目录

[第1章 CrossApp简介及开发环境搭建 2](#_Toc413689277)

[1.1 CrossApp简介 2](#_Toc413689278)

[1.1.1 CrossApp功能与特色 2](#_Toc413689279)

[1.1.2 CrossApp的优势 3](#_Toc413689280)

[1.1.3 官网介绍及技术支持 3](#_Toc413689281)

[1.2 CrossApp开发环境搭建 4](#_Toc413689282)

[1.2.1 Windows开发环境搭建 4](#_Toc413689283)

[1.2.2 Mac OS X开发环境搭建 4](#_Toc413689284)

[第2章 CrossApp基础概念 4](#_Toc413689285)

[2.1 架构和目录结构 4](#_Toc413689286)

[2.1.1 架构 4](#_Toc413689287)

[2.1.2 目录结构 4](#_Toc413689288)

[2.2 核心类 5](#_Toc413689289)

[2.3 内存管理 5](#_Toc413689290)

[2.4 坐标系 6](#_Toc413689291)

[2.5 适配方案 6](#_Toc413689292)

[第3章 CrossApp核心控件 6](#_Toc413689293)

[第4章 CrossApp数据解释与存储 6](#_Toc413689294)

[第5章 CrossApp设备功能调用 6](#_Toc413689295)

[第6章 CrossApp进阶 6](#_Toc413689296)

[第7章 CrossApp网络通信 6](#_Toc413689297)

[第8章 折800—实战讲解 6](#_Toc413689298)

[第9章 动漫之家—实战讲解 6](#_Toc413689299)

# CrossApp简介及开发环境搭建

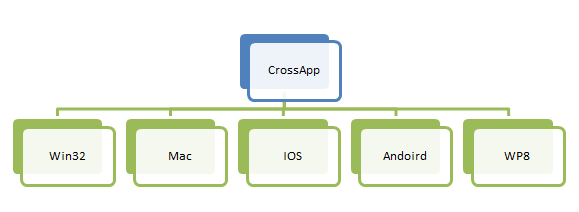
## CrossApp简介

 CrossApp是一款完全开源、免费、跨平台的移动应用开发引擎，基于最宽松的MIT开源协议，开发者根据自身情况使用CrossApp开发任何商业项目。CrossApp以C++作为开发语言，图形渲染基于OpenGL ES 2.0，采用MVC框架模式。使用CrossApp开发的应用程序支持导出到各大主流移动平台，真正实现"一次编码，多处运行"的跨平台开发技术。

CrossApp主要由9秒社团自研，官方制定了CrossApp的基本架构，确立了CrossApp的基本雏形，而其后的版本也将由来自各方的开发精英自愿组成的9秒社团常务贡献委员会共同进行更新和维护。

### CrossApp功能与特色

**跨平台**



CrossApp暂时主要支持目前最主流的移动平台IOS和Android，后续更新版本将完善更多平台的支持。

**整合第三方库**

CrossApp整合一些第三方库，例如常见的数据解析库jsoncpp、tinyxml、http等。

**基于OpenGL ES 2.0**

CrossApp的图形渲染使用的是Open GL ES 2.0，渲染效率高，可以使移动设备的GPU发挥到最佳效果。

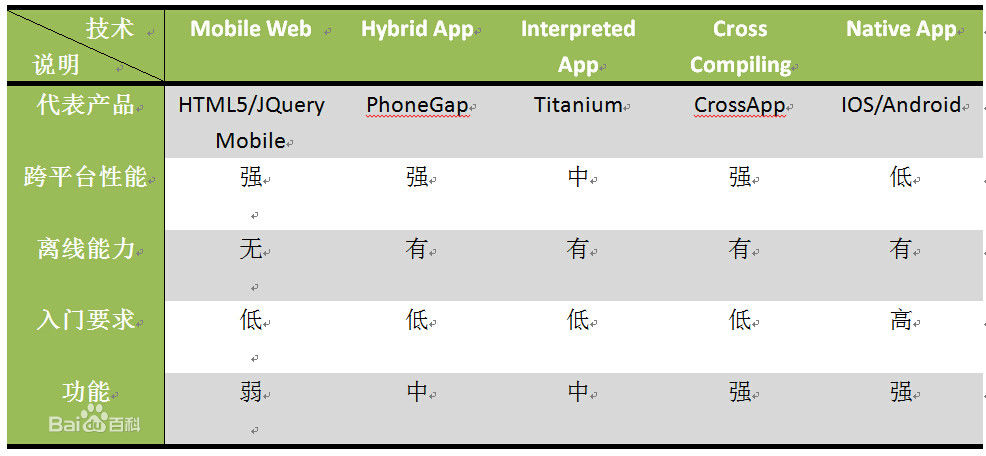
**丰富的UI控件**

CrossApp的设计宗旨在于为移动应用开发者提供快速、高效的开发解决方案。在此基础上，CrossApp封装了大量的UI控件，各类控件的功能十分丰富，开发者可以直接使用这些控件进行应用的开发。这些控件基本满足应用开发需求中的大部分功能，也免去了自己封装UI控件花费的大量时间，进一步提升开发的效率、节省开发的成本。同时CrossApp也会整合部分由第三方开发者提供的优秀控件资源，更大的丰富CrossApp的UI控件。

