用PY实现"Go元编程"

翁伟 @ 希品科技

北京 上海 珠海 苏州 杭州

主办: PyChina.org, GDG

关于我

- 翁伟，原常驻新加坡，现居深圳
- 不断折腾（10年？）的全端程序员
  - .net -> python -> go
- 再次回国创业，觅伙伴
  - wengwei@xipintech.com

元编程

- 指某类计算机程序的编写，这类计算机程序编写或者操纵其它程序（或者自身）作为它们的数据

- 与手工编写全部代码相比，程序员可以获得更高的工作效率

元编程其实无处不在

- 最常见的元编程工具是编译器，它可以将程序员使用高级语言编写的相对短小的程序转换为等价的汇编语言或者机器语言程序

- 其它元编程系统则允许以编程方式操纵一种语言。宏系统即是这样一种简单的系统

C宏

```
#define list_for_each(pos, head) \
for (pos = (head)->next; pos != (head);
pos = pos->next)
```

“Go，互联网时代的C，下一个C”

– 许式伟

Go元编程?

- 木有宏
- 木有泛型
- 反射很难用
- “gopher 这几年很努力，解决了许多别的语言中不存在的问题……” - @赖勇浩

看看很努力的gopher

如何用py来实现go的泛型?

```go
package set

type StringSet struct {
    data map[string]struct{}
}

func (s *StringSet) Init() {
    s.data = make(map[string]struct{})
}

func (s *StringSet) Contains(v string) bool {
    _, ok := s.data[v]
    return ok
}

func (s *StringSet) Add(v string) {
    s.data[v] = struct{}{}
}

func (s *StringSet) Len() int {
    return len(s.data)
}

func (s *StringSet) Copy() *StringSet {
    res := NewStringSet()
    res.Union(s)
    return res
}
```

```go
package set

type IntSet struct {
    data map[int]struct{}
}

func (s *IntSet) Init() {
    s.data = make(map[int]struct{})
}

func (s *IntSet) Contains(v int) bool {
    _, ok := s.data[v]
    return ok
}

func (s *IntSet) Add(v int) {
    s.data[v] = struct{}{}
}

func (s *IntSet) Len() int {
    return len(s.data)
}

func (s *IntSet) Copy() *IntSet {
    res := NewIntSet()
    res.Union(s)
    return res
}
```

- `StringSet`与`IntSet`代码很类似，仅类型不同

- 泛型的典型使用场景

```go
type SomeTypeSet struct {
    data map[SomeType]struct{}
}

func (s *SomeTypeSet) Init() {
    s.data = make(map[SomeType]struct{})
}

func (s *SomeTypeSet) Contains(v SomeType) bool {
    _, ok := s.data[v]
    return ok
}

func (s *SomeTypeSet) Add(v SomeType) {
    s.data[v] = struct{}{}
}

func (s *SomeTypeSet) Len() int {
    return len(s.data)
}
```

```
src = open("sometype_set.go").read()
for T in ["string", "int"]:
    f = open("%s_set.go" % T, "w+")
    f.write(src.replace("SomeTypeSet",
        "%sSet" % T.capitalize()).replace("SomeType", T))
    f.close()
```

简单、粗暴、有效

自动生成

- MakeFile?

Makefile

```
export GOPATH := $(shell pwd)
export PATH := ${PATH}:${GOPATH}/bin
export GOBIN := ${GOPATH}/bin

build:
    python gen_codes.py
    go install main
```

- 监控文件修改：github.com/gorakhargosh/watchdog

再来看看“ORM”

我们用thrift作对象

定义的DSL

```
struct UserProfile {
    1: string ID (
        label = "用户资料",
        baseURL = "/admin/portal/user"
        search = "Name"
        listedFields = "Name, Email"
    )
    2: required string Name (label="姓名")
    3: string MobilePhone (label="移动电话")
    4: string Email (label="电子邮箱")
    5: string HomeAddress (label="家庭地址")
    6: string PostCode (label="邮编")
    7: string Fax (label="传真")
}
```

用户资料

姓名*

移动电话

传真

电子邮箱

邮编

家庭地址

保存

取消

用户资料

新建

搜索...

