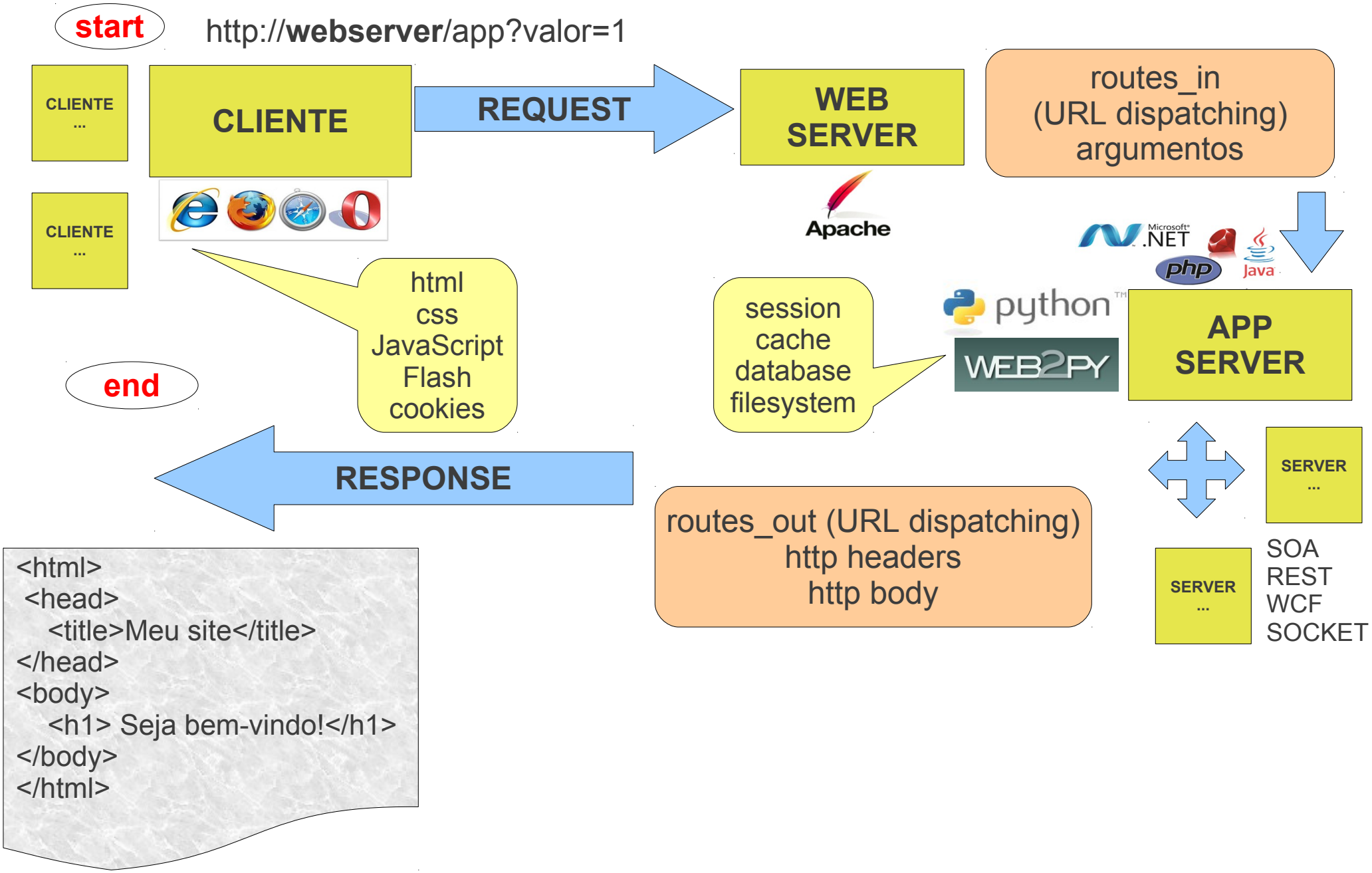
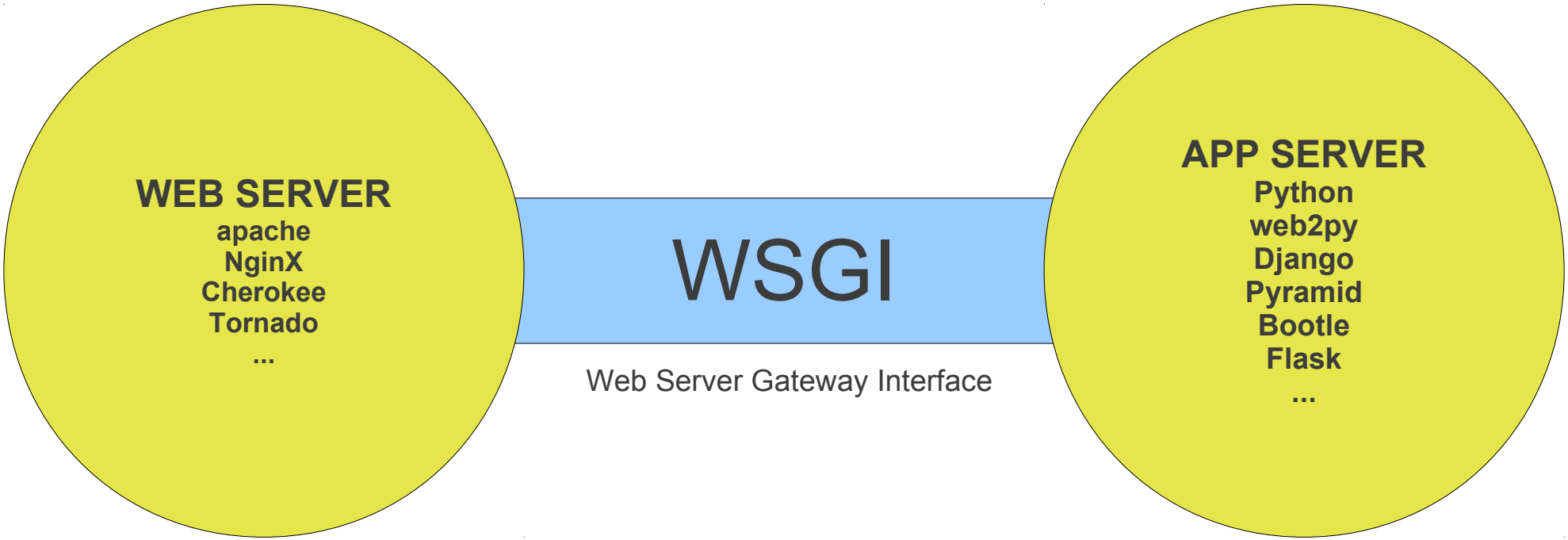
Curso de python .com.br

