<<C++ API设计>>

图书基本信息

书名: <<C++ API设计>>

13位ISBN编号: 9787115322999

10位ISBN编号:7115322996

出版时间:2013-8

出版时间:人民邮电出版社

作者:[美] Martin Reddy

译者:刘晓娜,臧秀涛,林健

版权说明:本站所提供下载的PDF图书仅提供预览和简介,请支持正版图书。

更多资源请访问:http://www.tushu007.com

<<C++ API设计>>

内容概要

现代软件开发中的一大难题就是如何编写优质的API。 API负责为某个组件提供逻辑接口并隐藏该模块的内部细节。 多数程序员依靠的是经验和冒险,从而很难达到健壮、高效、稳定、可扩展性强的要求。 Martin Reddy博士在自己多年经验基础之上,对于不同API风格与模式,总结出了API设计的种种最佳 策略,着重针对大规模长期开发项目,辅以翔实的代码范例,从而有助于设计决策的成功实施,以及 软件项目的健壮性及稳定性的实现。

主要内容:

API简介及其特点 API的一些设计模式及惯用法 影响API的设计的一些C++特性 把控API的风格与性能 版本控制与文档化的实现 各种优秀的测试方法 如何创建脚本绑定,以便API能被诸如Ruby和Python等语言调用 可扩展性API的基本实现方式 类库的编译器实现

<<C++ API设计>>

作者简介

作者简介:

Martin Reddy

博士是软件行业的一名老兵,有着15年以上的从业经验,共撰写过40多篇论文,拥有3项软件专利,并 与他人合著了Level of Detail for 3D Graphics。

另外,他还是ACM以及IEEE的会员。

早年,他曾在SRI International供职5年,主要从事分布式三维地形可视化技术方面的工作,他成功创建了在Web上描述3D地球空间信息模型的ISO标准,并且还连续两年被选为Web3D协会的会长。

他曾在Pixar动画工作室工作过6年,担任内部动画系统的首席工程师,设计并实现了很多高性能API, 这些API在一些奥斯卡获奖及提名影片的制作中都发挥了关键作用,这些影片有《海底总动员》、《 超人总动员》、《赛车总动员》、《料理鼠王》,以及《机器人总动员》等。

他还开办了一家咨询公司Code Reddy,为各家软件公司提供技术咨询,主要客户有Linden Lab和Planet 9 Studios,为大型在线3D虚拟世界《第二人生》设计了API并改善了其基本架构。

现在他担任ToyTalk公司的首席技术官。

译者简介:

刘晓娜

中国科学院计算技术研究所员工,在职博士,从事网络大数据采集及挖掘方面的研究,爱好读书、翻译和旅游。

臧秀涛

硕士毕业于中国科学院计算技术研究所。

曾从事网络游戏、操作系统方面的开发工作。

热爱技术、读书和翻译。

林健

中国科学院计算技术研究所博士生,从事机群与网格计算方面的研究,爱好DIY、开源软件与技术写作。

<<C++ API设计>>

书籍目录

目 录 第1章 API简介 1	
1.1 什么是API 1	
1.1.1 契约和承包人 2	
1.1.2 C++中的API 3	
1.2 API设计上有什么不同 4	
1.3 为什么使用API 5	
1.3.1 更健壮的代码 6	
1.3.2 代码复用 6	
1.3.3 并行开发 8	
1.4 何时应当避免使用API 9	
1.5 API示例 10	
1.5.1 API层次 10	
1.5.2 真实示例 12	
1.6 文件格式和网络协议 13	
1.7 关于本书 15	
第2章 特征 17	
2.1 问题域建模 17	
2.1.1 提供良好的抽象 17	
2.1.2 关键对象的建模 19	
2.2 隐藏实现细节 20	
2.2.1 物理隐藏:声明与定义 20	
2.2.2 逻辑隐藏: 封装 22	
2.2.3 隐藏成员变量 23	
2.2.4 隐藏实现方法 26	
2.2.5 隐藏实现类 28	
2.3 最小完备性 29	
2.3.1 不要过度承诺 29	
2.3.2 谨慎添加虚函数 30	
2.3.3 便捷API 31	
2.4 易用性 33	
2.4.1 可发现性 34	
2.4.2 不易误用 34	
2.4.3 一致性 36	
2.4.4 正交 38	
2.4.5 健壮的资源分配 40	
2.4.6 平台独立 43	
2.5 松耦合 44	
2.5.1 仅通过名字耦合 45	
2.5.2 降低类耦合 45	
2.5.3 刻意的冗余 47	
2.5.4 管理器类 48	
2.5.5 回调、观察者和通知 50	
2.6 稳定的、文档详细且经过测试的API	53
第3章 模式 54	

