# 为序列化的解析器定义结构体



# 定义序列化函数

将engine.Request转成能在网上传递的Request，即为了序列化新定义的Request结构体

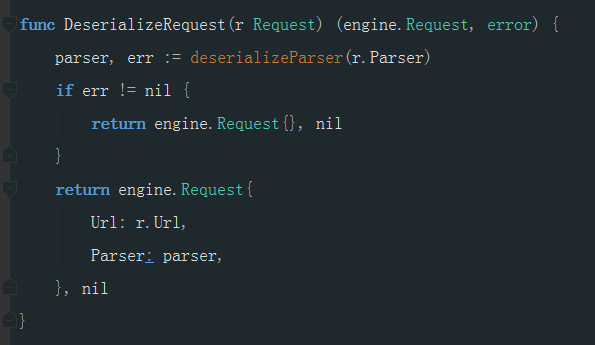
将engine.ParserResult转成能在网上传递的ParserResult，同上

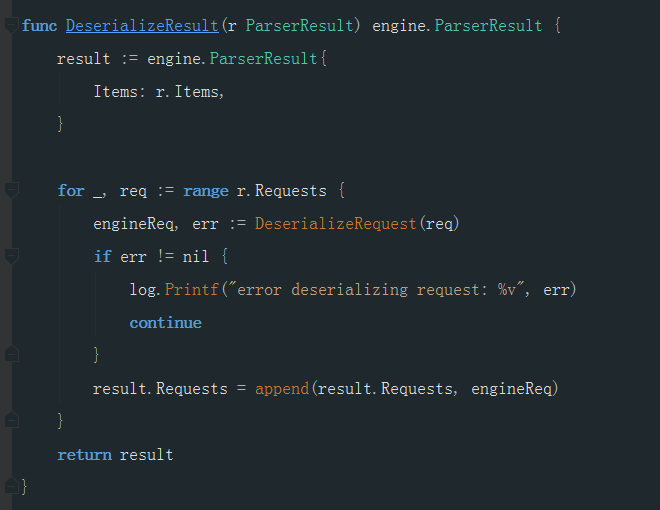
Parser接口的第二个方法Serialize就是用来返回可在网上传递的数据

再加上为了序列化新定义的Request、ParserResult结构体，这样就可以进行序列化操作



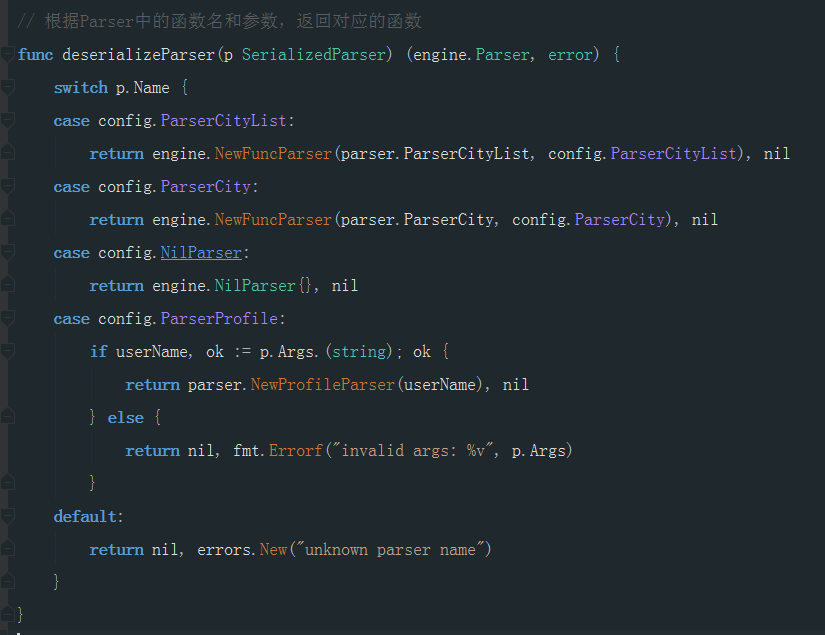
# 定义反序列化函数





此函数是Parser.Serialize函数的相反操作

用函数名和参数返回序列化前的Parser对象