Aula 1 – Desenvolvimento web com Python utilizando web2py

- Visão geral sobre web e http
- WSGI
- web2py framework
- MVC + Modules
- Roteamento, URLS e argumentos
- DAL (Database Abstraction Layer)
- FORMS & VALIDATORS
- HTML HELPERS
- Autenticação
- Enviando Email
- Views escrevendo Python template
- serviços web

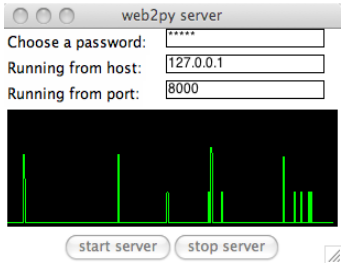
CursoDePython – [3,1]
WEB

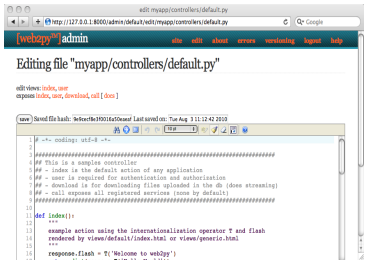



WEB SERVER
apache
NginX
Cherokee
Tornado
...
WSGI
Web Server Gateway Interface
APP SERVER
Python
web2py
Django
Pyramid
Bootle
Flask
...