搜索

序号	姓名	电子邮箱	操作
1	翁伟	wengwei@xipintech.com	编辑

当前显示所有 1 条记录　每页显示 20 条记录

同一对象元信息

- 增删改查
- 全文搜索
- 后台权限
- RPC

对象传递、赋值、显示

有大量相似的代码

```go
func (o *UserProfile) ReadForm(params map[string]string) (hasError bool) {
    if val, ok := params["Name"]; ok {
        o.Name = val
    }
    if val, ok := params["MobilePhone"]; ok {
        o.MobilePhone = val
    }
    if val, ok := params["Email"]; ok {
        o.Email = val
    }
    if val, ok := params["HomeAddress"]; ok {
        o.HomeAddress = val
    }
    if val, ok := params["PostCode"]; ok {
        o.PostCode = val
    }
    if val, ok := params["Fax"]; ok {
        o.Fax = val
    }
    return o.ValidateData()
}
```

```go
func (o *UserProfile) GetFieldAsString(fieldKey string) (Value string) {
    switch fieldKey {
    case "ID":
        Value = o.ID.Hex()
    case "Name":
        widget := o.NameWidget()
        if widget.Type == "select" {
            Value = widget.Val()
        } else if widget.GetBindData != nil {
            Value = widget.Val()
        } else {
            Value = o.Name
        }
    case "MobilePhone":
        widget := o.MobilePhoneWidget()
        if widget.Type == "select" {
            Value = widget.Val()
        } else if widget.GetBindData != nil {
            Value = widget.Val()
        } else {
            Value = o.MobilePhone
        }
    case "Email":
        widget := o.EmailWidget()
```

```go
func (o *UserProfile) Widgets() []*Widget {
    return []*Widget{
        o.NameWidget(),
        o.MobilePhoneWidget(),
        o.EmailWidget(),
        o.HomeAddressWidget(),
        o.PostCodeWidget(),
        o.FaxWidget(),
    }
}
```

```go
func (o *UserProfile) GetListedLabels() []*IDLabelPair {
    return []*IDLabelPair{
        &IDLabelPair{
            ID:    "Name",
            Label: "姓名",
        },
        &IDLabelPair{
            ID:    "Email",
            Label: "电子邮箱",
        },
    }
}
```

- 上述代码都根据对象元信息生成出来

- 生成的代码不会错

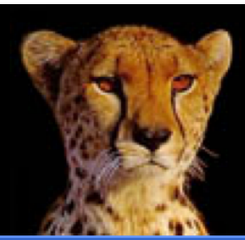
如何实现？

解析对象

- ptsd: github.com/wickman/ptsd
  - thrift lexer/parser using ply
- ply: www.dabeaz.com/ply/
  - An implementation of lex and yacc parsing tools for Python

代码生成模板

- www.cheetahtemplate.org



CHEETAH

The Python-Powered Template Engine

Overview | Download | Docs | Examples | Who Uses It? | Testimonials | Help Out

Fast, Flexible, Powerful

web development & code generation

Cheetah is an open source template engine and code generation tool, written in Python. It can be used standalone or combined with other tools and frameworks. Web development is its principle use, but Cheetah is very flexible and is also being used to generate C++ game code, Java, sql, form emails and even Python code.

Cheetah has a large and active user community. Products built with Cheetah are used by many of the Fortune 500.

Mailing List | Changes | github

"I'm enamored with Cheetah"

- ***Sam Ruby***, *senior member of IBM's Emerging Technologies Group & director of Apache Software Foundation*

"Give Cheetah a try. You won't regret it. ... Cheetah is a truly powerful system. ... Cheetah is a serious contender for the 'best of breed'"

