hatalyst 源代码分析

尹飞 [1]

[1] E-mail: manage@icebirds.net

0.1 本书许可

目录

Part I

基本结构分析

第一章　准备工作

1.1　什么是 haXe

haXe（读做 heks[①]）是一款开源的多平台编程语言（作者注：个人觉得称其为多输出编程语言更合适一些），不同于传统的编程语言，haXe 并不生成自己独立的字节码（如 Java），也不直接生成本地代码[②]（如 C++，不过未来有可能会被加入直接生成本地代码的功能），而是编译生成现有的字节码或程序语言源码。其所生成的每个种类的源码或字节码被称为编译目标。

目前 haXe 可以编译为以下目标：

1. **Javascript**：haXe 可以编译为一个单独的.js 文件，并嵌入 html 中执行。由于在编译时会解决所有的平台相关的问题，因此可以通过统一的接口方便地访问浏览器的 DOM 树。

2. **Flash**：你可以把 haXe 代码编译为.swf 文件，通过对 AS 2 和 AS 3 的 API 的分别支持，haXe 可以编译输出针对 Flash player 6-10.2 的任意版本 swf 程序。haXe 通过在编译时对 AVM 字节码的优化以及增加了一些更底层的编译支持，haXe 生成的.swf 文件可以获得比 Adobe Flash 平台编译程序略高的执行效率以及更好的代码体验。

① 官方网站给出的读音读法为“hex”，但这个注音方式在不同的语言背景下差异极大，部分中文用户甚至将其读为“哈希”、“海西”这样的读音，为了避免读音歧义，这里采用的是国际音标注音的方式，中文读者可以近似地认为其读音接近“嗨克斯”。

② 如有需要，开发人员可以给 haXe 增加本地码编译的目标，但目前 haXe 并没有这个功能。

3. **NekoVM**：haXe 可以编译为 NekoVM 字节码（NekoVM 是一种类 JVM 的轻量级虚拟机，采用 Neko 编程语言编写程序并生成字节码），可以用于 web 服务器或 socket 服务器的开发，亦可通过扩展用于桌面开发。

4. **PHP**：haXe 代码可以编译为.php 文件，其编译结果接近一个超轻量级的框架。通过对外部代码库的支持，可以很方便地将现有的 php 库用于你的 haXe 项目。

5. **C++**：haXe 代码可以生成 C++ 代码，并调用不同的 C++ 编译器编译为本地程序。这个编译目标在希望将你的程序生成某些本地程序（如 Iphone 应用）时非常有用。

6. 在本书编写时，有消息称 C# 和 Java 两种编译目标也即将发布。

1.2 haXe 可以做什么？

通过上一章的内容，我们知道了 haXe 可以生成的各种程序，而从其生成的程序中，也可以看出 haXe 的用途。haXe 最初被设计用来主要进行 Web 应用的开发，通过对 neko 和 Javascript 的生成支持，设计者希望得到一种可以同时开发客户端和服务器端应用的语言；在不久后，就增加了对 flash 程序输出的支持，随后又增加了对 php 的输出支持，此时，虽然通过 neko 的一些 API，可以实现用 neko 开发一些简单的桌面应用，但其主要目标仍然是 web 应用。

1.3 为什么使用 haXe？

在官方网站 haxe.org 上列举了 5 条使用 haXe 的理由[①]：

1. 同时用于客户端、服务器端及桌面开发的 ECMA 风格编程

2. 十分快速的编译

① 参见 http://haxe.org/doc/why。

3. 高效的类型检查机制

4. haXe 为相应的平台增加了更多的语言特性

5. 开放源代码

不过，这些理由并不足以说服人们使用 haXe。在实际使用中，人们选择一种编程语言的理由可谓千奇百怪，有的人是因为公司要求使用，有的是因为自己熟悉，还有的仅仅是因为“我喜欢”。更有甚者，部分人选择 haXe 的理由是“因为 haXe 编译 Flash 平台的程序执行效率比较高”，事实上，由于 Flash 更多地被用于多媒体视觉程序的开发，而这一部分代码的效率受限于 Flashplayer，导致 haXe 其实并不能使其 Flash 平台的程序执行效率有显著提高[①]。这也使得这个理由看起来多少有些自欺欺人。

那么，究竟为什么还要使用 haXe 呢？本书并不准备说服任何人脱离他原有知识范围的怀抱，改用 haXe。之所以用这一本书来介绍 haXe 的开发，只是希望能够提供给大家更多的选择。

1.4 安装 haXe

接下来，我们就要开始 haXe 的安装了。

首先，在 haXe 的官方网站（http://haxe.org/download）下载 haXe 的安装包，注意选择你所使用的平台。目前官网提供 Windows、MAC 和 Linux 的安装包。如果你使用 FreeBSD 等操作系统，可以按照页面上的指引手动编译 haXe。

下载后，Windows 平台直接双击下载的安装包文件，按照提示安装即可；Linux 平台下，请打开终端，转到安装包所在目录，执行以下命令[②]：

```
$sudo ./hxinst-linux
```

① 编者注：由于 haXe 在 Flash9 以上平台目标的编译中对字节码进行了大量优化，并且由于 format 等库的存在，的确可以让开发人员对数值计算过程和调用过程中的消耗降低。但 Flash 到目前为止，其最主要的运用方向仍然是包括游戏开发在内的显示编程，较大规模的数值计算应用仍然较少，因此 haXe 实际上在 Flash 平台的效率优化并不显著。特殊的例外情况是三维引擎的开发。

② Ubuntu 等发行版的软件源中虽然有 haXe，但往往版本更新有所延迟，因此建议 Linux 用户下载安装包安装。

同样按照提示安装即可。MAC 平台做法与上述类似。如果以前安装过 haXe，也可以通过上述方法更新。

本书中后面的章节还会使用一些 haXe 的库，因此在这里需要配置一下 haXe 的库路径。Windows 用户在安装好后就已经配置好了 haXe 的库路径[①]，可以跳过这一步[②]。

Linux 用户和 MAC 用户可以在终端下运行以下命令进行 haXe 库路径的配置。

```
$haxelib setup
Please enter haxelib repository path with write access
Path: ~/haxelib
```

然后输入一个准备好的空目录路径即可。如上面的 /haxelib。

1.5 选择一个编辑器

haXe 安装好后，其中包含了 haXe 的编译器、haxelib 库配置工具、haxedoc 文档生成工具等，但并没有包含一个合适的编辑器。因此我们还需要选择一个合适的编辑器以用来编辑 haXe。在 haXe 的官方网站上提供了大量的常见编辑器配置文件（http://haxe.org/com/ide），可以自行选择自己喜欢的编辑器。在本书中，由于篇幅所限，不能提供所有编辑器的安装和配置介绍。在本书中，我们推荐使用开源的编辑软件 Gedit。下面介绍 Gedit 的配置方法。

Gedit 是 Gnome 项目的一个子项目。目前有 Windows、Linux 和 MAC 版本。Linux 下 Gnome 环境默认即有 Gedit 软件，如采用 KDE 等桌面环境，可以自行在软件源中安装。Windows 版本的下载地址如下[③]：

① Windows 下 haXe 的默认库目录是 C:\Programs Files\Motion-Twin\haxe\lib。但 Windows 7 下由于系统增加了大量的访问限制，导致此目录不可用，因此默认的库目录变成了 C:\Motion-Twin\haxe\lib，关于此改动的详情请参见 http://haxe.org/download。

② Windows 一般不必配置 haxelib 的库路径，但由于 Windows 下库路径默认被放在系统分区中，因此一旦重新安装操作系统，就需要重新安装和配置库。而 Linux 下由于可以把库全部安装在用户目录中，所以重新安装后只需要将库路径重新设置即可。如 Windows 用户希望重装系统后比较方便，也可配置库路径。

③ 由于 MAC 下有更好的 TextMate 编辑器，因此请直接下载使用 TextMate 编辑器的语

```
Windows 版本:
http://ftp.gnome.org/pub/gnome/binaries/win32/gedit/
```

请自行选择最新版本下载安装。

安装好 Gedit 软件后，需要该软件其进行配置方可使用。在本书的附件中 tools/gedit config 目录下有 Gedit 的 haxe 高亮语法文件，linux 用户请将其复制到/usr/share/gtksourceview-2.0/language-specs 目录下（需要用 root 权限，或者在命令行中用 sudo 授予权限）。Windows 用户请复制到 Gedit 安装目录下的 share\gtksourceview-2.0\language-specs 目录下。

