Beego开发

目录

Beego安装与使用

1. go get github.com/astaxie/beego

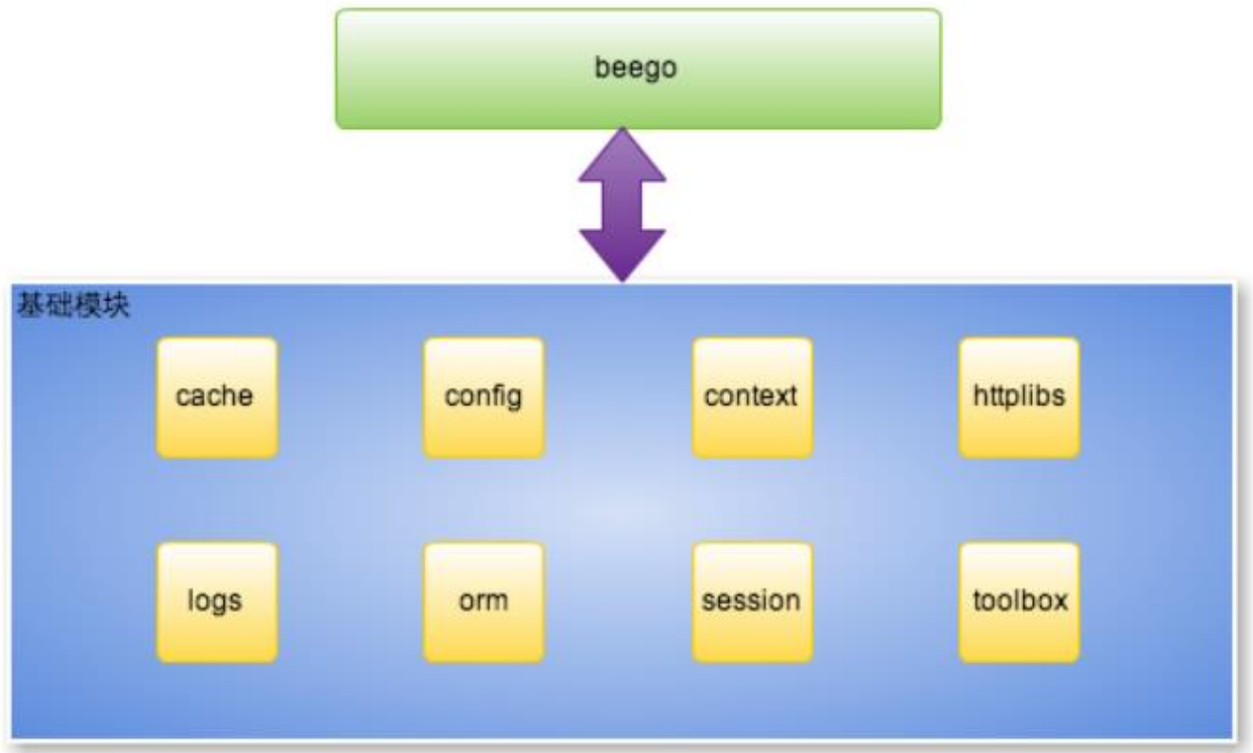
```
package main

import "github.com/astaxie/beego"

func main() {
    beego.Run()
}
```