3.1	Pimpl惯用法 55	
3.1.1	使用Pimpl 56	
3.1.2	复制语义 59	
3.1.3	Pimpl与智能指针 60	
3.1.4	Pimpl的优点 61	
3.1.5		
3.1.6		62
3.2	单例 64	
3.2.1	在C++中实现单例 64	
3.2.2	使单例线程安全 66	
3.2.3	单例与依赖注入 68	
3.2.4	单例与单一状态 69	
3.2.5	单例与会话状态 71	
3.3	工厂模式 71	
3.3.1	抽象基类 72	
3.3.2		
	扩展工厂示例 74	
	API包装器模式 76	
3.4.1	代理模式 76	
3.4.2	适配器模式 79	
3.4.3	外观模式 81	
	观察者模式 83	
	MVC架构 83	
	实现观察者模式 84	
3.5.3		
	5 设计 88	
4.1		
	积累技术债 89	
	偿还技术债 90	
4.1.3		
	收集功能性需求 92	
	什么是功能性需求 93	ì
4.2.2		,
4.2.3	维护需求 94	
4.3	创建用例 95	
4.3.1		
	用例模板 95	
4.3.3		
4.3.4		
4.4		
	架构设计 102	
4.5.1		
	架构的约束 104	
4.5.2		
4.5.4		
4.5.4		
	来构模式 109 架构的交流 110	
- T.J.()		

4.6	类的设计 111	
4.6.1	面向对象概念 112	
4.6.2	类设计选项 113	
4.6.3	使用继承 113	
	Liskov替换原则 115	
4.6.5		
	迪米特法则 119	
4.6.7	类的命名 120	
4.7		
4.7.1		
	函数67 22 32 32 32 32 32 32 32 32 32 32 32 32	
	函数即日 122 函数参数 123	
	留奴多奴 123 错误处理 125	
	5 风格 129	
5.1		
5.1.1	• • • —	
	ANSI C API的优点 132	
5.1.3		
	从C++中调用C函数 134	
5.1.5		
5.2	面向对象的C++ API 136	
5.2.1	面向对象API的优点 136	
5.2.2	面向对象API的缺点 136	
5.2.3	案例研究:FMOD C++ API 137	
5.3	基于模板的API 138	
5.3.1	基于模板的API示例 138	
5.3.2	模板与宏 139	
5.3.3	基于模板的API的优点 140	
5.3.4	基于模板的API的缺点 141	
	数据驱动型API 141	
5.4.1		
	数据驱动型API的优点 143	
	数据驱动API的缺点 144	
	支持可变参数列表 144	
	案例研究:FMOD数据驱动型API	147
	- エアルカル・1 MOD 数	177
为 0与 6.1	- C++/h/2 149 命名空间 149	
6.2		
6.2.1		
	定义构造函数和赋值操作符 153	
6.2.3	•	
6.3		
6.3.1		
	参数的const正确性 157	
	返回值的const正确性 157	
	模板 158	
6.4.1	模板术语 158	

