# Coredns启动流程

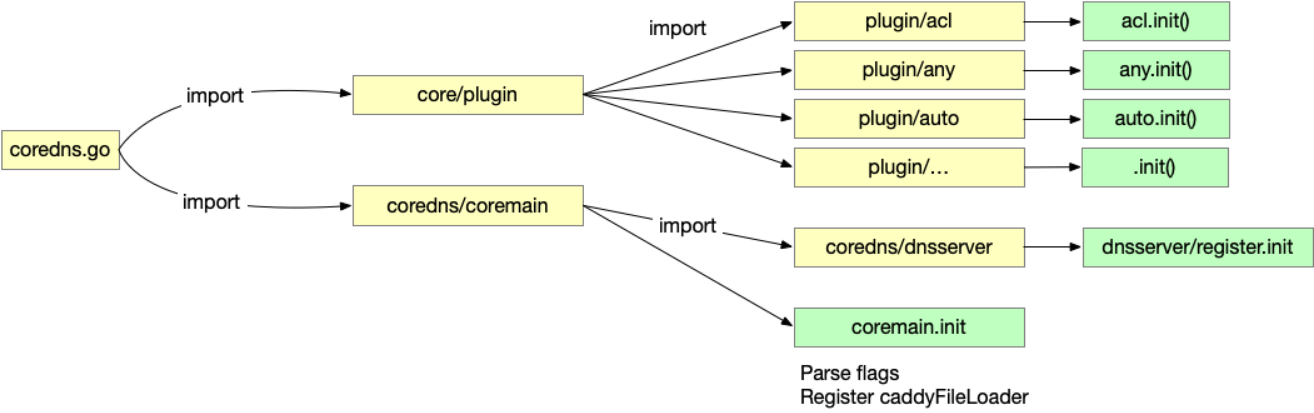
## 1.1 注册

// coredns.go

package main

// import两个包文件，执行包文件中的init函数

import (  
 \_ “github.com/coredns/coredns/core/plugin” // Plug in CoreDNS.  
 “github.com/coredns/coredns/coremain”  
)  
  
func main() {  
 coremain.Run()  
}



### 1.1.1 Plugin中init函数

Plugin 中的 init() 做的事情很简单，就是调用 coredns.plugin.Register 函数，将 Plugin 注册到 caddy 中去，告诉 caddy 两个事情：

1. 这个 Plugin 所支持的 ServerType 是 DNS
2. Plugin 的 Action，即初始化函数。这里只是注册，并没有运行过。

// generated by directives\_generate.go; DO NOT EDIT

// zplugin.go  
package plugin  
  
import (  
 // Include all plugins.  
 \_ "github.com/coredns/caddy/onevent"  
 \_ "github.com/coredns/coredns/plugin/acl"

………………………  
 \_ "github.com/coredns/coredns/plugin/whoami"  
)

以acl插件为例，acl包下

// setup.go

const *pluginName* = "acl"  
func init() { plugin.Register(*pluginName*, setup) }

func setup(c \*caddy.Controller) error {  
 a, err := parse(c)  
 if err != nil {  
 return plugin.Error(*pluginName*, err)  
 }  
  
 dnsserver.GetConfig(c).AddPlugin(func(next plugin.Handler) plugin.Handler {  
 a.Next = next  
 return a  
 })  
  
 return nil  
}

// plugin.go，该文件下唯一函数-

package plugin  
  
import "github.com/coredns/caddy"  
  
// Register registers your plugin with CoreDNS and allows it to be called when the server is running.  
func Register(name string, action caddy.SetupFunc) {  
 caddy.RegisterPlugin(name, caddy.Plugin{  
 ServerType: "dns",  
 Action: action,  
 })  
}