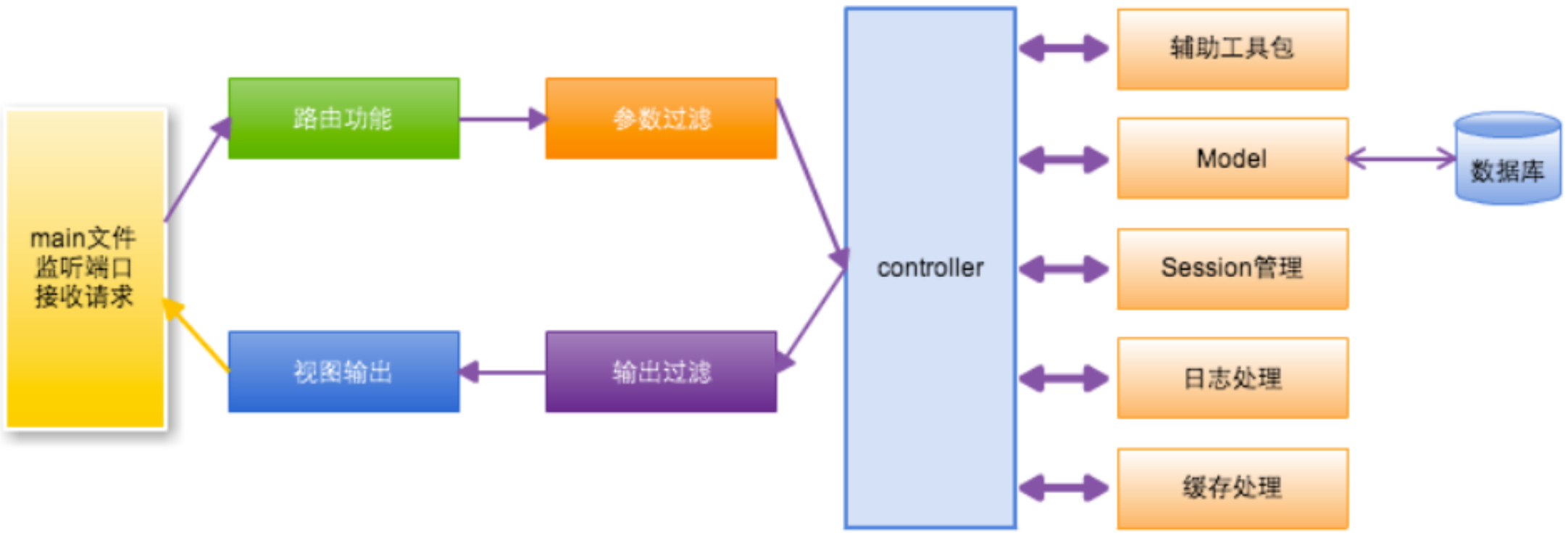
Beego的架构

2. beego中的组件



Beego架构

3. Beego执行流程



Beego使用

4. beego目录结构

```
├── conf
│   └── app.conf
├── controllers
│   ├── admin
│   └── default.go
├── main.go
├── models
│   └── models.go
├── static
│   ├── css
│   ├── ico
│   ├── img
│   └── js
└── views
    ├── admin
    └── index.tpl
```

Beego使用

5. bee工具安装

```
go get github.com/beego/bee
```

- 快速构建beego项目
- 创建web项目　bee new 项目的名字
- 创建api项目　bee api 项目的名字

Beego使用

6. bee路由配置

```
beego.Router("/", &controllers.MainController{})
```

- Router方法设置url和处理该url的controller
- 程序启动时路由配置加载好，保存在map中
- 请求处理时，通过请求的url进行查找对应的controller，把处理路由到controller进行执行
- 开发人员只需要编写自己的controller就可以了

Beego使用

7. controller代码分析

- 用户的controller嵌套bee.Controller， 继承bee.Controller所有属性和方法
- Controller默认使用restful的风格，Get请求对应Get方法
- 请求处理时，通过请求的url进行查找对应的controller，把处理路由到controller进行执行
- 通过业务逻辑处理之后，把数据赋值给Controller.Data这个map即可

Beego使用

8. model层实现业务逻辑

Beego使用

9. views层实现模板的渲染

- Beego模板默认支持 .tpl和.html两种后缀
- beego.AddTemplateExt增加新的模板后缀
- Beego模板用的就是Go官网自带的模板

Beego使用

10. 静态文件处理

- Beego默认使用static目录作为静态文件目录
- beego. SetStaticPath增加新的静态文件目录

Beego使用

11. 配置文件使用

- beego 默认会解析conf/app.conf

```
appname = beepkg
httpaddr = "127.0.0.1"
httpport = 9090
runmode ="dev"
autorender = false
recoverpanic = false
viewspath = "myview"
```

Beego使用

12. 配置文件使用

- 自定义配置读取

```
mysqluser = "root"
mysqlpass = "rootpass"
mysqlurls = "127.0.0.1"
mysqldb   = "beego"
```

```
beego.AppConfig.String("mysqluser")
beego.AppConfig.String("mysqlpass")
beego.AppConfig.String("mysqlurls")
beego.AppConfig.String("mysqldb")
```

