WEB2PY

DESENVOLVIMENTO
WEB ÁGIL COM PYTHON

O que é web2py?

web2py é um framework para desenvolvimento web escrito em **Python**, software livre e gratuito, que tem como um de seus principais objetivos **agilidade no desenvolvimento de aplicações web** seguras, baseadas em banco de dados. O framework segue o modelo MVC (Model-View-Controller), que permite melhor organização do código; ele também é autocontido: tudo o que você precisa para desenvolver alguma aplicação está nele: basta baixar e descompactar para começar - **nada de configurações!**

Com o foco em deixar o desenvolvedor pensar apenas na aplicação que está desenvolvendo, o web2py possui integração com mais de 10 sistemas de banco de dados e vários subssistemas: criação automática de formulários com validação automática, autenticação e autorização, gerador de códigos AJAX para melhor interação do usuário com a aplicação, upload seguro de arquivos, sistema de plugins, integração com vários padrões Web (XML, RSS etc.), grid, busca, agendador de tarefas, dentre outros.

A melhor maneira de conhecer o framework é na prática! Siga este breve tutorial e desenvolva seu primeiro web app com Python e web2py:

1 - Download e inicialização

O web2py não requer instalação. Efetue o download e inicialize o servidor de desenvolvimento através do interpretador Python (preferencialmente versão 2.7). Apesar de oferecer pacotes especiais para inicialização em ambientes Mac e Windows, recomendamos sempre o uso da versão "source code" que pode ser baixada em: **http://www.web2py.com/examples/static/web2py_src.zip**

Extraia os arquivos do pacote web2py_src.zip e então acesse o terminal (console) de seu sistema operacional e digite o seguinte comando:

```
cd pasta_onde_descompactou/web2py
python web2py.py -a 1234 -p 8000 -i 0.0.0.0 --nogui
```

No comando, você utiliza o interpretador **Python*** para iniciar o servidor de desenvolvimento e passa os argumentos **"-a 1234"** (que define a senha que será utilizada para acessar a interface admin), **"-p 8000"** (que define a porta TCP a ser utilizada), **"-i 0.0.0.0"** (que define que o servidor estará acessível através de todas as interfaces de rede) e **"--nogui"** (para executar em modo terminal).

*Veja a lista completa de comandos de inicialização utilizando "python web2py.py -h"

Não feche a tela do terminal (apenas minimize). O terminal ficará ativo durante todo o tempo de desenvolvimento e através dele você acompanhará mensagens de log e debug do servidor de desenvolvimento.

*Para executar o servidor de desenvolvimento é necessário ter o interpretador Python (2.6 ou 2.7) instalado em seu computador.

2 - Interface administrativa e web IDE

No navegador web acesse **http://localhost:8000/admin** e entre com a senha **"1234"**. Esta é a interface admin do desenvolvedor - através dela você poderá criar e dar manutenção aos apps efetuando tarefas como compilação (se necessário), deploy, controle de versão, testes, acompanhamento de cache e ticket de erros, tradução e até mesmo **editar os arquivos de seu app diretamente no browser sem a necessidade de utilizar uma IDE ou editor de texto!** É importante ressaltar que esta interface é apenas para o desenvolvedor e não deve ser exposta a usuários, por isso seu acesso é **100% seguro!** Outro detalhe é que a interface admin é opcional e você poderá utilizar qualquer editor ou IDE de sua preferência.

No formulário **"New simple application (Nova aplicação básica)"** digite "**microblog**" e clique em **"create (criar)"**. Na próxima página você irá desenvolver um app de micro blogging.

CONTINUE >>

A estrutura padrão dos aplicativos web2py é **MVC**, onde em "**model**" colocamos nossa estrutura de dados e seus métodos. Em "**controllers**" colocamos as actions que serão roteadas como páginas de seu app e receberão os argumentos para controlar o fluxo e controle de acesso. Em "**views**" criamos os templates utilizados para apresentar a informação.

3. Edite o arquivo **"web2py/applications/microblog/models/db.py" e** substitua todo o conteúdo do arquivo
(você pode fazer isso através da interface admin ou utilizando qualquer editor de textos)

MODEL

```
from gluon.tools import Auth, Service, prettydate #  Os toolkits do web2py ficam no módulo gluon
# Alguns objetos como DAL, request, response,  session e outros são builtin e não precisam de importação
db = DAL('sqlite://microblog.sqlite') #  Conecta com um novo banco de dados SQLite
response.generic_patterns = ['*'] #  Libera o acesso a views genéricas
auth = Auth(db) # O web2py já possui um sistema de autenticação e autorização RBAC
auth.define_tables(username=False, signature=False)
service = Service() #  E também um sistema completo para publicação de webservices

# a DAL irá fazer a migração automática do modelo de dados e as tabelas serão criadas e alteradas automaticamente!

Post = db.define_table("blog_post",  #  nome da tabela
                Field("author", "reference auth_user"),  # foreign key com a tabela de usuários
                Field("body", "text"),  #  campo de texto
                auth.signature )  # campos para auditoria do registro

Post.body.requires = [IS_NOT_EMPTY(error_message="Campo obrigatório"), IS_LENGTH(140)]  # Validadores de entrada
```

4. Edite agora o arquivo **"web2py/applications/microblog/controllers/default.py"** e inclua o seguinte conteúdo ao final do arquivo:

CONTROLLER

```
auth.settings.login_next = URL("timeline") # Redireciona o usuário após o login
@auth.requires_login() # exige que o usuário esteja logado
def timeline():  # cria uma rota para http://localhost:8000/microblog/default/timeline
    Post.author.default = auth.user_id # Define o usuário logado como padrão para novas postagens
    Post.author.writable = Post.author.readable =  False # altera proteção de acesso ao campo
    form = SQLFORM(Post, formstyle="divs", submit_button="Postar") # cria um formulário baseado na tabela
    if form.process().accepted: # Processa e valida o formulário
        response.flash = "Mensagem postada com sucesso!"
    posts = db(Post).select(orderby=~Post.created_on)  # Seleciona todas as postagens existentes ordenadas por data
    if request.extension in ['json', 'xml']:  return dict(posts=posts.as_list()) # garante a formatação para json e xml
    return dict(form=form, posts=posts) # retorna um dicionário para a view html
```

5. Para finalizar, crie um arquivo novo em
"web2py/applications/microblog/views/default/timeline.html"

VIEW

```
{{extend 'layout.html'}} <!-- herança de layout -->
<style>
ul {list-style:none;} li {border-bottom:1px solid black; margin:10px;}
form {width:100%;margin:30px;} textarea {width:500px; height:50px;}
</style>
<h1> Timeline de {{="%(first_name)s %(last_name)s" % auth.user}}</h1>
{{=form}} <!-- Gera o html para o objeto form -->
<hr>
<ul>
{{for post in posts:}} <!-- o web2py permite sintaxe Python no template -->
  <li>{{=post.body}}<br> <!-- acesso a propriedades -->
    <small>{{=prettydate(post.created_on)}}</small></li> <!-- formata data -->
{{pass}} <!-- pass é utilizada para finalizar um bloco Python -->
</ul>
```

PRONTO!

Agora acesse a url
http://localhost:8000/microblog/default/timeline

O sistema de autenticação irá solicitar que registre um novo usuário e faça login

Poste uma mensagem maior que 140 caracteres para ver a validação em funcionamento

Experimente alterar a extensão da URL para servir em outros formatos utilizando as views genéricas

microblog/default/timeline.**json**
microblog/default/timeline.**xml**
microblog.default/timeline.**pdf**