- ***Alex Martelli***, *Google uber techie, core Python developer & author of several popular Python books*

```
import sys
from ptsd.loader import Loader
from Cheetah.Template import Template

thrift_file = sys.argv[1]
out_path = sys.argv[2]

def main(thrift_file):
    loader = Loader(thrift_file)

    for module in loader.modules.values():
        for struct in module.structs:
            tpl = open('tmpl/go.tmpl', 'r').read().decode("utf8")
            t = Template(tpl, searchList=[{"obj": obj}])
            code = str(t)
            write_file(out_path + '/gen_' + obj.name.value.lower() + ".go", code)

main(thrift_file)
```

```
func (o *$obj.name.value) ReadForm(params map[string]string) (hasError bool) {
#for field in $obj.fields
    if val, ok := params["$field.name.value"]; ok {
    #if $field.type == "string"
        o.$field.name.value = val
    #else if $field.type == "i32"
        intVal, _ := strconv.Atoi(val)
        o.$field.name.value = int32(intVal)
    #else if $field.type == "double"
        floatVal, _ := strconv.ParseFloat(val, 64)
        o.$field.name.value = floatVal
    #else if $field.type == "bool"
        o.$field.name.value = (val != "" && val != "false")
    #else if $field.type == "list<string>"
        o.$field.name.value = strings.Split(val, "\n")
        for i, k := range o.$field.name.value {
            o.$(field.name.value)[i] = strings.Trim(k, "\r")
        }
    #end if
    }
#end if
#end for
    return o.ValidateData()
}
```

```go
func (o *UserProfile) ReadForm(params map[string]string) (hasError bool) {
    if val, ok := params["Name"]; ok {
        o.Name = val
    }
    if val, ok := params["MobilePhone"]; ok {
        o.MobilePhone = val
    }
    if val, ok := params["Email"]; ok {
        o.Email = val
    }
    if val, ok := params["HomeAddress"]; ok {
        o.HomeAddress = val
    }
    if val, ok := params["PostCode"]; ok {
        o.PostCode = val
    }
    if val, ok := params["Fax"]; ok {
        o.Fax = val
    }
    return o.ValidateData()
}
```

```
func (o *$obj.name.value) GetFieldAsString(fieldKey string) (Value string) {
    switch fieldKey {
#for field in $obj.fields
    case "$field.name.value":
    #if $field.name.value == "ID"
        Value = o.ID.Hex()
    #else if $field.type == "string"
        Value = o.$field.name.value
        #end if
    #else if $field.type == "double"
        Value = strconv.FormatFloat(o.$field.name.value, 'f', 2, 64)
    #else if $field.type == "bool"
        Value = strconv.FormatBool(o.$field.name.value)
    #end if
#end for
    }
    return
}
```

```go
func (o *UserProfile) GetFieldAsString(fieldKey string) (Value string) {
    switch fieldKey {
    case "ID":
        Value = o.ID.Hex()
    case "Name":
        widget := o.NameWidget()
        if widget.Type == "select" {
            Value = widget.Val()
        } else if widget.GetBindData != nil {
            Value = widget.Val()
        } else {
            Value = o.Name
        }
    case "MobilePhone":
        widget := o.MobilePhoneWidget()
        if widget.Type == "select" {
            Value = widget.Val()
        } else if widget.GetBindData != nil {
            Value = widget.Val()
        } else {
            Value = o.MobilePhone
        }
    case "Email":
        widget := o.EmailWidget()
```

```
func (o *$obj.name.value) Widgets() []*Widget {
    return []*Widget{
#for field in $obj.fields
#if $field.name.value != "ID"
        o.${field.name.value}Widget(),
#end if
#end for
    }
}
```

```go
func (o *UserProfile) Widgets() []*Widget {
    return []*Widget{
        o.NameWidget(),
        o.MobilePhoneWidget(),
        o.EmailWidget(),
        o.HomeAddressWidget(),
        o.PostCodeWidget(),
        o.FaxWidget(),
    }
}
```

```
func (o *$obj.name.value) GetListedLabels() []*IDLabelPair {
    return []*IDLabelPair{
        #for field in $obj.listedFields
        &IDLabelPair {
            ID : "$field.key",
            Label : "$field.label",
            #if hasattr(field, "order")
            Order : "$field.order",
            #end if
            #if hasattr(field, "widget")
            Widget: "$field.widget",
            #end if
        },
        #end for
    }
}
```

```go
func (o *UserProfile) GetListedLabels() []*IDLabelPair {
    return []*IDLabelPair{
        &IDLabelPair{
            ID:    "Name",
            Label: "姓名",
        },
        &IDLabelPair{
            ID:    "Email",
            Label: "电子邮箱",
        },
    }
}
```

