



## Álvaro Justen

Graduando em Engenharia de Telecomunicações pela UFF (Universidade Federal Fluminense). Desenvolve software de monitoramento inteligente de datacenters na Intelie, é apaixonado por software livre e faz parte do time de desenvolvimento do web2py.

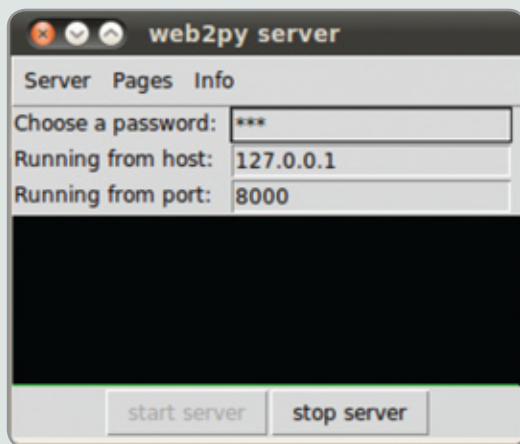
Blog: [blog.justen.eng.br](http://blog.justen.eng.br) | Twitter: @turicas | E-mail: [alvaro@justen.eng.br](mailto:alvaro@justen.eng.br)

## web2py

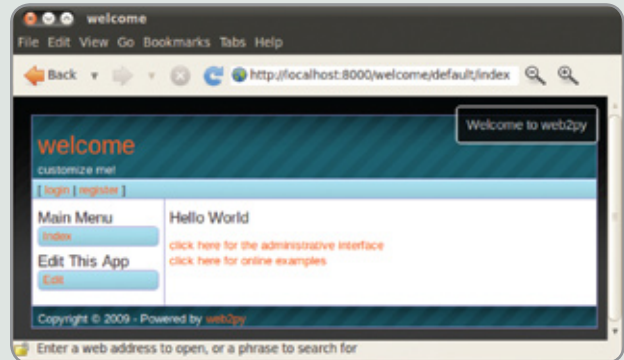
O web2py é um framework inovador para desenvolvimento de aplicações web escrito em Python, cujo principal foco é a agilidade no desenvolvimento de aplicações baseadas em banco de dados, seguras, escaláveis e portáteis. Por ser escrito totalmente em Python, ele é multiplataforma: roda em GNU/Linux, Mac OS, Windows, JVM (Jython), Google App Engine (nativamente), entre outros.

Fortemente baseado em funcionalidades dos frameworks Django e Ruby on Rails, o web2py é um novato no cenário de frameworks web: sua primeira versão estável foi lançada em outubro de 2007 pelo italiano Massimo Di Pierro. Desde seu lançamento, é software livre (GPL2) e todas as novas versões são compatíveis com as anteriores. Apesar de novo, o framework já vem fazendo sucesso lá fora (e agora no Brasil) – o sistema de inscrição, pagamento e palestras da PyCon (evento internacional de Python), por exemplo, foi desenvolvido utilizando o web2py.

Para falar mais das características do framework, vou mostrar exemplos de código de uma aplicação que armazena seleções de futebol e seus respectivos jogadores. O sistema que utilizei para desenvolver foi o Ubuntu 10.04. Antes de começar a escrever código, precisamos iniciar o web2py. Para isso, baixe-o em [www.web2py.com](http://www.web2py.com), descompacte-o e, dentro do diretório "web2py", rode o comando `python web2py.py`. Será, então, aberta a tela abaixo:



Insira uma senha para acesso à aplicação de administração e clique em "start server". Seu navegador será direcionado para a aplicação **welcome**:



O web2py incorpora, em sua distribuição, três aplicações: **admin**, **examples** e **welcome**. **Admin** é utilizada para administrar as aplicações instaladas; **examples** são exemplos de código; e **welcome** é a aplicação-modelo, na qual vamos basear nosso exemplo. Para desenvolvimento, utilizamos o servidor web embutido no framework, porém poderíamos utilizar o Apache, o Lighttpd, o Cherokee ou qualquer outro que suporte WSGI, CGI/FastCGI ou atue como proxy.

