Flask

Desenvolvendo aplicações Web com o Framework Flask - Aula 3

Organizando o projeto

Flask – Estrutura básica da aplicação

Revisão rápida

```
1  #Inicialização
2  from flask import Flask
3  app = Flask(__name__)
4
5  #Rotas e Funções de Views
6  @app.route("/")
7  def index():
8      return u'Página principal!'
9
10 #Inicialização do Servidor
11 if __name__ == "__main__":
12     app.run()
```

Flask – Organizando o projeto

Introdução

O Flask é tão flexível que deixa a organização do projeto para você. Isso significa que você tem que pensar em como estruturar seu código.

Você pode colocar todo o aplicativo em um único arquivo ou distribuí-lo em pacotes.

Existem alguns padrões de organização que você pode seguir para tornar o desenvolvimento e implantação da aplicação mais fáceis.

Flask – Organizando o projeto

Padrões de organização – Módulo único

Muitos exemplos do Flask que você irá encontrar manterão todo o código em um único arquivo, muitas vezes, o arquivo será o app.py. Isso é bom para pequenos e rápidos projetos. Como exemplo podemos citar os códigos usados para tutoriais e treinamentos em Flask, quando o objetivo é explicar algumas funcionalidades.

```
1  app.py
2  config.py
3  requirements.txt
4  static/
5  templates/
```

app.py => Contém a lógica da aplicação.

config.py => Armazena as configurações.

requirements.txt => Lista as dependências do pacote facilitando a preparação de um ambiente idêntico em outro computador.

static/ => Armazena arquivos estáticos, como CSS, JavaScript e imagens.

templates/ => Local de armazenamento dos modelos.

Flask – Organizando o projeto

Padrões de organização – Pacote

Quando estamos trabalhando em um projeto mais complexo, manter tudo em um único módulo pode ser complicado e desorganizado. Precisaremos definir classes para modelos e formulários, e os mesmos serão misturados com o código de suas rotas e configuração. Isso pode nos deixar um pouco frustrados, porém, podemos resolver este problema, avaliando os diferentes componentes do nosso aplicativo em um grupo de módulos interligado (pacote).

Flask – Organizando o projeto

Padrões de organização – Pacote

Esta estrutura permite agrupar os diferentes componentes do aplicativo de forma lógica. As definições de classe para modelo ficam juntas no arquivo models.py, as definições de rotas estão em views.py e os formulários são definidos em forms.py.

Estes componentes serão encontrados na maioria dos aplicativos, mas provavelmente você terá muitos outros arquivos em seu repositório.

```
config.py
requirements.txt
run.py
instance/
    config.py
yourapp/
    __init__.py
    views.py
    models.py
    forms.py
    static/
    templates/
```

Flask – Organizando o projeto

Padrões de organização – Pacote

run.py	Arquivo invocado para iniciar um servidor de desenvolvimento. Obtém uma cópia do aplicativo do seu pacote e o executa. (não utilizado em produção)
requirements.txt	Lista os pacotes Python que sua aplicação depende. Você pode ter arquivos separados de dependências de produção e desenvolvimento.
config.py	Este arquivo contém a maioria das variáveis de configuração que seu aplicativo precisa.
/instance/config.py	Este arquivo contém variáveis de configuração que não devem estar no controle de versão. Isso inclui coisas como chaves de API and URIs de bancos de dados contendo senhas. Contém também variáveis específicas para essa instância particular da aplicação. Por exemplo, você pode ter DEBUG = False em config.py, mas definir DEBUG = True em instance/config.py em sua máquina local de desenvolvimento. Como este arquivo será lido após o config.py, ele vai substituir o DEBUG = False, por DEBUG = True.
/yourapp/	Este pacote contém sua aplicação.
/yourapp/__init__.py	Este arquivo inicializa seu aplicativo e reúne todos os componentes.
/yourapp/views.py	Este é o arquivo onde as rotas são definidas. Pode ser dividido em um pacote próprio (*yourapp/views/*) com views relacionadas agrupadas em módulos.
/yourapp/models.py	Neste arquivo você define os modelos da sua aplicação. Pode ser dividido em vários módulos da mesma forma que views.py
/yourapp/static/	Este diretório contem CSS público, JavaScript, imagens e outros arquivos que você deseja tornar público através do seu aplicativo. É acessível à partir de yourapp.com/static/ por padrão.
/yourapp/templates/	Neste diretório você colocará os templates Jinja2 para sua aplicação.

```
config.py
requirements.txt
run.py
instance/
    config.py
yourapp/
    __init__.py
    views.py
    models.py
    forms.py
    static/
    templates/
```