6.4.2	隐式实例化API设计	160	
6.4.3	显式实例化API设计	162	
6.5	操作符重载 164		
6.5.1	可重载的操作符	164	
6.5.2	自由操作符与成员摸	操作符	165
	为类添加操作符	166	
6.5.4	操作符语法 168		
	转换操作符 170		
6.6	函数参数 171		
6.6.1	指针与引用参数	171	
	默认参数 172		
	避免使用#define定义常		173
	避免使用友元 175	,	
	导出符号 176		
	编码规范 179		
	5 性能 181		
	通过const引用传递输。		182
	最小化#include依赖		
	避免"无所不包型"	' 头文件	184
	前置声明 184		
	冗余的#include警戒i	语句	186
	声明常量 188		
	初始化列表 190		
	内存优化 192		
	除非需要,勿用内联	196	
	写时复制 198		
7.8			
7.8.1	迭代器 202 随机访问 203		
	数组引用 204		
	性能分析 205		
7.9.1	时效性分析 205		
	基于内存的分析	207	
7.9.3	多线程分析 208		
	5 版本控制 209		
	版本号 209		
8.1.1	版本号的意义 20	09	
	小众的编号方案		
8.1.3	提供API的版本信息	211	
8.2	软件分支策略 213	}	
	分支策略 213		
8.2.2	分支方针 213		
	API和并行分支		
8.2.4	文件格式和并行发布	5产品	215
8.3	API的生命周期 21	16	
8.4	兼容性级别 217		
8.4.1	向后兼容性 217		

8.4.2	功能兼容性	218	3	
8.4.3	源代码兼容性	Ε :	218	
8.4.4	二进制兼容性	Ε :	219	
8.4.5		22	1	
8.5	怎样维护向后,	東容性	<u> </u>	222
8.5.1	添加功能	222		
	修改功能			
8.5.3	弃用功能	224		
8.5.4	移除功能	226		
8.6	API审查 22	6		
8.6.1	API审查的目的	的	226	
	API预发布审			
	API预提交审:		228	
第9章	过 文档 230			
9.1			230	
	定义行为			
9.1.2	为接口契约编	写文	档	232
	告知行为的改			
	文档涉及的内		234	
9.2	文档的类型	236		
	自动生成的A		当	237
9.2.2	概述文档	237		
9.2.3	示例和教程 发布说明	238	3	
9.2.4	发布说明	238		
9.2.5	授权信息			
9.3				
9.4	使用Doxygen	242	<u>-</u>	
9.4.1	配置文件	242		
9.4.2	注释风格和命	令	242	
9.4.3	API注释 2	243		
9.4.4	文件注释	245		
9.4.5	类注释 24	1 5		
9.4.6	方法注释	246		
9.4.7	枚举注释	247		
9.4.8	带有文档的示	例头	文件	247
第10章	章 测试 25	0		
10.1	编写测试的理	由	250	
10.2	API测试的类型	텐	252	
10.2.1	单元测试	253		
10.2.2	集成测试	255		
10.2.3	性能测试	257		
10.3	编写良好的测	试	259	
10.3.1	良好测试的物	诗征	25	9
10.3.2	测试对象	260		
10.3.3	关注测试工作	乍量	26	1
10.3.4	与QA一起工	作	261	
10.4	编写可测试的	代码	26	62