### Register中init函数

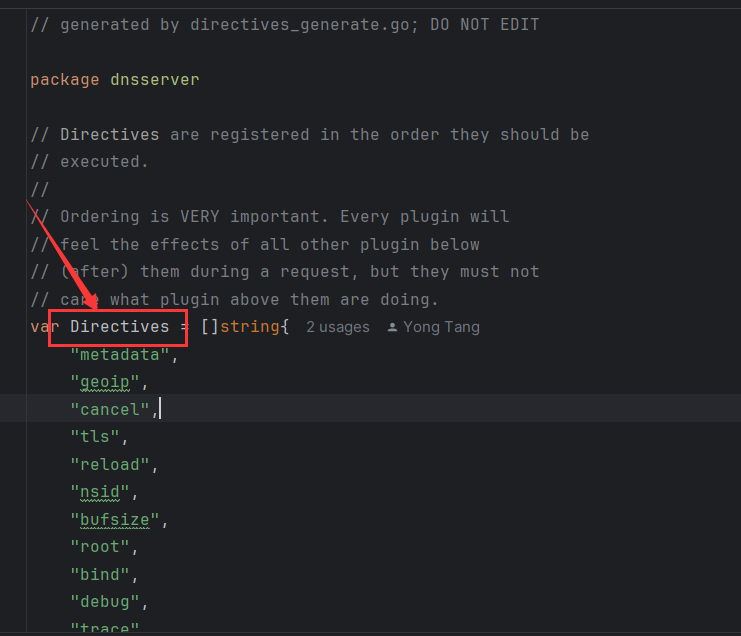
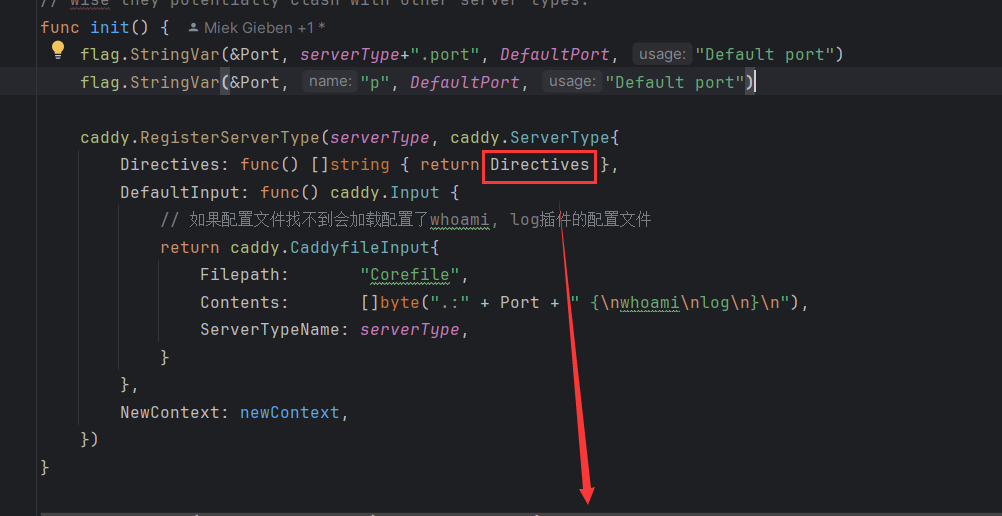
主要的动作是使用 caddy 的接口 caddy.RegisterServerType 注册上了一个新的 Server 类型

