 16

4.8. 高阶函数 16

4.9. 闭包 16

4.10. 表达式特化 16

4.11. 表达式合成 17

4.12. 外循环 17

4.13. 内循环 17

4.14. 惰性求值 17

5. 类与对象实现 章节 19

5.1. class模板 19

5.2. dock type 19

5.3. is-a has-a 19

5.4. mix in 19

5.5. oo三大特性 19

5.6. solid原则 19

5.7. 抽象方法 19

5.8. 抽象类和 19

5.9. 单根继承 19

5.10. 单例 19

5.11. 动态绑定（dynamic binding 19

5.12. 动态对象 20

5.13. 对象持久性（object persistence 20

6. 章节 22

7. dsl 章节 22

8. io输入输出 章节 22

8.1. 文件的io 22

8.2. 网络的io 22

8.3. rest接口 22

8.4. http 22

8.5. socket 22

8.6. console重写 22

8.7. 异常打印string 22

8.8. 字符流 22

8.9. 字节流 23

8.10. io体系 23

8.11. 图片的io 23

8.12. 二进制io 23

8.13. 文本io 23

8.14. 字符编码 23

8.15. request 23

9. 并发与锁机制 章节 24

9.1. Volatile 内存屏障 24

9.2. cas 24

9.3. atomic 24

9.4. 同步关键字 24

9.5. ReentrantLock 24

9.6. 自旋锁 24

9.7. 信号量 24

9.8. 互斥锁Mutex 24

9.9. 线程死锁 24

9.10. 线程死锁kill 24

9.11. 异步 25

9.12. 线程池 25

9.13. task模型 25

9.14. futuretask模式 25

9.15. 毒丸 25

9.16. threadlocalhost 25

10. api设计 章节 26

10.1. 方法连 26

10.2. jquery 26

10.3. guava 26

10.4. option 26

10.5. 面向接口编程 26

10.6. 工厂方法优于构造函数 26

10.7. polyfill 26

10.8. dsl命名法 26

10.9. 参考知名api 26

10.10. 归一化 27

10.11. 命名参数 27

11. gc资源释放 章节 28

11.1. 分代 28

11.2. 基于时间 28

11.3. 引用计数 28

11.4. amr 28

11.5. watchdog 28

11.6. 28

12. 调试技术 章节 28

12.1. log调试 28

12.2. ide调试 28

12.3. 监测工具 28

12.4. sdk调试支持api 28

13. 反射 章节 29

14. 异常处理 章节 30

14.1. 异常捕获策略 30

14.2. 处理策略 30

14.3. 抛出策略 30

14.4. 跨语言异常 30

14.5. 异常转换 30

14.6. 异常序列化 30

14.7. Atitit 异常and 流程控制 30

14.8. 返回码模式 30

14.9. 终止模式 vs 恢复模式 32

14.10. 异常机制的设计原理 32

14.11. 类库实现异常catch 32

14.12. Check ex vs unchk ex 32

14.13. 业务异常 32

14.14. 异常分类 32

14.15. 重新抛出异常和异常链 32

14.16. 全局异常捕获 33

15. 事件机制 章节 33

15.1. event对象 34

15.2. 事件目标（event target 34

15.3. 事件传播（event propagation 34

15.4. event handler 34

15.5. 事件分发器 34

15.6. 事件委托 34

15.7. 事件代理 34

15.8. 事件循环 34

15.9. 事件驱动 vs 轮询机制 34

15.10. 事件队列 35

15.11. vs消息机制 35

15.12. 事件注册 35

15.13. 事件调用 35

15.14. 事件对象模型 35

15.15. 事件冒泡 35

15.16. 事件捕获 35

15.17. 事件底层 中断原理 35

16. 重构 章节 36

17. 章节 36

18. 开发规范 章节 36

19. 序列化 章节 36

19.1. xml序列化 37

19.2. php hash序列化 37

19.3. json序列化 37

19.4. 二进制序列化 37

19.5. 循环引用解决 37

19.6. base64 37

19.7. bin2txt 37

20. ioc 章节 37

21. 性能提升 章节 37

22. gui 章节 38

22.1. gui原理 38

22.2. 界面引擎 38

22.3. 游戏引擎 38

22.4. 布局引擎 38

22.5. 控件系统 38

22.6. 复合控件 38

22.7. pagging翻页控件 38

22.8. css框架 38

22.9. 前端ui框架 38

22.10. css3 39

22.11. 动画 39

22.12. canvas 39

22.13. 音视频 39

23. other 章节 39

