构建 Go Web 应用

无闻 @Unknwon

个人简介

Go 语言爱好者，目前主要参与 Gogs 和 Docker.cn 的项目开发。

Go 语言的优势

二进制分发

编译严格

原生跨平台

Web 应用的类型

服务形式的应用

软件形式的应用

服务形式的应用

统一部署和升级

易于进行不兼容修改

掌握运行环境和依赖版本

★ 软件形式的应用

用户自主部署升级

不兼容修改需要慎重

松散的应用依赖版本

Gogs 💥 Git 自助服务

二进制或源码构建

支持三种数据库

支持主流操作系统

Web 框架的选择

选择框架的考虑因素

- 框架性能
- 开发成本
- 扩展能力
- 社区支持

框架性能

只要性能不是特别渣。😏

开发成本

成本过高不利于快速成型。😱

扩展能力

利于扩展才能保证续航。 

社区支持

没有文档的开源都是骗子。 

Gogs 💡 Martini

性能不高 但过得去

函数为主 编写简单

依赖注入 极易扩展

社区支持 😂😂😂😂

应用功能模块化

必然现象

一个大的项目在开发过程中必然会产生许多可以被复用的代码模块。

功能设计

一个模块做且只做好一件事，模块之间相互独立，实现与应用的解耦。

最佳实践

一个模块为一个包，包内根据功能类型区分文件，文件内容根据主次关系进行排序。

Martini ✨ Macaron

利用接口替代反射

将模块封装成中间件

提供更多高级功能

详细文档与技术支持

应用部署升级

二进制分发

通过直接下载对应平台的
二进制即可完成部署或者升级，
用户不需要下载安装开发环境。

源码构建

通过拉取项目源码进行构建或者更新升级，适合快速升级和 Bug 调试。

数据库选择

自由选择

不要随便绑架用户，给予用户足够的选择空间。

技术选型

在大多数情况下，都可以使用 ORM 来完成数据库相关操作，从而兼容更多的数据库。

Gogs 😎 XORM

SQLite3

MySQL

PostgreSQL

依赖版本管理

不使用任何工具

 命令简单，快速构建

 兼容性高，全球统一

 毫无版本概念

Godep

 颗粒度小，精准依赖

 缓存副本，无网构建

 版本控制冗余的内容

Gopm

- 👍 只下源码，不要历史
- 👍 颗粒度小，精准依赖
- 👍 缓存中转，天朝特供
- ❌ 依赖缓存服务器

Gogs 原生 + Gopm

使用原生命令构建最新版

发布时保存依赖版本
旧版仍可用 Gopm 构建

让我用一分钟打动你...

Q&A

新浪微博：@无闻Unknown

GitHub：@Unknwon