func init() {

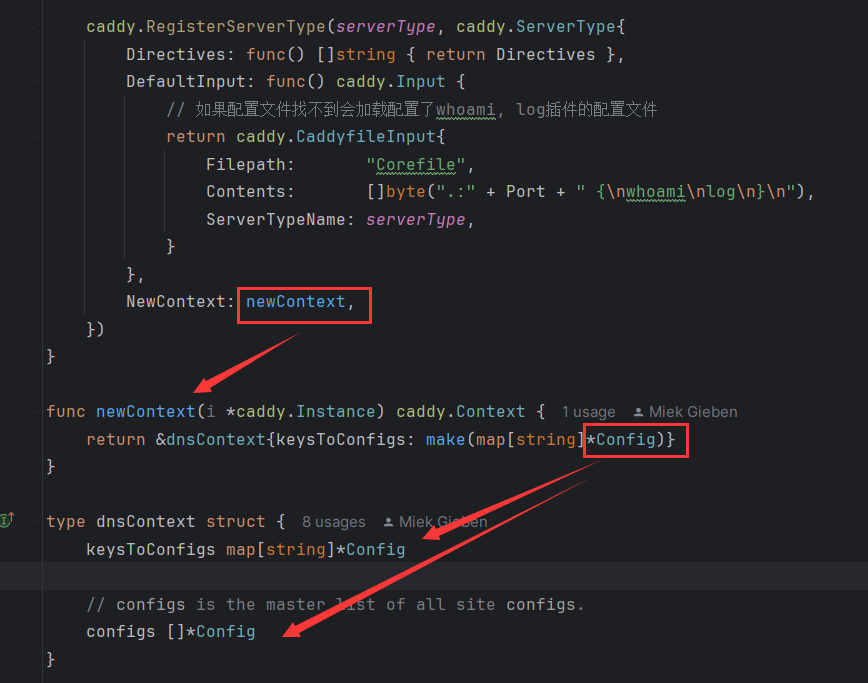
caddy.RegisterServerType(*serverType*, caddy.ServerType{  
 Directives: func() []string { return Directives },  
 DefaultInput: func() caddy.Input {  
 // 如果配置文件找不到会加载配置了whoami, log插件的配置文件  
 return caddy.CaddyfileInput{  
 Filepath: "Corefile",  
 Contents: []byte(".:" + Port + " {\nwhoami\nlog\n}\n"),  
 ServerTypeName: *serverType*,  
 }  
 },  
 NewContext: newContext,  
 })  
}

注册的时候，要按照 caddy 的接口告诉 caddy：

1. Directives: 新的 ServerType 支持的 Directives 是什么；



1. DefaultInput: 在没有配置文件输入的时候，默认的配置文件是什么，这个选项其实不重要；
2. NewContext: 这个是最重要的，如何生成对应这个 ServerType 的 Context，Context 是后面管理 Config 实例的主要入口；

