Flask 蓝本

一. 为什么使用蓝本
    当开发的web系统越来越大，结构越来越复杂. 比如有了登入登出 权限管理等许多功能模块的时候，如果用传统的方式编写Flask WEB程序，势必我们的Run.py文件会越来越大，代码行数越来越多. 网站的所有路由都写在一个文件里，功能也在一个文件里，这样就很容易出错并且出错后难以定位问题. 而且大型WEB项目一定是多人共同开发，比如A开发登录模块，B开发权限管理模块，那么大家需要同时对run.py进行开发. 此时可能将run.py拷贝多份，那么最终项目上线又面临代码合并的问题，非常不优雅. 此时我们就可以使用蓝本来解决这个问题，蓝本的本质就是让我们的程序更加松耦合 更加灵活 增加复用性 提高查错效率 降低出错的概率.

1.1 蓝本的例子

例：一个项目分为注册 登录 两个部分，我们需要将这两个部分的代码分开，每个功能是一个蓝本，最终把他们合并到Flask APP中.

项目的目录结构如下：

```
(proj_b) [root@backup-platform blue]# tree
.
├── login
│   ├── __init__.py
│   └── views.py
├── main.py
└── regist
    ├── __init__.py
    └── views.py
```

1.1.1 regist目录下的views.py

In [ ]:

```
from flask import Blueprint
bp_regist = Blueprint('bp_regist', __name__)
@bp_regist.route('/')
def regist():
    return 'This is regist Page!'
```

1.1.2 login目录下的views.py

In [ ]:

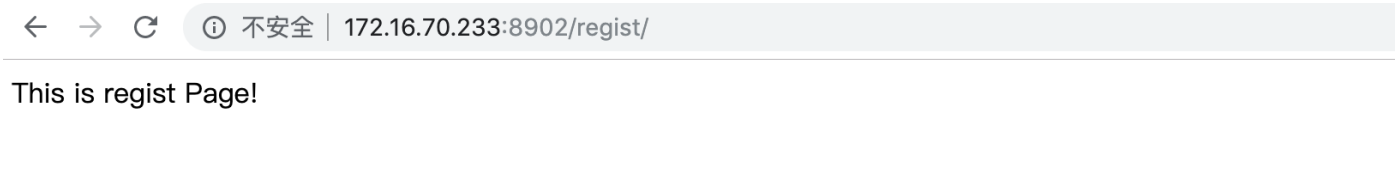
```
from flask import Blueprint
bp_login = Blueprint('bp_login', __name__)
@bp_login.route('/')
def login():
    return 'This is login Page!'
```

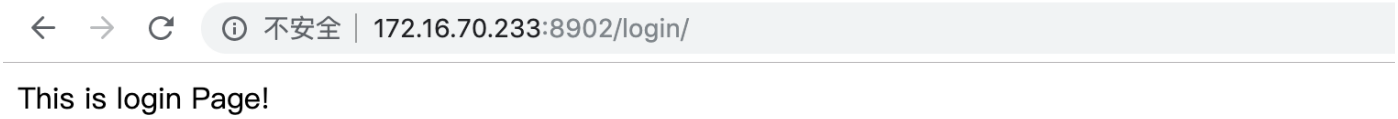
1.1.3 main.py

In [ ]:

```
from flask import Flask
from login.views import  bp_login
from regist.views import bp_regist
app = Flask(__name__)
@app.route('/')
def index():
    return 'This is main Page!'
app.register_blueprint(bp_login, url_prefix='/login')
#将bp_login蓝本注册到app, 并指定url前缀
app.register_blueprint(bp_regist, url_prefix='/regist')
#将bp_regist蓝本注册到app, 并指定url前缀
if __name__ == '__main__':
    app.debug = True
    app.run(host = '0.0.0.0', port=8902)
```

测试网页：





In [ ]: