chromium之content_shell源代码分析（一） - flyonok的专栏 - 博客频道

分类：
chromium（22）

目录(?)[+]

介绍和入口函数

为了测试chromium和webkit的兼容性，chromium project 中有一个content shell，相对整个项目来说，它简单了很多，如果运行的化，就是一个简单的windows窗口程序，有利于程序员测试webkit的最新特征，和掌握html5等相关的新功能。其入口点在content/shell目录下的shell_main.cc（对于windows系统是如此）,打开该文件，可以看到如下代码：

```
#if defined(OS_WIN)
int APIENTRY wWinMain(HINSTANCE instance, HINSTANCE, wchar_t*, int) {
  sandbox::SandboxInterfaceInfo sandbox_info = {0};
  content::InitializeSandboxInfo(&sandbox_info);
  content::ShellMainDelegate delegate;
  return content::ContentMain(instance, &sandbox_info, &delegate);
}
#else
int main(int argc, const char** argv) {
#if defined(OS_MACOSX)
  // Do the delegate work in shell_content_main to avoid having to export the
  // delegate types.
  return ::ContentMain(argc, argv);
#else
  content::ShellMainDelegate delegate;
  return content::ContentMain(argc, argv, &delegate);
#endif  // OS_MACOSX
}
#endif  // OS_POSIX
```

和chrome工程的区别

可以看到它和chrome/app/chrome_exe_main_win.cc中的：

```
int APIENTRY wWinMain(HINSTANCE instance, HINSTANCE prev, wchar_t*, int) {
  // Initialize the commandline singleton from the environment.
```

```
  CommandLine::Init(0, NULL);
  // The exit manager is in charge of calling the dtors of singletons.
  base::AtExitManager exit_manager;
  MetroDriver metro_driver;
  if (metro_driver.in_metro_mode())
    return metro_driver.RunInMetro(instance, &RunChrome);
  // Not in metro mode, proceed as normal.
  return RunChrome(instance);
}
```

的不同之处：chrome中的main delegate是ChromeMainDelegate，content shell中的main delegate 是ShellMainDelegate，是content::ContentMainDelegate的不同子类实现版本，这样的设计就决定了上层界面有很大的不同，相对chrome来说，content shell的界面功能简单了很多，有利于大家理解浏览器的程序结构；

进入MainRunner

进入到Content_main.cc中的如下代码后：

```
// Main routine for running as the Browser process.
int ShellBrowserMain(const content::MainFunctionParams& parameters) {
  scoped_ptr<content::BrowserMainRunner> main_runner_(
      content::BrowserMainRunner::Create());
  int exit_code = main_runner_->Initialize(parameters);
  if (exit_code >= 0)
    return exit_code;
  if (CommandLine::ForCurrentProcess()->HasSwitch(
        switches::kCheckLayoutTestSysDeps)) {
    MessageLoop::current()->PostTask(FROM_HERE, MessageLoop::QuitClosure());
    main_runner_->Run();
    main_runner_->Shutdown();
    return 0;
  }
  bool layout_test_mode =
      CommandLine::ForCurrentProcess()->HasSwitch(switches::kDumpRenderTree);
  if (layout_test_mode) {
    content::WebKitTestController test_controller;
    std::string test_string;
    CommandLine::StringVector args =
        CommandLine::ForCurrentProcess()->GetArgs();
    size_t command_line_position = 0;
    bool ran_at_least_once = false;
#if defined(OS_ANDROID)
```

```
25.     std::cout << "#READY\n";
26.     std::cout.flush();
27. #endif
28.     while (GetNextTest(args, &command_line_position, &test_string)) {
29.       if (test_string.empty())
30.         continue;
31.       if (test_string == "QUIT")
32.         break;
33.       bool enable_pixel_dumps;
34.       std::string pixel_hash;
35.       FilePath cwd;
36.       GURL test_url = GetURLForLayoutTest(
37.           test_string, &cwd, &enable_pixel_dumps, &pixel_hash);
38.       if (!content::WebKitTestController::Get()->PrepareForLayoutTest(
39.               test_url, cwd, enable_pixel_dumps, pixel_hash)) {
40.         break;
41.       }
42.       ran_at_least_once = true;
43.       main_runner_->Run();
44.       if (!content::WebKitTestController::Get()->ResetAfterLayoutTest())
45.         break;
46.     }
47.     if (!ran_at_least_once) {
48.       MessageLoop::current()->PostTask(FROM_HERE, MessageLoop::QuitClosure());
49.       main_runner_->Run();
50.     }
51.     exit_code = 0;
52.   } else {
53.     exit_code = main_runner_->Run();
54.   }
55.   main_runner_->Shutdown();
56.   return exit_code;
57. }
```