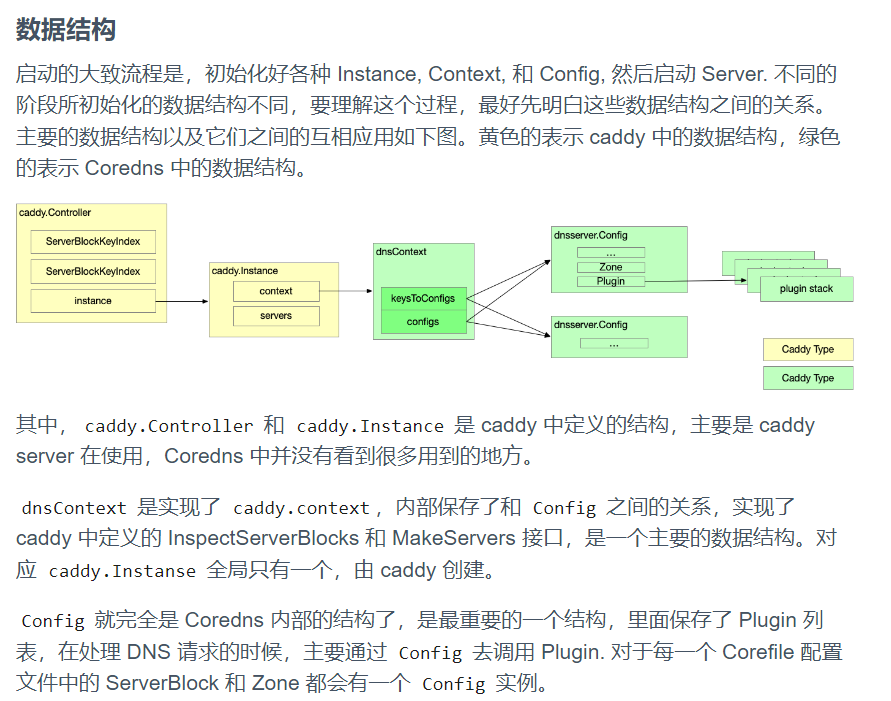


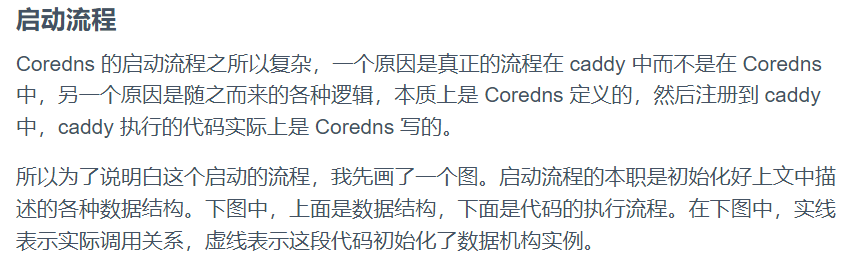
 

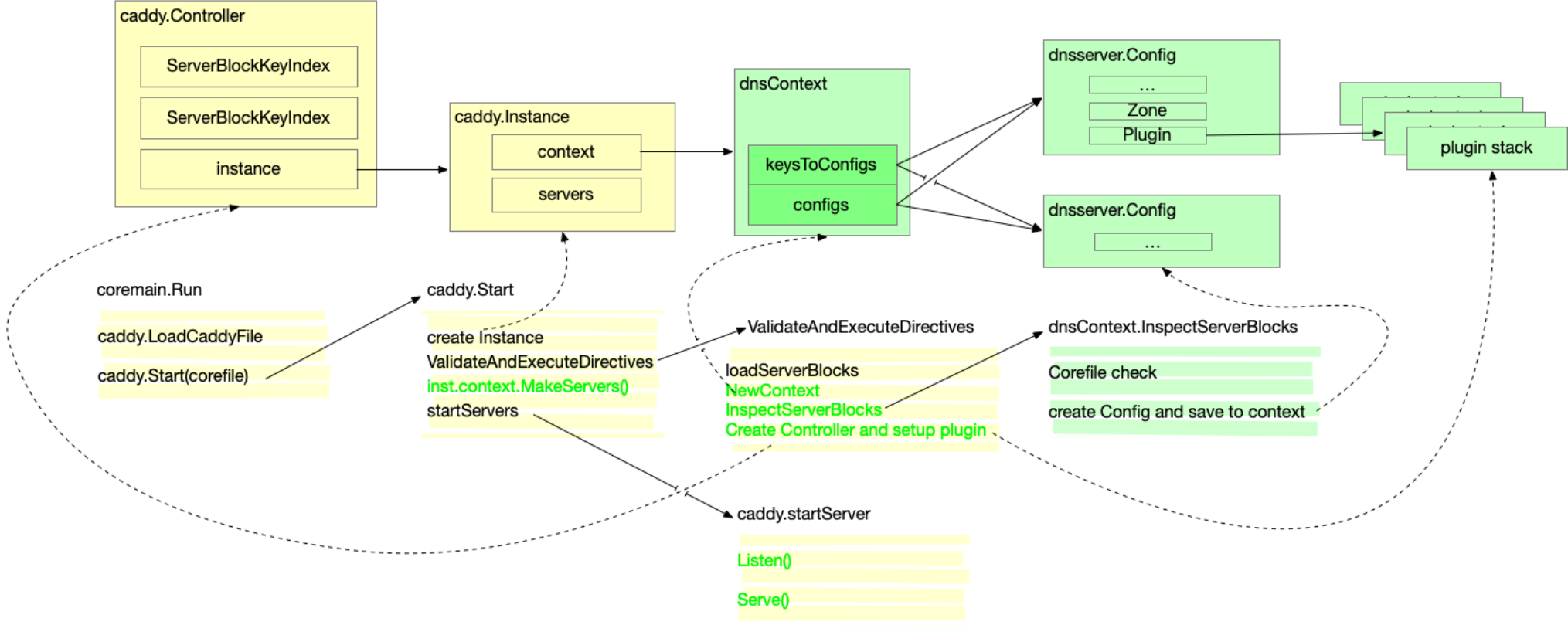