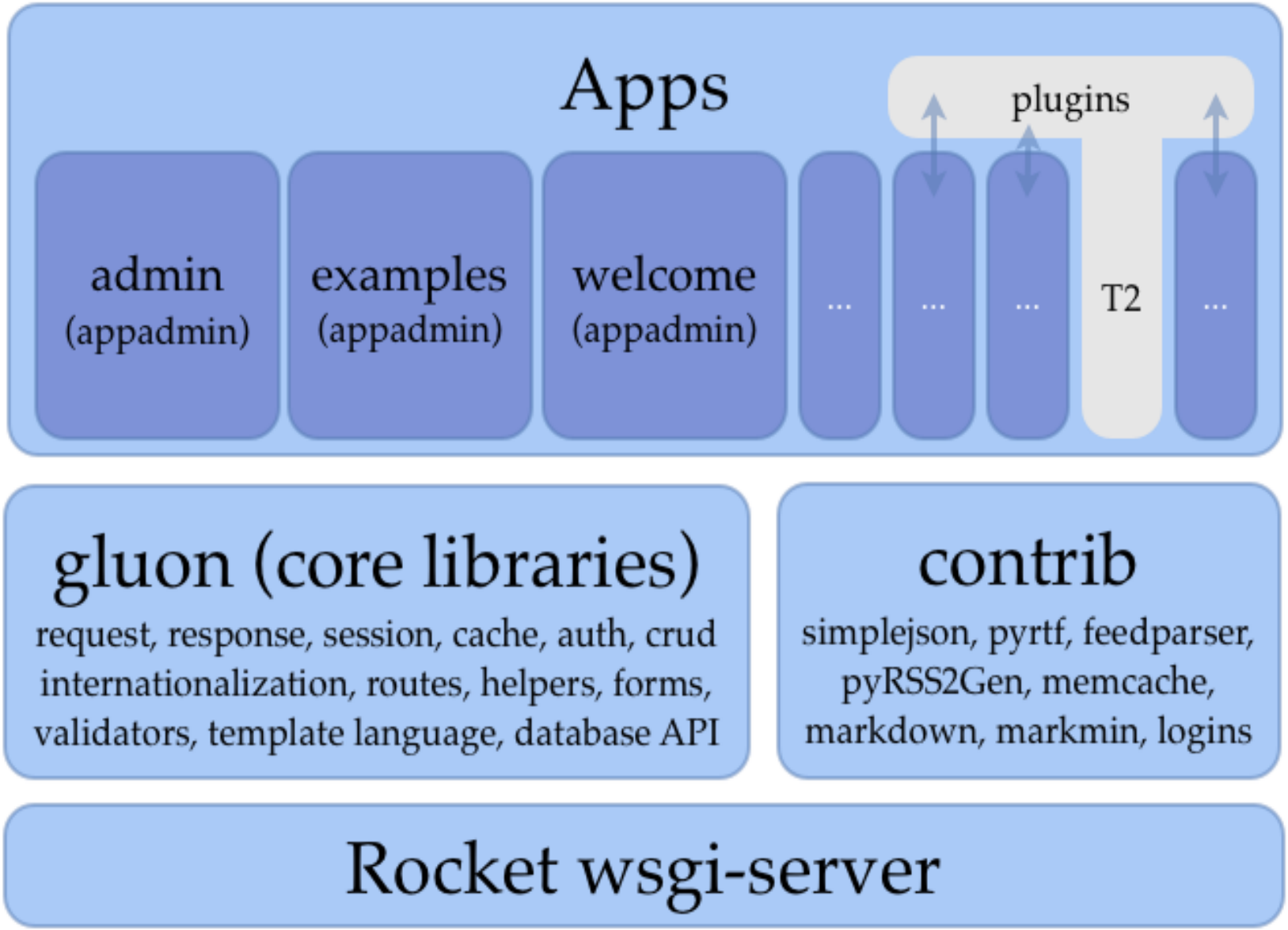
”Framework é um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação.”

– Fayad Schmidit





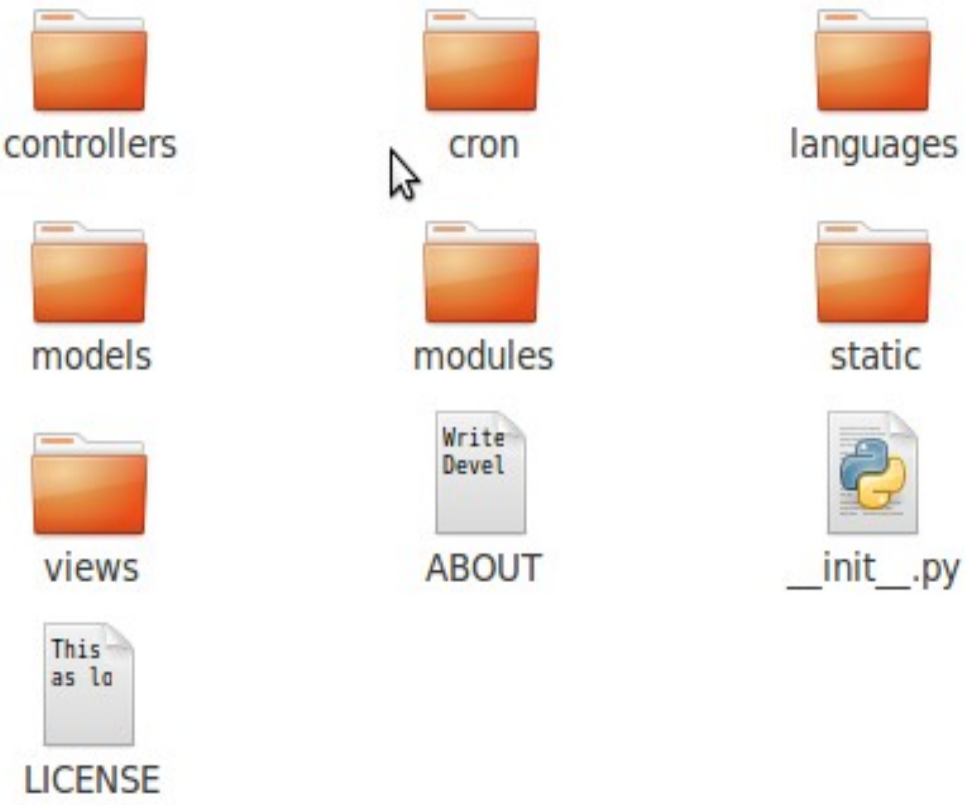
- Não requer instalação
- Web server integrado
- Suporte a multiplos bancos de dados
- Camada de abstração de acesso a dados
- Padrão MVC
- Padrão WSGI
- MultiPlataforma
- Geradores de formulários, uploads, validadores
- Segurança e controle de acesso
- Web services
- Interface administrativa
- Editor de códigos via web
- Sistema de testes unitários
- Sistema de ticket/log para erros
- Sistema de plugins
- Compatibilidade ETERNA!