Beego使用

13. 配置文件使用

- 自定义配置读取

AppConfig 的方法如下:

- Set(key, val string) error
- String(key string) string
- Strings(key string) []string
- Int(key string) (int, error)
- Int64(key string) (int64, error)
- Bool(key string) (bool, error)
- Float(key string) (float64, error)
- DefaultString(key string, defaultVal string) string
- DefaultStrings(key string, defaultVal []string)
- DefaultInt(key string, defaultVal int) int
- DefaultInt64(key string, defaultVal int64) int64
- DefaultBool(key string, defaultVal bool) bool
- DefaultFloat(key string, defaultVal float64) float64
- DIY(key string) (interface{}, error)
- GetSection(section string) (map[string]string, error)
- SaveConfigFile(filename string) error

Beego使用

14. 配置文件使用

- 不同级别的配置读取

```
[dev]
httpport = 8080
[prod]
httpport = 8088
[test]
httpport = 8888
```

beego.AppConfig.String(“dev::httpport”)。

Beego使用

15. beego路由详解

- 固定路由

```
beego.Router("/", &controllers.MainController{})
beego.Router("/admin", &admin.UserController{})
beego.Router("/admin/index", &admin.ArticleController{})
beego.Router("/admin/addpkg", &admin.AddController{})
```

Beego使用

16. beego路由详解

- 正则路由

  - beego.Router(“/api/?:id”, &controllers.RController{})

    默认匹配 //匹配 /api/123 :id = 123 可以匹配 /api/ 这个URL

  - beego.Router(“/api/:id”, &controllers.RController{})

    默认匹配 //匹配 /api/123 :id = 123 不可以匹配 /api/ 这个URL

  - beego.Router(“/api/:id([0-9]+)“, &controllers.RController{})

    自定义正则匹配 //匹配 /api/123 :id = 123

  - beego.Router(“/user/:username([\w]+)“, &controllers.RController{})

    正则字符串匹配 //匹配 /user/astaxie :username = astaxie

  - beego.Router(“/download/*.*”, &controllers.RController{})

    *匹配方式 //匹配 /download/file/api.xml :path= file/api :ext=xml

```
this.Ctx.Input.Param(":id")
this.Ctx.Input.Param(":username")
```

Beego使用

17. beego路由详解

- 自定义方法

```
beego.Router("/api/list",&RestController{},"*:ListFood")
beego.Router("/api/create",&RestController{},"post:CreateFood")
beego.Router("/api/update",&RestController{},"put:UpdateFood")
beego.Router("/api/delete",&RestController{},"delete:DeleteFood")
```

Beego使用

18. 获取用户提交的参数

- Controller中的方法获取

  - GetString(key string) string
  - GetStrings(key string) []string
  - GetInt(key string) (int64, error)
  - GetBool(key string) (bool, error)
  - GetFloat(key string) (float64, error)

配置文件里设置 copyrequestbody = true

Beego使用

19. 获取用户提交的参数

- 直接通过Ctx.Input.RequestBody获取原始的数据
- 配置文件里设置 copyrequestbody = true

```
func (this *ObjectController) Post() {
    var ob models.Object
    json.Unmarshal(this.Ctx.Input.RequestBody, &ob)
    objectid := models.AddOne(ob)
    this.Data["json"] = "{\"ObjectId\":\"" + objectid + "\"}"
    this.ServeJSON()
}
```

配置文件里设置 copyrequestbody = true

Beego使用

20. 日志使用

- beego.SetLogger("file", `{"filename":"logs/test.log"}`)
- beego.BeeLogger.DelLogger("console")

```
beego.Emergency("this is emergency")
beego.Alert("this is alert")
beego.Critical("this is critical")
beego.Error("this is error")
beego.Warning("this is warning")
beego.Notice("this is notice")
beego.Informational("this is informational")
beego.Debug("this is debug")
```

配置文件里设置 copyrequestbody = true

Beego使用

21. 日志使用

- beego.SetLevel(beego.LevelInformational)
- beego.SetLogFuncCall(true) 设置打印行号

```
LevelEmergency
LevelAlert
LevelCritical
LevelError
LevelWarning
LevelNotice
LevelInformational
LevelDebug
```

课后练习

1. 使用beego把上节课的短url项目重写一遍