24. mvc 章节 39

25. ide 章节 39

26. exop互操作接口 章节 39

27. 常用协议 章节 39

28. 常用web服务器 章节 40

29. 代码管理与同步 章节 40

30. 安全与加密 章节 40

30.1. rsa 40

30.2. 证书 40

30.3. md5 40

30.4. aes 40

30.5. 3des 40

30.6. 注入 40

30.7. crsf 40

30.8. xss 40

31. 架构知识 章节 41

32. 文字处理 章节 41

32.1. 汉字处理 41

32.2. 正则表达式 41

32.3. auth 41

32.4. 全文检索 41

32.5. tie树 41

32.6. 规则查询 41

32.7. 参见图像处理知识图谱 41

33. 图像处理 章节 42

34. uml 章节 42

35. web体系 章节 42

36. 简单的图像处理 章节 42

37. vm 章节 42

38. api标准化 章节 43

38.1. 核心core标准化 43

38.2. 文本处理 43

38.3. 网络处理 43

38.4. 图像处理 43

38.5. 数据库处理 43

38.6. 进程处理 43

38.7. 鼠标键盘管理 43

38.8. 时间处理 43

38.9. 数字处理 43

38.10. 容器 44

39. 常用算法 章节 44

40. 软件工程 章节 44

41. 设计模式 章节 44

42. 标准化 章节 44

42.1. ecma标准 45

42.2. iso标准 45

42.3. apache fund 45

42.4. jcp jsr 45

42.5. gbk 45

42.6. rfc 45

42.7. w3c 45

42.8. ansi 45

42.9. omg 45

42.10. BCP，即Best Current Practice 45

43. 字符编码 章节 46

43.1. gbk 46

43.2. utf8 46

43.3. unicode 46

43.4. 大头端小头 46

43.5. ansi 46

43.6. ascii 46

43.7. utf bom 46

43.8. bigend 46

43.9. littleend 46

43.10. cjk 46

43.11. charset 47

43.12. base64 47

43.13. mime 47

44. bug管理 章节 47

45. 编程原则准则 章节 48

45.1. S.O.L.I.D原则(oo fp) 3 48

45.2. 3.1. Single Responsibility Principle (SRP)–职责单一原则 3 48

45.3. 3.2. Open/Closed Principle (OCP)–开闭原则 3 48

45.4. 3.3. Liskov substitution principle (LSP)–里氏代换原则 3 48

45.5. 3.4. Interface Segregation Principle (ISP)–接口隔离原则 3 48

45.6. 3.5. Dependency Inversion Principle(DIP)–依赖倒置原则 3 50

45.7. 4. Dky法则 dry kiss yagni 3 50

45.8. 5. Other原则 4 50

45.9. 5.1. 可读性优先原则 4 50

45.10. 5.2. 拥抱变化原则 扩展性 4 50

45.11. 5.3. ：迪米特法则最少知识原则”（Principle of Least Knowledge） 4 50

45.12. 5.4. 迪米特法则（Law?ofDemeter,?LOD） 4 50

45.13. 5.5. Don’t Repeat Yourself(DRY) 4 52

45.14. 5.6. Keep It Simple, Stupid (KISS) 4 52

45.15. 5.7. Program to an interface, not an implementation4 52

45.16. 5.8. You Ain’t Gonna Need It(YAGNI) 4 52

45.17. 5.9. 最小惊讶原则 遵守标准约定注释说什么代码就做什么 5 52

45.18. 5.10. 最小化耦合原则与最大内聚 5 52

45.19. 5.11. 隐藏实现细节原则 5 54

45.20. 5.12. 避免过早优化原则 5 54

45.21. 5.13. 得墨忒耳定律 5 54

45.22. 5.14. Common Closure Principle（CCP）–共同封闭原则 5 54

45.23. 5.15. Common Reuse Principle (CRP)–共同重用原则 5 54

45.24. 5.16. Hollywood Principle–好莱坞原则 5 54

45.25. 5.17. High Cohesion & Low/Loose coupling & –高内聚，低耦合6 55

45.26. 5.18. Convention over Configuration（CoC）–惯例优于配置原则 惯例优先原则6 55

45.27. 5.19. Acyclic Dependencies Principle (ADP)–无环依赖原则 6 55

45.28. 5.20. Postel's law伯斯塔尔法则 后向兼容性设计 6 55