## 启动流程



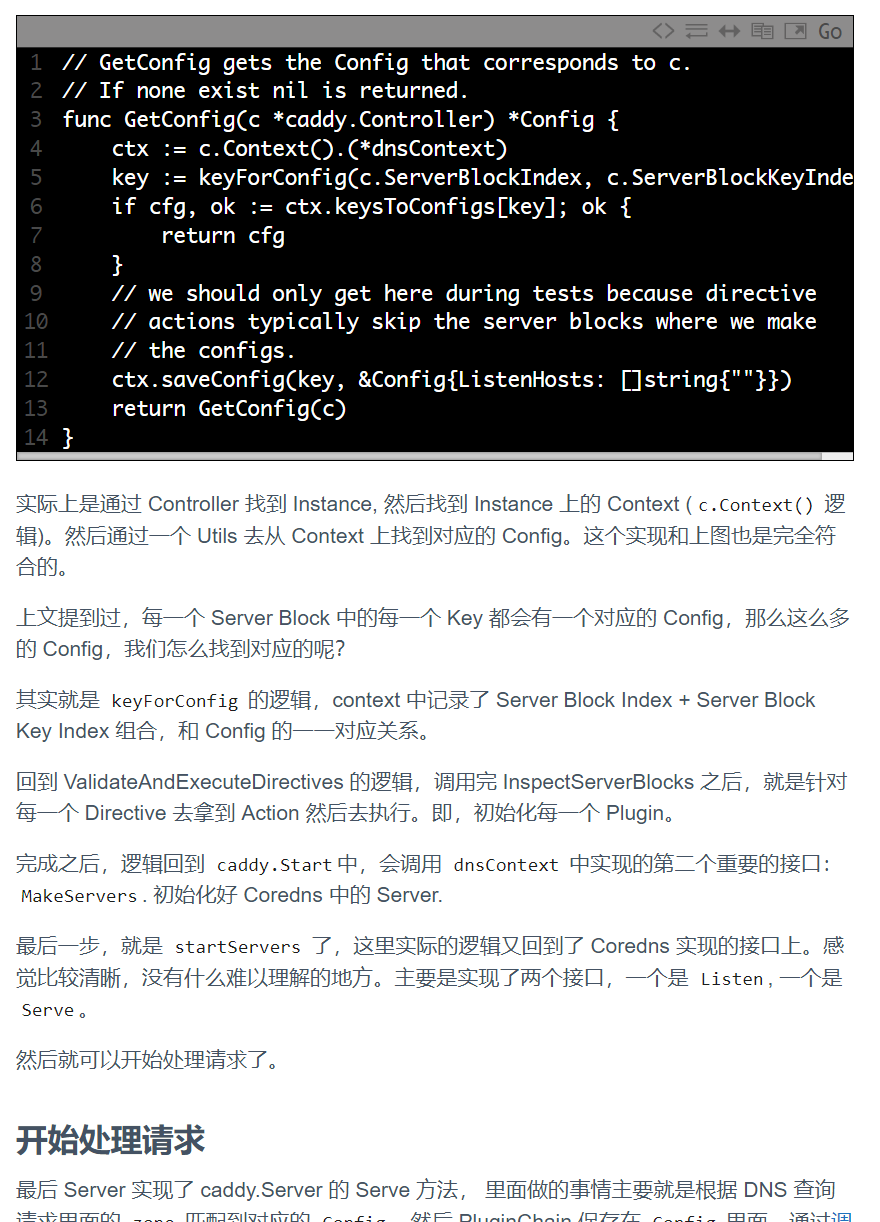


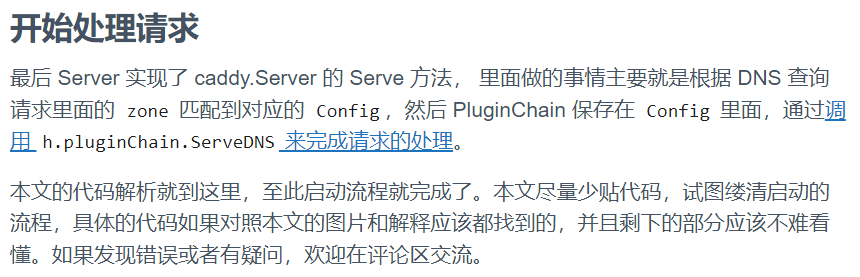












# 2.Coredns插件配置