**CrossApp耗电优化**

由于之前CrossApp抽取了cocos2d-x的渲染驱动模式，程序生命周期中，在不断重绘，渲染驱动模式的缺点就是CPU占用高因此耗电。显而易见，这种不断的重绘方式对于游戏可能影响不大，但对于APP来说太浪费了。因此，我们针对应用程序的特性，将渲染机制改为事件驱动模式。这种模式的渲染要有外界触发才会重绘，在没有外界触发的时候画面静止，渲染停止，以达到节能的效果。

### CrossApp的优势



**通过上表分析我们可以得出：**

**1.Mobile Web**

以HTML5和JQuery为代表的web开发技术是以纯浏览器为基础的，所以没有离线能力可言，更无法充分发挥各平台的功能和特性。

**2.Hybrid App**

Hybrid App即混合模式应用程序，是指介于web和native直接的app，具备一定Native原生App优秀的用户体验和Web App跨平台的优势。但由于Hybrid仍旧以前端技术为基础，无法对内存和系统资源进行有效的管理。

**3.Interpreted App**

以前端技术为基础，同样不能有效对内存和系统资源进行管理。

**4.Native App**

原生开发以iOS和Android为代表，虽然原生开发能够实现最佳的用户体验和高优化，但开发的成本较高。而且因为原生开发几乎不支持跨平台特性，相对于跨平台技术来说，带来的实际效益更低。

综上所述，还是Native App和Cross compiling的综合效率更高，开发者需要根据实际情况，考虑各方面因素来选择合适的开发解决方案。

### 官网介绍及技术支持

## CrossApp开发环境搭建

### Windows开发环境搭建

### Mac OS X开发环境搭建

# CrossApp基础概念

## 架构和目录结构

### 架构

### 目录结构

从github下载CrossApp资源包后，或解压稳定版ZIP压缩包后，就会看到如下的目录结构：

|  |  |
| --- | --- |
| **目 录** | **说 明** |
| **CrossApp** | 此目录为引擎的源码目录，其主要的两个目录  **CrossApp.h:**  所有引擎功能需要包含的头文件，对使用CrossApp开发应用的开发者，屏蔽了底层的系统差异，在需要跨平台的情况下，这个目录下的头文件会包含CrossApp\platform目录下对应的头文件。  **CrossApp\platform:**  用一堆#if宏判断当前操作系统，不同系统include相应系统下的头文件。CrossApp基于OPENGL ES（OPENGL的子集，专用于移动设备），所以底层的图像和动画绘制已经是跨平台了，所以cocos2d中真正需要跨平台的不多（platform下的头文件并不多），一些像消息循环响应机制等不同平台有不同方式的才需要跨平台 |
| **CocosDenshion** | 音频库 |
| **extensions** | json、http、sqlite3库等第三方库 |
| **licenses** | 本引擎依赖了很多的开源工程，他们的所有许可声明都在这里了。 |
| **scripting** | 支持的脚本 |
| **template** | 包含在不同IDE和平台下来创建CrossApp工程的模板 |
| **projects** | 工程目录(创建工程后才出现此目录，默认无) |
| **samples** | 示例Demo |
| **tools** | 各种脚本处理文件、包含工程创建等 |

## 核心类

### CAObject

CAObject是CrossApp引擎的根类，主要是进行内存的管理，定义部分回调函数指针。引擎绝大部分类派生自CAObject，其共同遵从同一套内存管理方式。

CAObject主要的函数有:

|  |  |
| --- | --- |
| **函数** | **说明** |
| **void** retain(void) | 对象计数器+1 |
| **void** release(void) | 对象计数器-1 |
| **void** autorelease(void) | 设置内存自动管理 |
| **unsigned int** retainCount(void) | 返回当前对象计数器的值 |
| **bool** isSingleReference(void) | 实例化对象是否只有一个使用者 |
| **bool** isEqual(**const CAObject** \*pObject) | 与指定object实例对象是否相同 |

## 内存管理

## 坐标系

## 适配方案

# CrossApp核心控件

# CrossApp数据存储与解析

在开发app时，我们需要对数据持久保持和解析，CrossApp也对常见的数据类型解析提供了支持如json、xml、sqlite3等。

## CAUserDefault简单存储

在CrossApp中提供了自带的存储类：CAUserDefault,适合存储数据量比较小，结构比较简单的数据。如果你需要存储大量的复杂数据，建议使用SQlite比较合适。