Apps
plugins
admin
(appadmin)
examples
(appadmin)
welcome
(appadmin)
...
...
...
T2
...
gluon (core libraries)
request, response, session, cache, auth, crud
internationalization, routes, helpers, forms,
validators, template language, database API
contrib
simplejson, pyrtf, feedparser,
pyRSS2Gen, memcache,
markdown, markmin, logins
Rocket wsgi-server

CursoDePython – [3,1]
APLICAÇÃO WEB2PY - MVC

- **Models**
  - definição de modelos de dados.
- **Controllers**
  - Programação do controle de fluxo.
- **Modules**
  - Classes, funções (a programação dos objetos da aplicação)
- **Views**
  - Interface da aplicação, apresentação, estilos, layouts.
- **Static**
  - Arquivos estáticos (não Python) css, js, imagens.
- **Languages**
  - Traduções da aplicação
- **Cron**
  - Agendador de tarefas

- **Databases**
  - Arquivos de controle do banco de dados
- **Uploads**
  - Arquivos enviados pelos usuários da aplicação
- **Sessions**
  - Arquivos de controle de sessões físicas
- **Cache**
  - Cache de disco
- **Errors**
  - tickets/logs de erro



http://servidor:8000/aplicação/default/index/56/?valor=1

/**models**/qualquercoisa.py

```
import modulox
db = modulo.Database()

db.define_table('pessoa',
    Field('nome'),
    Field('idade'),
)
```

/controllers/**default**.py

```
def index():
    query = db.p.id==request.args(0)
    valor = request.vars.valor
    pessoas = db(query).select()
    ...
    from modulox import Funcoes
    f = Funcoes()
    f.funcao(pessoas)
    ...
    return dict(pessoas=pessoas)
```

/views/default/**index.html**

```
{{extend 'layout.html'}}
...
{{for p in pessoas:}}

<b>{{=p.nome}}</b>

{{pass}}
...
```

/**modules**/modulox.py

```
class Database(object):
   ...
class Funcoes(object):
   ...
```