45.29. 5.21. 软件，就像所有技术一样，具有天然的政治性。代码必然会反映作者的选择、偏见和期望。 6 55

45.30. 5.22. 首先为人类设计，其次为机器设计。 6 56

45.31. 5.23. 抽象原则 6 56

45.32. 6. 《编程法则》二十一条 6 56

46. datastruts数据结构 章节 57

46.1. map 57

46.2. list 57

46.3. 元祖 57

46.4. turl 57

46.5. meta、 57

46.6. dic 57

46.7. hashtable 57

46.8. hash 57

46.9. set 57

46.10. array 57

46.11. graph 58

46.12. linklist 58

46.13. tree 58

46.14. collect、 58

46.15. queuo 58

46.16. stack 58

46.17. heap 58

46.18. sort 58

46.19. 查找搜索 58

46.20. 遍历 58

46.21. fifo 58

46.22. datatable 59

47. 源码组织管理 章节 59

48. 标准库 章节 59

48.1. 字符串函数 60

48.2. 数学函数库 60

48.3. 日期运算 60

48.4. 聚合函数 60

48.5. 转换函数 60

48.6. 字符函数 60

48.7. 系统函数 60

48.8. 分析函数（高级统计函数 60

48.9. 条件判断函数 60

48.10. 加密函数 61

48.11. 格式化函数 61

# Atitit.提升稳定性之道

1. 前言 2

2. 为什么会发生稳定性问题 2

2.1. 单点故障(单点故障率较高) 2

2.2. 复杂 2

2.3. 资源耗尽 2

2.4. 死锁与等待 2

2.5. 崩溃 2

3. 第一章 资源耗尽 2

3.1. 资源占比过高 （ 2

3.2. 磁盘空间 磁盘空间不足,造成许多的莫名其妙的问题.也许提示连接耗尽.. 2

3.3. /io读写 3

3.4. /cpu/内存/ 3

3.5. 网络等占用过高） 3

3.6. 文件句柄数量 3

3.7. 线程数量 3

3.8. 句柄 网络链接数量 3

3.9. 非托管资源的释放 3

3.10. 病毒造成网络句柄耗尽 3

4. 第二章死锁与等待 3

4.1. 等待与超时 单点故障 单进程 单线程等 3

4.2. 网络死锁 3

4.3. 文件死锁，文件并发读写 3

4.4. 数据库连接死锁 3

4.5. 代码死锁 4

4.6. Cli命令行死锁 4

4.7. GUI界面死锁（文件格式错误，有确认框弹出等情况） 4

4.8. 类库冲突 4

4.9. 热更新热部署(业务可用性 4

4.10. 程序崩溃 4

4.11. Timeout 4

5. 程序崩溃 4

5.1. 解决方案：多进程 4

5.2. 进程隔离 4

5.3. Rest接口隔离 4

5.4. 守护进程 4

5.5. 类库冲突,造成部署问题 5

5.6. 热更新的支持不足,部署比较麻烦 5

5.7. Web服务跟数据库服务崩溃 5

5.8. 其他的潜在隐患: 5

5.9. 子线程异常造成主线程崩溃(java不影响,.net有这个问题) 5

5.10. 别的网络socket连接释放问题... 5

5.11. 直接内存读写 5

5.12. Stream的关闭释放. 5

5.13. native method调用的内存 5

6. #----解决方法 大方向 5

6.1. 杀毒 木马查杀软件 5

6.2. 适当冗余 故障集群 5

6.3. 守护模式开启 守护各大资源的监控 循环守护机制 6

6.4. 检测机制 定时检测主要服务问候 6

6.5. 重试机制(包括自动重连) 6

6.6. 奔溃日志总结 6

6.7. 尽可能的减少单点并发与读写 6

6.8. Gc自动释放资源 与超时自动释放 6

6.9. 热更新热部署) 更简化的开发架构(.. 6

6.10. 更好用的第三方框架类库 6

6.11. 最佳推荐流程( 6

6.12. 更简化的编程语言 6

6.13. 监测(线程 句柄),warnning,跟自动恢复 6

6.14. 压力测试 6

6.15. 预警机制 6

6.16. 超时回收资源gc 6

7. #----解决方法总结细则 7

7.1. 类库冲突避免(ide,检测工具,开发时,运行时) 7

7.2. 引擎+脚本结构(c++,java+python,lua,php等) 7

7.3. 语言级的新的特性 7

7.4. php/.net 7

7.5. 建立基于提升稳定性的内部封装框架/流程文档 7

7.6. Finalize/Dispose 7

7.7. 容错(包括自动重连) 7

7.8. SoftReference 7

7.9. 连接池的配置: 自动超时回收Connection+超时自动断开conn 7

7.10. 语句块回收资源/using块中自动调用Dispose 8

7.11. 崩溃时候儿core dump并且重启 8

7.12. 日志，缓存等文件，尽可能按时间生成多个文件。。 8

7.13. 重要业务服务和页面gui监测 8

7.14. 监测程序(cpu,内存占用, io队列深度, 磁盘空间,数据库连接数,数据库死锁监测) 8

7.15. 网络，文件操作使用wrap类库secury方式调用 8

7.16. 死锁自解除(数据库,文件等) 8

7.17. #----压力测试 8