Através da interface administrativa, vamos criar a aplicação **copa2010**, que ficará acessível através da URL `http://localhost:8000/copa2010`. Como o web2py é baseado em MVC (Model View Controller), começaremos criando e configurando nosso modelo e nossa persistência de dados. Adicione ao arquivo `applications/copa2010/models/db.py`:

```
db = DAL('sqlite://copa_do_mundo.sqlite') #Conecta ao banco SQLite
Selecao = db.define_table('selecao',
    #Tabela de seleções
    Field('pais', 'string',
        length=100), #País de origem da
        seleção
    Field('favorita', 'boolean'), #É
        favorita para esse mundial?
)
Jogador = db.define_table('jogador',
    #Tabela de jogadores
    Field('nome', 'string',
        length=250), #Nome do jogador
    Field('posicao', 'string',
        length=50, label='Posição'), #Posição
```

```

em campo
    Field('selecao', Selecao,
label='Seleção'), #Chave estrangeira
)
# Validador: Jogador.posicao somente
aceita um dos valores abaixo
posicoes = ['Goleiro', 'Defesa',
'Meio-campo', 'Ataque']
Jogador.posicao.requires = IS_IN_
SET(posicoes)
# Validador: Jogador.selecao tem que
coincidir com algum Selecao.id
Jogador.selecao.requires = IS_IN_
DB(db, Selecao.id, '%(pais)s')

```

Não se preocupe em criar ou alterar o banco, as tabelas ou mesmo com SQL: o web2py faz tudo automaticamente para você. A camada de abstração de banco de dados (DAL) suporta atualmente dez tipos de bancos de dados: SQLite, PostgreSQL, MySQL, MSSQL, FireBird, Oracle, IBM DB2, Informix, Ingres e Google App Engine (BigTable).

Vamos agora criar um *controller* para inserção, edição e listagem de seleções. Adicione ao arquivo `applications/copa2010/controllers/selecao.py`:

```

def index():
    "Retorna seleções cadastradas"
    return {'selecoes': db().
select(Selecao.ALL, orderby=Selecao.
pais)}
#O retorno do controller será
utilizado na view, como veremos
abaixo
def edita():
    "Retorna e processa formulário de
inserção/edição de seleções"
    selecao = request.args(0) #Pega
primeiro parâmetro passado por URL
    if selecao:
        form = SQLFORM(Selecao, selecao)
#Formulário de alteração de registro
    else:
        form = SQLFORM(Selecao)
#Formulário de inserção de registro
    if form.accepts(request.post_vars,
session): #Valida e insere (caso ok)
mensagem = 'Team added!'
    if selecao is None else 'Team

```

```

modified!' if selecao is None else
'Team modified!'
    response.flash = T(mensagem) #Mostra
mensagem usando tradução
    return {'form': form, 'editando':
selecao is not None}

```

A função *index* chama o objeto *db* (criado no *model*) e seleciona todos os registros de *selecao*; a função *edita* cria um formulário de edição ou inserção, dependendo do valor passado pela URL – após a criação do formulário, ela faz a verificação de validação do mesmo e, caso positivo, insere/atualiza o registro no banco de dados.

A função *T* é responsável pela tradução da aplicação. O web2py detecta quais línguas o navegador do usuário suporta e então utiliza um dicionário interno para traduzir as *strings*. Dessa forma, podemos construir nossa aplicação toda em uma só língua e, ao final, traduzir as strings para diversas outras, dando suporte a vários idiomas sem precisar modificar o código-fonte.

O *controller* acima poderia ser alvo de ataques como *SQL Injection* ou outros problemas de segurança. Não se preocupe, o web2py implementa boas práticas e protege sua aplicação das falhas de segurança mais comuns na web, para que você foque seus esforços exclusivamente em funcionalidades que agreguem valor à aplicação que está desenvolvendo.