**先介绍一下CAUserDefault的API:**

|  |  |
| --- | --- |
| **方法名** | **说明** |
| **void** setBoolForKey(**const char\*** pkey,**bool** valuer) | 根据pkey存储一个bool类型 |
| **void** setIntegerForKey(**const char\*** pkey,**int** valuer) | 根据pkey存储一个int类型 |
| **void** setFloatForKey(**const char\*** pkey,**float** valuer) | 根据pkey存储一个float类型 |
| **void** setDoubleForKey(**const char\*** pkey,**double** valuer) | 根据pkey存储一个double类型 |
| **void** setStringForKey(**const char\*** pkey, **const std::string &** value) | 根据pkey存储一个string类型 |
| **bool** getBoolForKey(**const char\*** pKey) | 根据pkey读取相应的值，如果没有在UserDefault.xml找出则返回false |
| **bool** getBoolForKey(**const char\*** pKey,**bool** defaultValue) | 根据pkey读取相应的值，如果没有在UserDefault.xml找出则返回defaultValue |
| **int** getIntegerForKey**(const char\*** pKey**)** | 根据pkey读取相应的值，如果没有在UserDefault.xml找出则返回0 |
| **int** getIntegerForKey**(const char\*** pKey**, int** defaultValue**)** | 根据pkey读取相应的值，如果没有在UserDefault.xml找出则返回defaultValue |
| **float** getFloatForKey**(const char\*** pKey**)** | 根据pkey读取相应的值，如果没有在UserDefault.xml找出则返回0.0f |
| **float** getFloatForKey**(const char\*** pKey**,float** defaultValue**)** | 根据pkey读取相应的值，如果没有在UserDefault.xml找出则返回defaultValue |
| **double** getDoubleForKey**(const char\*** pKey**)** | 根据pkey读取相应的值，如果没有在UserDefault.xml找出则返回0.0 |
| **double** getDoubleForKey**(const cha \*** pKey**,double** defaultValue**)** | 根据pkey读取相应的值，如果没有在UserDefault.xml找出则返回defaultValue |
| **void** flush() | 存储到UserDefault.xml，不写的话不会存入 |
| **const string&** getXMLFilePath() | 获得UserDefault.xml的存储路径 |

**存储代码如下:**

//存储名字

CAUserDefault::sharedUserDefault()->setStringForKey("name", "zero");

//存储年龄

CAUserDefault::sharedUserDefault()->setIntegerForKey("age", 25);

//存储身高

CAUserDefault::sharedUserDefault()->setDoubleForKey("height",1.75);

//存储体重

CAUserDefault::sharedUserDefault()->setFloatForKey("wight", 75.0f);

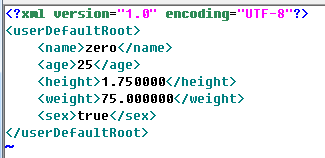
//存储性别 true为男 fasle 女

CAUserDefault::sharedUserDefault()->setBoolForKey("sex", true);

//这里一定要提交写入哦，否则不会记录到xml中，下次启动游戏你就获取不到value了

CAUserDefault::sharedUserDefault()->flush();

**我们将看到UserDefault.xml文件如下:**



**读取示例代码:**

//读取name

std::string name = CAUserDefault::sharedUserDefault()->getStringForKey("name");

//读取Email,如果没有Email则返回"default"

std::string email = CAUserDefault::sharedUserDefault()->getStringForKey("Email","default");

//读取age

int age = CAUserDefault::sharedUserDefault()->getIntegerForKey("age");

//读取id,如果没有id则返回10000

int id = CAUserDefault::sharedUserDefault()->getIntegerForKey("id", 10000);

//读取身高体重，如果没有找到则返回0

float height =CAUserDefault::sharedUserDefault()->getFloatForKey("height");

double weight =CAUserDefault::sharedUserDefault()->getDoubleForKey("weight");

//读取sex，如果没有找到sex则返回true

bool sex = CAUserDefault::sharedUserDefault()->getBoolForKey("false", true);

## SQLite的使用

在CrossApp中，简单数据存储，可以使用CAUserDefault。那么如何存储大量，不规则的数据？我们可以使用SQLite数据库存储数据。SQLite是使用非常广泛的嵌入式数据库，它有小巧 、高效、跨平台、开源免费和易操作的特点。

SQLite数据库是使用C语言来编写的,在CrossApp中使用也是非常容易的。

CrossApp已经添加了SQlite的，在CrossApp\extensions\sqlite3目录，我直接使用就可以了。

**引入头文件**

#include "CrossAppExt.h"

**创建数据库**

//数据库指针

sqlite3 \*pdb=NULL;

//保存数据库的路径

std::string path= FileUtils::getInstance()->getWritablePath()+"save.db";

std::string sql;

int result;

//打开一个数据，如果该数据库不存在，则创建一个新的数据库文件

result=sqlite3\_open(path.c\_str(),&pdb);

if(result!=SQLITE\_OK)

{

log("open database failed, number%d",result);

}

**SQL语句**

//创建数据库表的sql语句

sql="create table student(ID integer primary key autoincrement,name text,sex text)";

**创建Talbe**

**插入**

**查询**

**删除**

**保存路径**

## Json解析

## XML解析

# CrossApp设备功能调用

# CrossApp进阶

# CrossApp网络通信

# 折800—实战讲解

# 动漫之家—实战讲解