进入MainLoop

这样的话，还是会进入到Browser_main_runner.cc的BrowserMainRunnerImpl类中，并呼叫其实列的Initialize函数和Run函数；在BrowserMainRunnerImpl的Initialize中会看到如下代码：

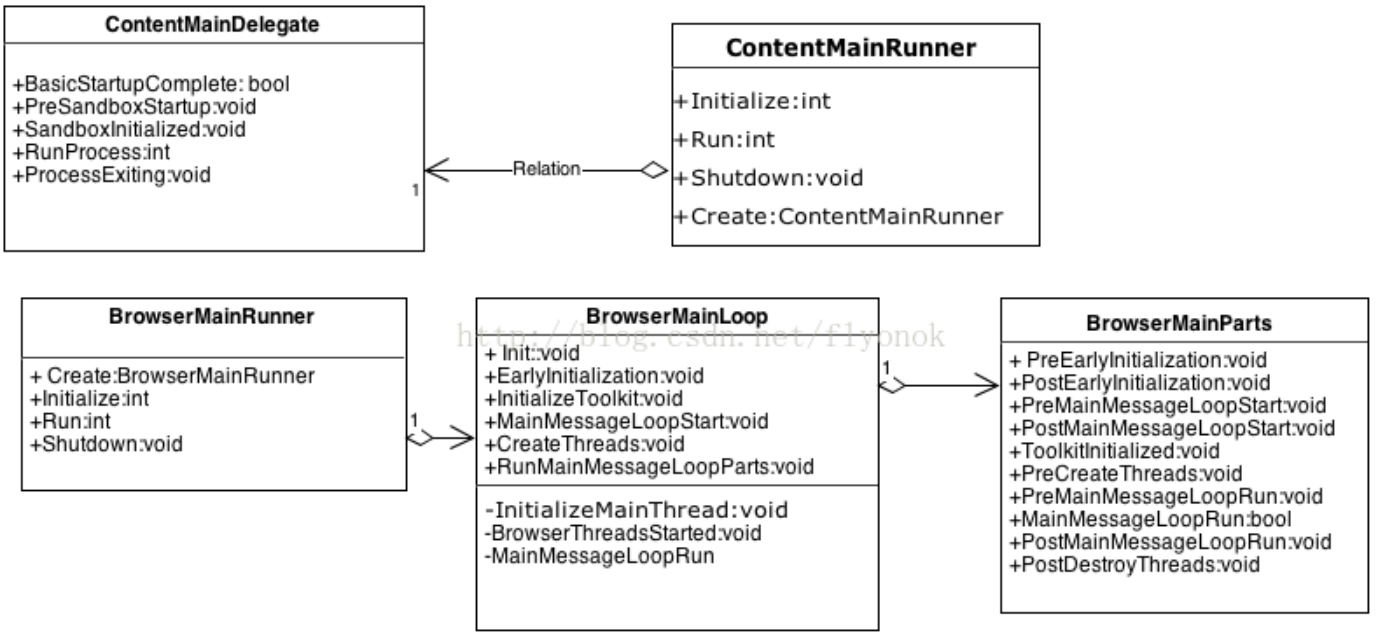
parts_的定义如下：

函数GetContentClient()在content/public/common/content_client.cc中定义如下：

主要类图

理解了以上过程，对解释BrowserMainLoop的运行过程有帮助。

下面我把content_shell启动过程中涉及的几个主要类图贴上来，希望对理解有帮助



ContentMainDelegate类的子类有ChromeMainDelegate和ShellMainDelegate等，其实例一般通过Conten::Main()传入，我们一般关注BasicStartupComplete和RunProcess两个接口，负责浏览器的系统初始化和进入浏览器主循环的入口。

（完）