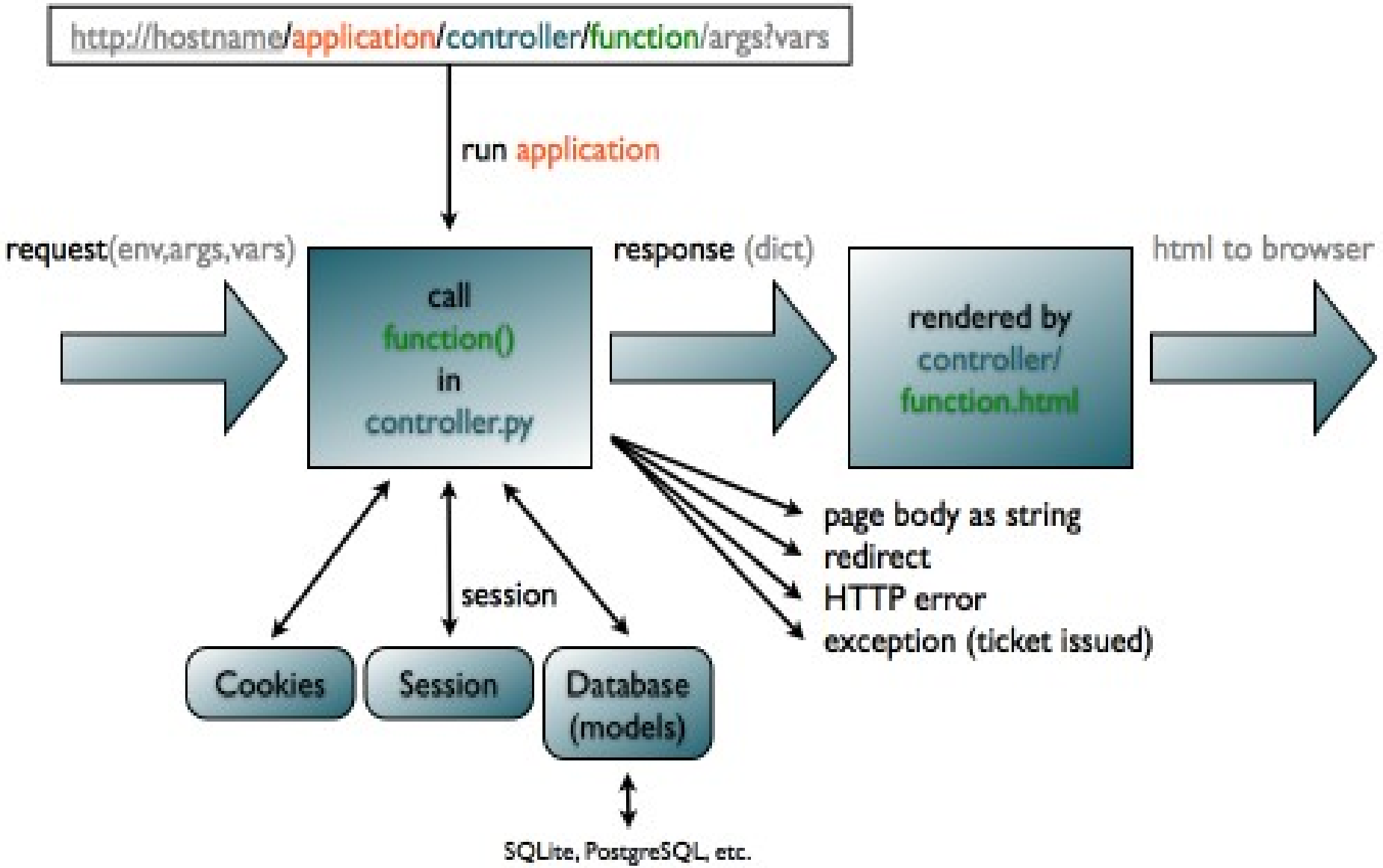
/**static**/style.css

```
# p { color: #ffcc00; }
```

/**views**/layout.html

```
<html>
<link .. style.css>
{{include}}
</html>
```

CursoDePython – [3,1]
ROTEAMENTO



CursoDePython – [3,1]
HTML HELPERS

Considere o seguinte código:

```
DIV('isto',  'é',  'um',  'teste',  _id='123',  _class='minhaclasse')
```

Que será renderizado como:

```
<div id="123" c l a s s="minhaclasse">istoéumteste< /div>
```

DIV, por exemplo, é uma classe helper, um tipo de classe que pode ser usada para criar HTML programaticamente.
Em uma classe helper, os argumentos são interpretados como objetos contidos dentro da tag HTML. Os argumentos nomeados que iniciam com um underline serão interpretados como atributos. Alguns helpers recebem argumentos nomeados que não iniciam com underline; estes argumentos são específicos para cada tipo de TAG

Um do mais importantes é o helper URL, que automatiza o processo de criação de URLs de acordo com as rotas.

```
A(”LINK”, _href=URL('default','index',args=['Bruno',2],vars={'valor':20}))
```

será renderizado como:

```
<a href='http://server:porta/aplicacao/default/index/Bruno/2/?valor=20'>LINK</a>
```

CursoDePython – [3,1]
DAL (Acesso a Dados)

DAL
É a própria classe de abstração a acesso a dados
define a conexão com o banco de dados.

```
db = DAL('sqlite://banco.sqlite')
```

Table e Field
Classes que representam a tabela e seus campos.

```
db.define_table('pessoa', Field('nome'))
```

Query
Representa uma cláusula SQL **SELECT**...**WHERE**
para consulta ao banco

```
query = (db.pessoa.nome=='Bruno')
query2 = (db.pessoa.nome!='John')
```

Set
Representa um comando SQL .
que pode ser composto por uma ou mais queries

```
nerds = db(query & query2)
pessoas = nerds.select()
nerds.update(nome='xyz')
nerds.delete()
```

Expression
Conjunto de expressões SQL como orderby e limitby

```
ordem = db.nome.upper() | db.nome.id
db(query).select(orderby=ordem)
```

Rows e Row
Classes que representam um conjunto de registros
ou um registro individual

```
pessoas = db(query).select()
for p in pessoas: print p.nome

p = db(query).select().first()
```

definir ou alterar a lista de campos ou tipos de campo em um modelo, provoca uma migração automática, ou seja, web2py gera SQL para alterar a tabela de acordo.