# 实现worker的RPC

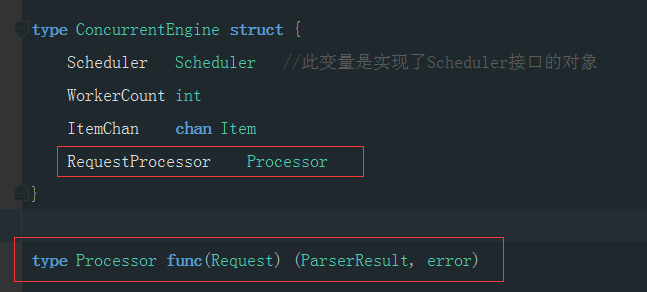


## Worker的RPC server端



## 把worker方法定义成engine结构体的属性

在engine的结构体里增加一个方法类型的属性，原版worker或RPC版worker都可通过参数传递过来

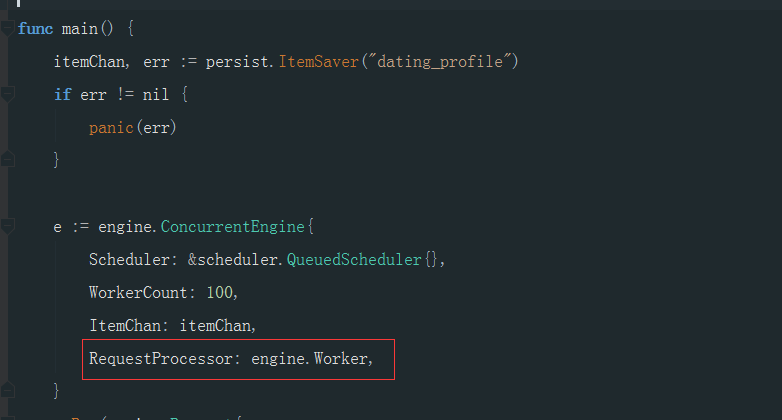


把createWorker改成engine结构体的方法，使其可以使用RequestProcessor属性

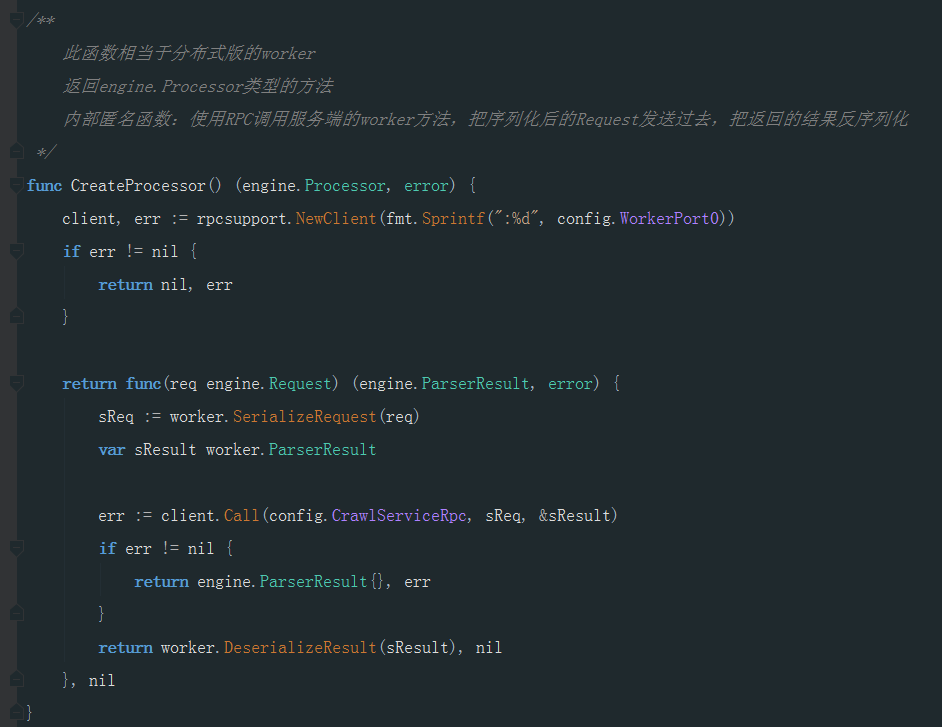


并发版爬虫的main文件里，engine就需要把这个属性加上了，属性值是原版的worker

分布式的main文件属性值则是RPC版的worker



## Worker的PRC client端



修改分布式版的main文件，在engine结构体里传入RequestProcessor属性（worker方法）