接下来，我们把 Gedit 配置得更像一个 IDE，以方便我们在开发中的使用。

选择菜单中的编辑 -首选项，先打开“插件”选项卡，在插件列表中点击右键，选择“全部激活”。

接下来切换到“字体和颜色”选项卡，配色方案选择“Oblivion”。

之后切换到“编辑器”选项卡，将跳格宽度设置为 8，去掉“插入空格代替制表符”的勾选；勾选“启用自动缩进”；接着取消本选项卡后面所有选项的勾选。

最后切换回“查看”选项卡，除“显示右边距”选项外，其他选项全部勾选。点击确定关闭首选项。

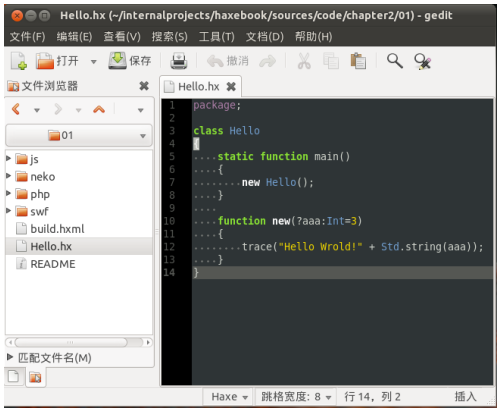

图 1.1: 配置好的 Gedit

配置完毕后，在“查看”菜单中，勾选侧边栏选项，即完成了 Gedit 的配置。配置后的 Gedit 软件如图1.5所示。

法文件，在此不再介绍 MAC 平台下的配置。

第二章　初识 haXe

2.1　万能的 Hello World

许多的编程书籍或者教程，都把这个没什么大用的 Hello world 作为第一课。我觉得这是一个很有趣的事情，我们的第一课，就是这个万能的 Hello world 程序。

首先，把下面的源码用 Hello.hx 这个文件名保存到一个单独的目录里：

```
1  package;
2
3  class Hello
4  {
5      static function main()
6      {
7          new Hello();
8      }
9
10     //构造函数
11     function new(?aaa:Int=3)
12     {
13         trace("Hello Wrold!" + Std.string(aaa));
14     }
15 }
```

之所以说它是万能的 Hello world，不仅仅是因为其他书里也有这个，也是因为，这个东西几乎可以编译为 haXe 的任何目标！

下面是编译文件，保存到和上面的文件同一目录下，命名为 Build.hxml：

```
#生成 neko
-neko neko/hello.n
-main Hello

#生成 php
--next
-php php
-main Hello

#生成 javascript
--next
-js js/hello.js
-main Hello

#生成 swf
--next
-swf swf/hello.swf
-swf-version 9
-swf-header 600:400:30:FF0000
-main Hello
```

上面的文件并没有编译以下两种格式：swf8 和 C++，但如果愿意，也可以编译出来的。swf8 即 AS 2 程序没有编译是因为个人觉得已经没有什么必要，AS 2 相对简单，开发也快捷的多，还不如直接学习使用 AS 2 来的方便。而 C++ 没有编译则是因为必须安装 C++ 编译器，windows 下必须是 VC，或者采用免费的 VC express 版，而且还可能会出现很多问题，必须把环境变量配置好了才能使用（其实原因是 VC 的命令行编译工具所依赖的一大堆库并没有配置环境变量，所以运行时找不到）。Linux 下则需要安装 gcc 和相关的包（ubuntu 下比较简单，直接 sudo apt-get install build-essential 安装完毕即可）。至于 MAC，需要什么我并不清楚，因为我不用 MAC。

如果你的条件具备，可以把以下代码添加到上面的 build.hxml 文件末尾，用于输出 C++。

```
21 #生成 cpp
22 --next
23 -cpp cpp
24 -main Hello
```

将这两个文件全都保存好后。打开命令行窗口（在 windows 下就是 DOS 窗口，点开始 -运行，然后输入 cmd 出来的那个），切换到你保存上面源码的目录，输入：`haxe build.hxml`，然后回车，就可以编译出一大堆的目录和文件了。在 windows 下也可以直接双击 build.hxml 文件编译。

编译结果应该是下面的情况：

1. neko 目录下有一个 hello.n 文件，在命令行窗口中进入这个目录，执行 neko hello.n 命令，就能看到运行结果 Hello.hx: 12 : Hello world!