Se a tabela não existir ele é criada. Ações de migração são registradas no arquivo Sql.log acessível através da interface administrativa. A migração pode ser desligada por tabele em uma base de dados, passando o parâmetro migration = False no método define_table.

```
db.define_table('empresa',
    Field('CNPJ','string', length=18, unique=True),
    Field('atividade','text'),
    Field('logotipo','upload'),
    migrate = True
)
```

Relacionamento
UM para **MUITOS**

Para todas as tabelas a DAL cria automaticamente o campo 'id' como um auto-incremento.

Podemos definir o id da seguinte maneira: Field('meu_id','id')
para isso migrate deve ser False

```
db.define_table('pessoa',
    Field('nome','string', length=50, unique=True),
    Field('idade','integer'),
    Field('foto','upload'),
    Field('empresa', db.empresa),
    migrate = True
)
```

CursoDePython – [3,1]
FORMULÁRIOS

FORM é um helper que retorna um formulário html puro e permite que seja efetuada a validacão.

```
form = FORM('Seu nome:', INPUT(_name='name'), INPUT(_type='submit'))
```

SQLFORM é um helper que cria um formulário baseado em uma tabela de nosso modelo de dados, faz a validação e insere os dados na tabela automaticamente.

```
form = SQLFORM(db.pessoa, formstyle='divs')
```

CUSTOM FORMS é uma maneira para customizar nas views os formulários criados automaticamente.

```
{{=form.custom.begin}}
    <b>Nome</b><div>{{=form.custom.widget.name}}</div>
{{=form.custom.end}}
```

CRUD é um helper que automatiza a criação de formulários para consulta, inserção, edição, busca e remoção de registros.

```
form = crud.create(db.pessoa)
updateform = crud.update(db.pessoa, id_do_registro)
searchform = crud.search(db.pessoa)
```

CursoDePython – [3,1]
VALIDATORS

Validadores são classes usadas para validar campos de entrada em formulários (incluindo os gerados através do banco de dados).

Exemplo de uso de um validador na criação de um formulário.

```
INPUT(_name='marca',  requires=IS_NOT_EMPTY(error_message='Você deve informar a marca!') )
```

Aqui um exemplo de uso de validador na base de dados:

```
banco.define_table('carros',  Field('marca') )
banco.carros.marca.requires = IS_NOT_EMPTY( )
```

Os validadores sempre são atribuídos através do atributo requires da classe Field. Um campo pode ter um ou mais validadores. Múltiplos validadores são formados com uma lista

```
bancos.carros.marca.requires =   [IS_NOT_EMPTY( ) , IS_NOT_IN_DB(banco, 'carros.marca') ]
```

Os validadores são invocados através da função accepts em um **FORM** ou qualquer outro **HTML** helper que contenha um **FORM**. São invocados na exata ordem em que são listados.

CursoDePython – [3,1]
Acesso e Autenticação

```
from gluon.tools import Auth
auth = Auth(db)
auth.define_tables(username=False)
```

```
@auth.requires_login()
def minhapagina():
    dados = 'segredo'
    return dict(dados=dados)
```

```
@auth.requires_membership('grupo')
def minhapagina():
    dados = 'segredo'
    return dict(dados=dados)
```

```
@auth.requires(auth.user_id==1 or request.client=='127.0.0.1')
def minhapagina():
    dados = 'segredo'
    return dict(dados=dados)
```

O web2py possui uma classe especializada no envio de emails, esta classe funciona com qualquer servidor smtp e já vem pré configurada para trabalhar com o servidor do gmail.

```
from gluon.tools import Mail
mail = Mail()

mail.settings.server = 'smtp.gmail.com:587'
mail.settings.sender = 'seuemail@gmail.com'
mail.settings.login = 'seuemail@gmail.com:suasenha'
```

```
mail.send(
    to = 'destino@email.com',
    cc=['alguem@dominio.com'],
    bcc=['pessoa@oculta.com'],
    reply_to='alguem@dominio.com ',
    subject='Assunto do email',
    message=["texto puro", "<b>texto em html</b>"],
    attachments = [Mail.Attachment('/pasta/arquivo.zip', content_id='zip')]
    )
```

CursoDePython – [3,1]
VIEWS

Views são as estruturas de hipertexto que criam a interface de apresentação da aplicação.

Podem ser feitas nos formatos: html, xml, rss, json, pdf, xls, xlsx, atom ou customizadas.

Tudo o que é escrito entre os caracteres {{ e }} é executado como código Python, qualquer coisa fora destas tags será apenas apresentado sem executar.

Views podem herdar um layout atraves da instrução {{extend}} podem embutor um arquivo html externo com {{include}} e possuem blocos {{block …}} {{end}}

```
def index():
    ...
    return dict(var=value)
```



```
{{extend 'layout.html'}}
…
{{for i in range(10):}}

<b>{{=var}}</b>

{{pass}}
…

{{=DIV(var)}}
```

*A imagem é apenas uma brincadeira, na verdade todos amamos Java, Ruby, Perl e C# nem se fala!