10.4.1	测试驱动开发 262	
10.4.2	桩对象和模拟对象 264	
10.4.3	测试私有代码 267	
10.4.4	使用断言 269	
10.4.5	契约编程 270	
	记录并重放功能 272	
10.4.7	支持国际化 273	
10.5	自动化测试工具 273	
10.5.1	自动化测试框架 274	
10.5.2	代码覆盖率 277	
10.5.3	缺陷跟踪系统 279	
10.5.4	持续构建系统 280	
	起脚本化 282	
11.1	添加脚本绑定 282	
11.1.1		
11.1.2	脚本化的优点 283	
	语言兼容性问题 284	
11.1.4	跨越语言障碍 285	
11.2	脚本绑定技术 286	
11.2.1	Boost Python 286	
11.2.2	SWIG 286	
11.2.3	Python-SIP 287	
11.2.4	COM自动化 287	
11.2.5		
11.3	使用Boost Python添加Python绑定	289
11.3 11.3.1	使用Boost Python添加Python绑定 构建Boost Python 290	
11.3 11.3.1 11.3.2	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API	
11.3 11.3.1 11.3.2 11.3.3	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292	
11.3 11.3.1 11.3.2 11.3.3 11.3.4	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8 11.4	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8 11.4 11.4.1	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8 11.4 11.4.1 11.4.2	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301 调整Ruby API 303	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8 11.4 11.4.1 11.4.2 11.4.3	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301 调整Ruby API 303 构造函数 304	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8 11.4 11.4.1 11.4.2 11.4.3 11.4.4	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301 调整Ruby API 303 构造函数 304 扩充Ruby API 304	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8 11.4 11.4.1 11.4.2 11.4.3 11.4.4 11.4.5	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301 调整Ruby API 303 构造函数 304 扩充Ruby API 304 C++中的继承 305	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8 11.4 11.4.1 11.4.2 11.4.3 11.4.4 11.4.5 11.4.6	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301 调整Ruby API 303 构造函数 304 扩充Ruby API 304 C++中的继承 305 跨语言多态 307	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.4.1 11.4.1 11.4.2 11.4.3 11.4.4 11.4.5 11.4.5 11.4.6 11.4.7	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301 调整Ruby API 303 构造函数 304 扩充Ruby API 304 C++中的继承 305 跨语言多态 307 综合应用 307	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8 11.4 11.4.1 11.4.2 11.4.3 11.4.4 11.4.5 11.4.5 11.4.7 第12章	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301 调整Ruby API 303 构造函数 304 扩充Ruby API 304 C++中的继承 305 跨语言多态 307 综合应用 307 综合应用 307	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8 11.4 11.4.1 11.4.2 11.4.3 11.4.4 11.4.5 11.4.5 11.4.7 第12章 12.1	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301 调整Ruby API 303 构造函数 304 扩充Ruby API 304 C++中的继承 305 跨语言多态 307 综合应用 307 综合应用 307 统一可扩展性 310 通过插件扩展 310	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8 11.4 11.4.1 11.4.2 11.4.3 11.4.4 11.4.5 11.4.5 11.4.7 第12章 12.1	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301 调整Ruby API 303 构造函数 304 扩充Ruby API 304 C++中的继承 305 跨语言多态 307 综合应用 307 适可扩展性 310 通过插件扩展 310 插件模型概览 311	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.3.8 11.4 11.4.1 11.4.2 11.4.3 11.4.4 11.4.5 11.4.5 11.4.5 11.4.7 第12.1 12.1.1 12.1.2	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301 调整Ruby API 303 构造函数 304 扩充Ruby API 305 跨语含应用 307 综合应用 307 综合应用 307 综合应用 310 通过插件扩展 310 通过插件扩展 310 插件模型概览 311 插件系统设计问题 313	
11.3 11.3.1 11.3.2 11.3.3 11.3.4 11.3.5 11.3.6 11.3.7 11.4.1 11.4.2 11.4.3 11.4.4 11.4.5 11.4.5 11.4.7 第12.1 12.1.1 12.1.1 12.1.2 12.1.3	使用Boost Python添加Python绑定 构建Boost Python 290 使用Boost Python包装C++ API 构造函数 292 扩充Python API 293 C++中的继承 295 跨语言多态 296 支持迭代器 298 综合应用 298 使用SWIG添加Ruby绑定 300 使用SWIG包装C++ API 301 调整Ruby API 303 构造函数 304 扩充Ruby API 304 C++中的继承 305 跨语言多态 307 综合应用 307 适可扩展性 310 通过插件扩展 310 插件模型概览 311	

12.1.5	插件:	示例	317	
12.1.6	插件	管理器	318	
12.1.7	插件	版本控制	刮 321	
12.2	通过继	承扩展	322	
12.2.1	添加	功能	322	
12.2.2	修改:	功能	323	
12.2.3	继承	与STL	324	
12.2.4	继承	与枚举	325	
12.2.5	访问:	者模式	326	
12.2.6	禁止	子类化	331	
12.3	通过模	板扩展	332	
12.3.1	基于	策略的	莫板 3	332
12.3.2	奇特	的递归	莫板模式	334
附录A	库	336		
参考文	で献	351		
索引	355			

<<C++ API设计>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介,请支持正版图书。

更多资源请访问:http://www.tushu007.com