O web2py presa por agilidade e, por isso, os conceitos *Don't Repeat Yourself* e *Convention over Configuration* estão sempre presentes nas diversas partes do framework, como na segurança, citada acima. Um outro exemplo é o fato de que em nenhum momento é preciso importar os objetos *db*, *SQLFORM*, *T* etc. – tudo é importado automaticamente.

Vamos agora criar a parte responsável pela apresentação das funções acima. Crie os arquivos `index.html` e `edita.html` em `copa2010/views/selecao`:

```

{{extend 'layout.html'}}
<h1>Listagem de Seleções</h1>
<a href="{{=URL(r=request,
f='edita')}}">Cadastre uma seleção</
a><br />
{{if len(selecoes) == 0:}}
Ainda não existem seleções

```

```

cadastradas!
{{else:}}
Clique em uma seleção para editá-la:
<table>
  <tr> <th>País</th> <th>Favorita?</
th> </tr>
  {{for selecao in selecoes:}}
  <tr>
    <td><a href="{{URL(r=request,
f='edit', args=selecao.
id)}}"{{selecao.pais}}</a></td>
    <td>{{='Sim' if selecao.favorita
else 'Não'}}</td>
    <td>{{='Sim' if selecao.favorita
else 'Não'}}</td>
  </tr>
  {{pass}}
</table>
{{pass}}

```

```

{{extend 'layout.html'}}
{{if editando:}}
<h1>Editar Seleção</h1>
Modifique os dados abaixo:
{{else:}}
<h1>Cadastro de Seleção</h1>
Insira os dados da nova seleção:
{{pass}}
{{=form}}
<a href="{{URL(r=request,
f='index')}}">Voltar para listagem</a>

```

Tudo que está entre `{{ e }}` é código Python. No arquivo `index.html`, herdamos o *design* padrão da aplicação (`layout.html`) e listamos as seleções, caso existam; a variável `selecoes` vem do *controller*. No arquivo `edita.html`, exibimos o título de edição ou cadastro, dependendo do valor da variável `editando` que foi passada pelo controller, e então exibimos o formulário.

A classe `SQLFORM` do `web2py` detecta automaticamente os campos que definimos no *model* e cria os componentes de formulário HTML necessários para exibir cada campo, como, por exemplo: um `input text` para o nome do país (*string*) e uma `checkbox` para favorito (*boolean*).

Para o cadastro de jogadores, o código é praticamente o mesmo. Por isso, economizarei código aqui e mostrarei apenas as telas finais:

País	Favorita?
Argentina	Não
Brasil	Sim
França	Não
Itália	Não

Nome	Posição	Seleção
Julio César	Goleiro	Brasil
Kleberison	Meio-campo	Brasil
Lúcio	Defesa	Brasil
Robinho	Ataque	Brasil

Além das funcionalidades citadas, o `web2py` possui muitas outras também necessárias ao desenvolvimento de aplicações web, como gerenciamento de sessões, autenticação e autorização, views genéricas, cache (RAM, disk e memcache), CRUD, mail, AJAX com jQuery, gerenciamento de exceções através da geração de tickets, entre outras – porém não estão no escopo deste artigo.

Se você está procurando um framework para economizar tempo de desenvolvimento e assegurar qualidade à sua aplicação, além de curto tempo de aprendizado, o `web2py` é uma ótima alternativa.

Aprenda mais sobre o `web2py`: entre na lista de usuários brasileiros ([tinyurl.com/web2py-br](http://tinyurl.com/web2py-br)) e assista ao *screencast* que criei exclusivamente para os leitores da TIdigital, abordando todos os passos descritos acima, com exemplos e explicações extras. O endereço é: [tinyurl.com/web2py-tidigital](http://tinyurl.com/web2py-tidigital).