Curso de Python

Nível 3 (Básico) - Aula 1
(desenvolvimento web com web2py)

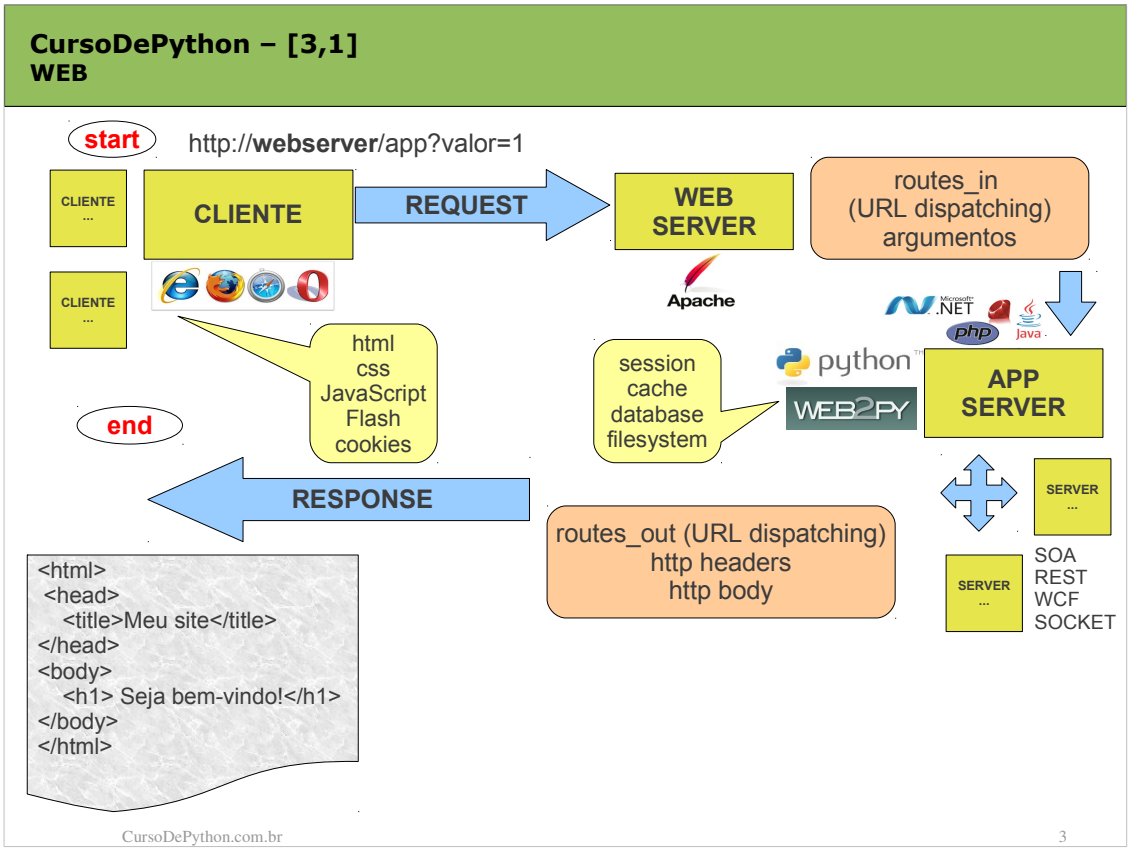
Bruno Cezar Rocha
bruno@blouweb.com

blouweb.com Consultoria Digital.

CursoDePython – [3,1]

Aula 1 – Desenvolvimento web com Python utilizando web2py

- Visão geral sobre web e http
- WSGI
- web2py framework
- MVC + Modules
- Roteamento, URLS e argumentos
- DAL (Database Abstraction Layer)
- FORMS & VALIDATORS
- HTML HELPERS
- Autenticação
- Enviando Email
- Views escrevendo Python template
- serviços web



Visualizar uma página web ou outro recurso disponibilizado normalmente inicia ou ao digitar uma URL no navegador ou seguindo (acessando) uma hiperligação.