2. php 目录下会有一个 index.php 文件和一个 lib 目录，把它们拷贝到 apache 或者 lighthttpd 的虚拟目录下，在浏览器中打开 http://localhost/ 就能看到输出结果。

3. js 目录下会有一个 hello.js 文件，把下面的代码保存到这个目录中，双击打开它就能看到结果。

```
1 <html>
2 <head><title>haXe JS</title></head>
3 <body>
4
5 <div id="haxe:trace"></div>
6 <script type="text/javascript"  src="hello.js">
7 </script>
8
9 </body>
10 </html>
```

4. swf 目录下会有一个 hello.swf 文件，用 flashplayer 打开它，就能看到运行结果。

5. 如果你配置了 C++ 编译器，并且正确编译好后，应该能再 cpp 目录下找到一个叫做 Hello.exe(windows 下) 或者 Hello(linux 下) 的文件。windows 下在命令行中进入这个目录执行 Hello.exe 就可以看到结果，Linux 下在命令行下进入这个目录执行./Hello 即可。

这个编译结果是不是很神奇？下面我们来分析一下这段神奇的代码：

首先，我们先来看一下 Hello.hx 这个文件，它是这段程序的源码。在第一行，我们首先声明了这个程序在默认的根包中（有关包的概念我们会在后文中详细讲解），接下来，我们声明了一个叫做 Hello 的类，花括号中就是这个类的详细代码。接着，我们为这个类定义了一个带有 static 标识的方法 main，表示它是一个静态方法，它是整个程序的唯一入口，在程序编译的时候会用到这个方法。main 方法中，我们用一行代码让程序执行 Hello 类的构造函数，就是后面定义的 new 方法，于是程序的执行过程跳转到了 new 方法中，在 new 方法中，我们调用了一个系统函数 trace，这个函数的功能是用来输出显示一些我们在程序执行过程中需要观察的信息。以方便我们确认程序的执行情况。在这里，它仅仅是输出了“Hello.hx: 12 : Hello world!”这样一段信息。整个程序就只是做了这些事情。

接下来，我们来看一下，这段源码是怎么变成好几种在不同环境下执行的程序的。将程序源代码转换为执行代码的过程，我们称之为编译，而执行编译这个任务的程序，就被称作编译器。在之前我们执行的 haxe build.hxml 这个命令，就是调用 haXe 的编译器，并且把 build.hxml 做为参数，让编译器处理。在 build.hxml 文件中，一共有四个部分，第一部分让 haXe 编译器生成一个 nekovm 的字节码程序。位置是在 neko 目录中，名字叫 hello.n。这个字节码程序可以被 neko 虚拟机打开执行。其余几部分分别是生成 Flash、PHP、Javascript 程序代码的。其中的“--next”参数代表让 haXe 编译器另外再生成另一个程序，这样我们就可以在一个工程中一次性用一个命令生成多个不同的程序。其他的参数大同小异，只有最后第 18 行的 -swf-version 9 参数，它表示生成的 swf 文件版本为 Flash player 9.0 或更高版本的播放器使用。

2.2 程序的基本结构

在上一节，我们写了一个没有什么用处的 Hello world 程序，从这一节开始，我们将以此为基础逐步深入，带你逐步深入了解 haXe 的世界。

第三章　面向对象编程

3.1　对象

Part II

进阶篇

第四章　计算机编程导论

4.1　写在本章前面的话

在本书的序中，曾提到学习编程所应当具备的一些条件，以及其他的一些说明，没有阅读序言的读者朋友希望能够阅读一下。在这里，再详述一下这些条件，不论学习哪方面的程序开发，这些都非常重要！

学习编程，首先我们需要具备的知识，并不是什么计算机的基础理论、原理这些东西，而是对数学的理解能力、抽象思维能力、准确的逻辑思维能力、自主的学习能力、对各种知识的快速过滤和查找的能力，以及英语；当然最重要的，还有肯吃苦努力的精神。后者就不必说了，前面提到的各项，将在下面详细说明，希望读者仔细阅读，并仔细思考自己是否愿意去锻炼和学习这些知识以及能力。

4.2　什么是程序

4.3　关于数据

4.4　算法如何体现

4.5　数据和算法究竟是怎么结合的

4.6　一些有用的参考