Flask – Organizando o projeto

Padrões de organização – Blueprints

Agora imagine a complexidade de ter várias rotas em um mesmo arquivo, seu arquivo views.py pode ficar enorme. Para resolver este problema temos um recurso chamado Blueprints. Usando Blueprints, em vez de termos um arquivo para todas as rotas do sistema, modularizamos as rotas utilizando arquivos separados e registrando Blueprints para cada um deles. Um Blueprint define uma coleção de views, models, arquivos estáticos e outros elementos que podem ser aplicados a um sistema.

Flask – Organizando o projeto

Padrões de organização – Blueprints

Por exemplo, vamos imaginar que temos um blueprint para um painel de administração. Este blueprint definirá as views para rotas como /admin/login e /adm/dashboard. Também pode incluir os modelos e arquivos estáticos que serão atendidos nessas rotas. Podemos então, usar este modelo para adicionar um painel de administração à nossa aplicação, seja uma rede social, CRM ou outra aplicação.

Usando Blueprints podemos organizar nossa aplicação em componentes distintos. Isso nos permite estruturar nosso aplicativo como vários “aplicativos” menores onde cada um faz uma coisa distinta.

Flask – Organizando o projeto

Padrões de organização – Blueprints

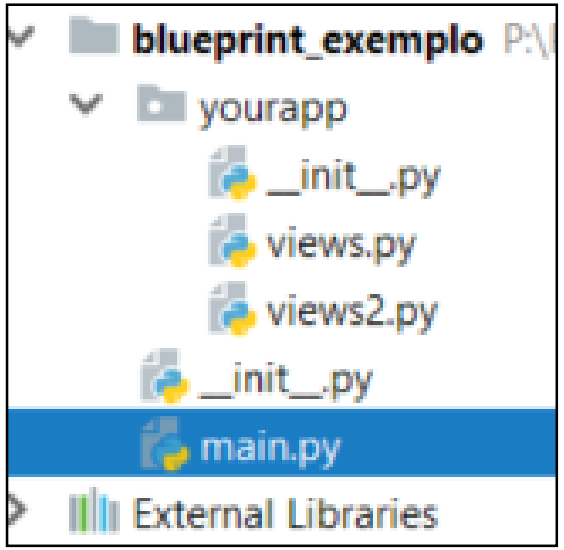
Usando Blueprints, em vez de termos um arquivo para todas as rotas do sistema, modularizamos as rotas utilizando arquivos separados e registrando Blueprints para cada um deles.

Um Blueprint funciona de forma parecida com o objeto de aplicação Flask, mas não é a mesma coisa. Poderiam ser criadas várias aplicações separadas, mas elas não compartilham as mesmas configurações e seriam tratadas na camada WSGI e não na camada do Flask. Já os Blueprints compartilham as mesmas configurações e são uma forma de separar funções dentro da mesma aplicação.

Flask – Organizando o projeto

Padrões de organização – Blueprints

Segue exemplo de como registrar um Blueprint para cada módulo do sistema. Você pode organizar seu projeto em um pacote como a seguir, criar os arquivos e executar o sistema.




```python
from flask import Flask
from yourapp.views import views
from yourapp.views2 import views2

app = Flask(__name__)
app.register_blueprint(views)
app.register_blueprint(views2)

if __name__ == '__main__':
    app.run()
```

```python
from flask import Blueprint

views = Blueprint('views', __name__)

@views.route('/')
def index():
    return '<h1>Olá mundo!</h1>'
```

```python
from flask import Blueprint

views2 = Blueprint('views2', __name__)

@views2.route('/teste')
def teste():
    return '<h1>Esta é a segunda view!</h1>'
```

FIM