YAML也是好东西~

工作流程

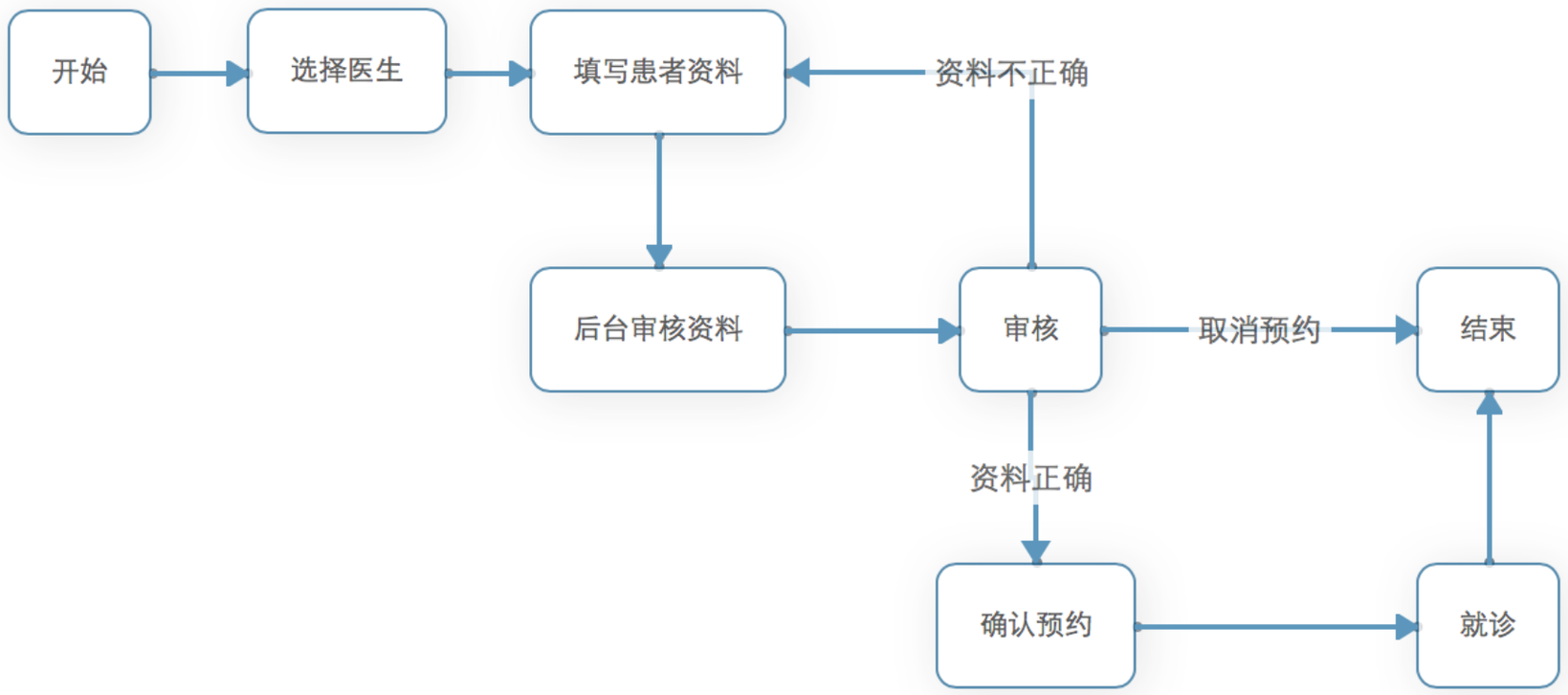
流程定义:

```
医生预约流程:

    开始: 选择医生

    选择医生:
        next: 填写患者资料

    填写患者资料:
        next: 后台审核资料

后台审核资料:
    appr:
        - 资料正确:
            next: 确认预约
        - 资料不正确:
            next: 填写患者资料
        - 取消预约:
            next: 结束

确认预约:
    next: 就诊

就诊:
    next: 结束
```

格式检查结果(json)

```
{ '医生预约流程':
   { '开始': '选择医生',
     '选择医生': { next: '填写患者资料' },
     '填写患者资料': { next: '后台审核资料' },
     '后台审核资料':
      { appr:
         [ { '资料正确': { next: '确认预约' } },
           { '资料不正确': { next: '填写患者资料' } },
           { '取消预约': { next: '结束' } } ] },
     '确认预约': { next: '就诊' },
     '就诊': { next: '结束' } } }
```

医生预约流程



```python
# -*- coding: utf-8 -*-
import yaml
from Cheetah.Template import Template

def main(flow_file):
    source = open(flow_file, 'r').read()
    data = yaml.load(source)
    flowObj = tranformData(data)

    tpl = open('tmpl/go_flow.tmpl', 'r').read().decode("utf8")
    t = Template(tpl, searchList=[flowObj])

    outpath = outpath + "/gen_" + filename + ".go"
    open(outpath, "w").write(res)

main(flow_file)
```

- 实现与ORM类似，只是把DSL从thrift换成YAML
- YAML
  - 表达能力更强
  - 需求人员友好

外部接口

- 需要多端实现：
  - web
  - mobile web
  - mobile client
- 各端业务一致，但调用业务可能需要不同封装

```go
func UserSendMsg(fromUser *User, toUserID string, msg *Msg) (err error) {
    ...
    return
}
```

```go
func UserSendMsg(ctx *web.Context) (err error) {
    fromUser := ctx.GetCurrentUser()
    toUserID := ctx.Params["toUserID"]
    msg := model.NewMsgFromForm(ctx.Params)

    return msgService.UserSendMsg(fromUser, toUserID, msg)
}
```

```go
func UserSendMsg(ctx *web.Context) {
    fromUser := ctx.GetCurrentUser()
    toUserID := ctx.Params["toUserID"]
    var msg *model.Msg
    json.Unmarshall(ctx.Params["msg"], &msg)

    err = msgService.UserSendMsg(fromUser, toUserID, msg)
    ctx.Write(json.Marshall(err))
}
```

```go
func  (x *ApiImpl) UserSendMsg(token string, toUserID string, msg *Msg) (err error) {
    fromUser, apiErr := x.getUser(token)
    if apiErr != nil {
        return nil, apiErr
    }
    return msgService.UserSendMsg(fromUser, toUserID, msg)
}
```

封装代码

都不应该人肉写

```go
// export: webCtrl, jsonRPC, thriftRPC
func UserSendMsg(fromUser *User, toUserID string, msg *Msg) (err error) {
    ...
    return
}
```

所以我们项目有~~30万~~80万行代码

```
     663 ./watch/scan.go
     184 ./watch/watchdir.go
     563 ./xuanwu/src/common.go
  836733 total
➜  zfw_code git:(develop) ✗ find . -name '*.go' | xargs wc -l
```

- 上述仅仅是“演示代码”
  - 实际代码要复杂得多多
- ORM有开源：github.com/sipin/xuanwu
  - 仅供参考、切勿使用
  - 无文档
  - 不维护
  - 内部已推倒重来

如何定制?

- 生成一次，后面随便改
- 随模板修改反复生成:
  - 提供定制接口方法回调
  - 覆盖特定方法

```go
func UserProfileAddInsertCallback(cb func(obj IXuanWuObj)) {
    insertCB_UserProfile = append(insertCB_UserProfile, cb)
}

func UserProfileAddUpdateCallback(cb func(obj IXuanWuObj)) {
    updateCB_UserProfile = append(updateCB_UserProfile, cb)
}

func UserProfileAddExtraValidation(validate func(o *UserProfile) bool) {
    extraValidation_UserProfile = append(extraValidation_UserProfile, validate)
}
```

```go
package userprofile

import (
    "xuanwu"
    "app/models/userprofile"
)

func init() {
    //call back
    userprofile.UserProfileAddInsertCallback(func(obj xuanwu.IXuanWuObj) {
        sendNoticeToUser(obj.(*userprofile.UserProfile))
    })
}
```

- 泛型、ORM、流程、接口封装等等场景均可考虑“元编程”
- 显然也不只上述场景
- 把“元编程”视为一种重构手段
    - 先手动写一遍代码
    - 把反复出现、类似的代码抽象为模板
    - 选择DSL
- Python简单、库资源丰富；可以很容易的验证“元编程”的想法

“元编程”的难点

- 选择合适的DSL
- 提供高层次抽象能力
- 维持细节定制能力
  - www.joelonsoftware.com/articles/LeakyAbstractions.html

代码生成

- 解析DSL调用模板生成代码的模式很容易实现
- 生成的都是简单代码，容易debug
- 编译器检查代码

反射甚至LISP教?

- 无须重复编译
  - 运行时检查错误
  - 实现更有难度
  - 代码更加精巧、优雅
  - 显然高大上

“When in doubt, use brute force.”

-Ken Thompson

Q & A

其实

- Python不快
  - 比方说，体验过git的极速，就很难忍hg的“不快”
- 元编程最好还是解析语言本身
  - golang.org/pkg/go/parser/
- go generate
  - docs.google.com/document/d/1V03LUfjSADDooDMhe-_K59EgpTEm3V8uvQRuNMAEnjg