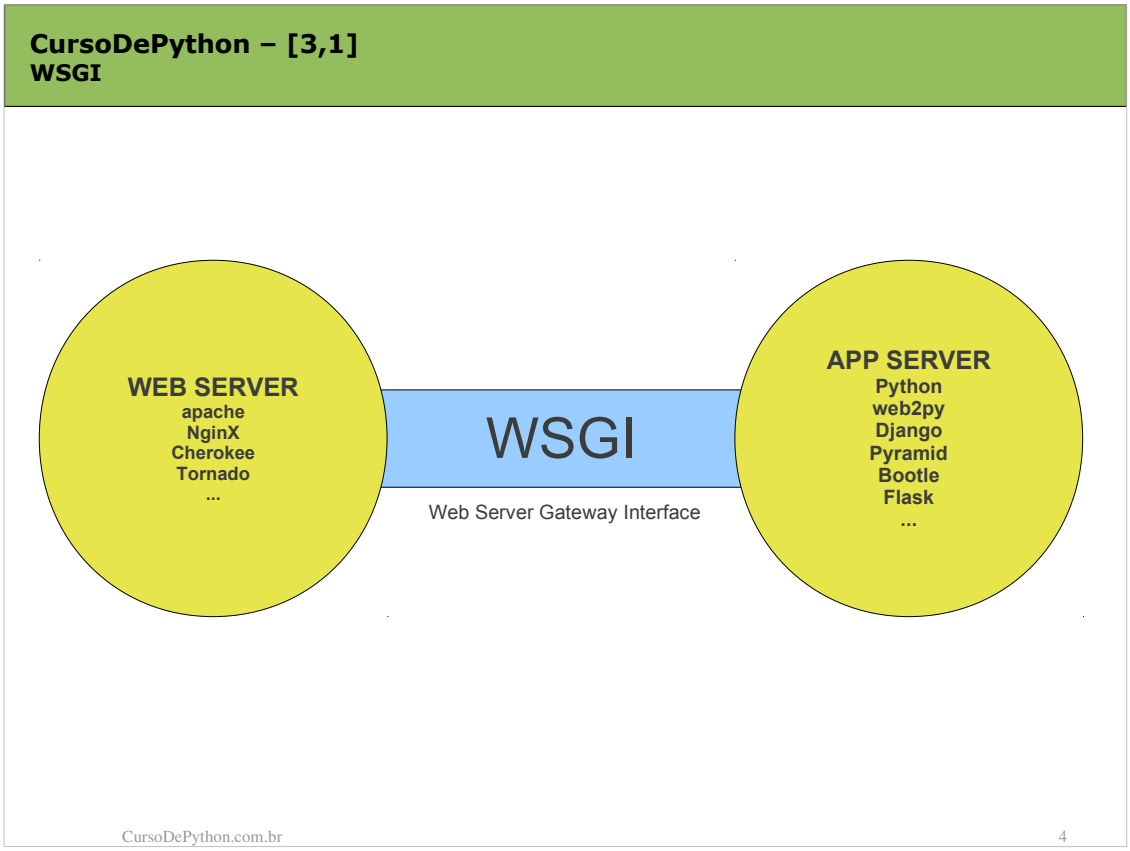
Primeiramente, a parte da URL referente ao servidor web é separada e transformada em um endereço IP, por um banco de dados da Internet chamado Domain name system (DNS).

O navegador estabelece então uma conexão TCP-IP com o servidor web localizado no endereço IP retornado.

O próximo passo é o navegador enviar uma requisição HTTP[8] ao servidor para obter o recurso indicado pela parte restante da URL (retirando-se a parte do servidor).

No caso de uma página web típica, o texto HTML é recebido e interpretado pelo navegador, que realiza então requisições adicionais para figuras, arquivos de formatação, arquivos de script e outros recursos que fazem parte da página.

O navegador então renderiza a página na tela do usuário, assim como descrita pelos arquivos que a compõe.



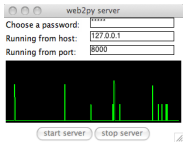
A Web Server Gateway Interface (**WSGI**) é uma especificação para comunicação entre servidores de aplicação, ou servidores web, e aplicações web como definida na PEP 333. WSGI é um padrão Python e tem como objetivos ser simples e de fácil implementação.

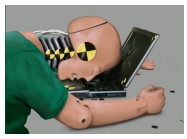
Historicamente, as aplicações Python para web eram um problema para desenvolvedores pois a escolha do framework era um limitante para a escolha do servido, e vice-versa. Os aplicativos Python eram arquitetados para serem utilizados com CGI, FCGI, mod_python ou alguma API específica do servidor web.

Com a WSGI criou-se uma padronização da especificação de toda a comunicação entre webserver e aplicação escrita em Python.

CursoDePython – [3,1]
FRAMEWORK

”Framework é um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação.”
– Fayad Schmidit





- Não requer instalação
- Web server integrado
- Suporte a multiplos bancos de dados
- Camada de abstração de acesso a dados
- Padrão MVC
- Padrão WSGI
- MultiPlataforma
- Geradores de formulários, uploads, validadores
- Segurança e controle de acesso
- Web services
- Interface administrativa
- Editor de códigos via web
- Sistema de testes unitários
- Sistema de ticket/log para erros
- Sistema de plugins
- Compatibilidade ETERNA!

CursoDePython.com.br 5

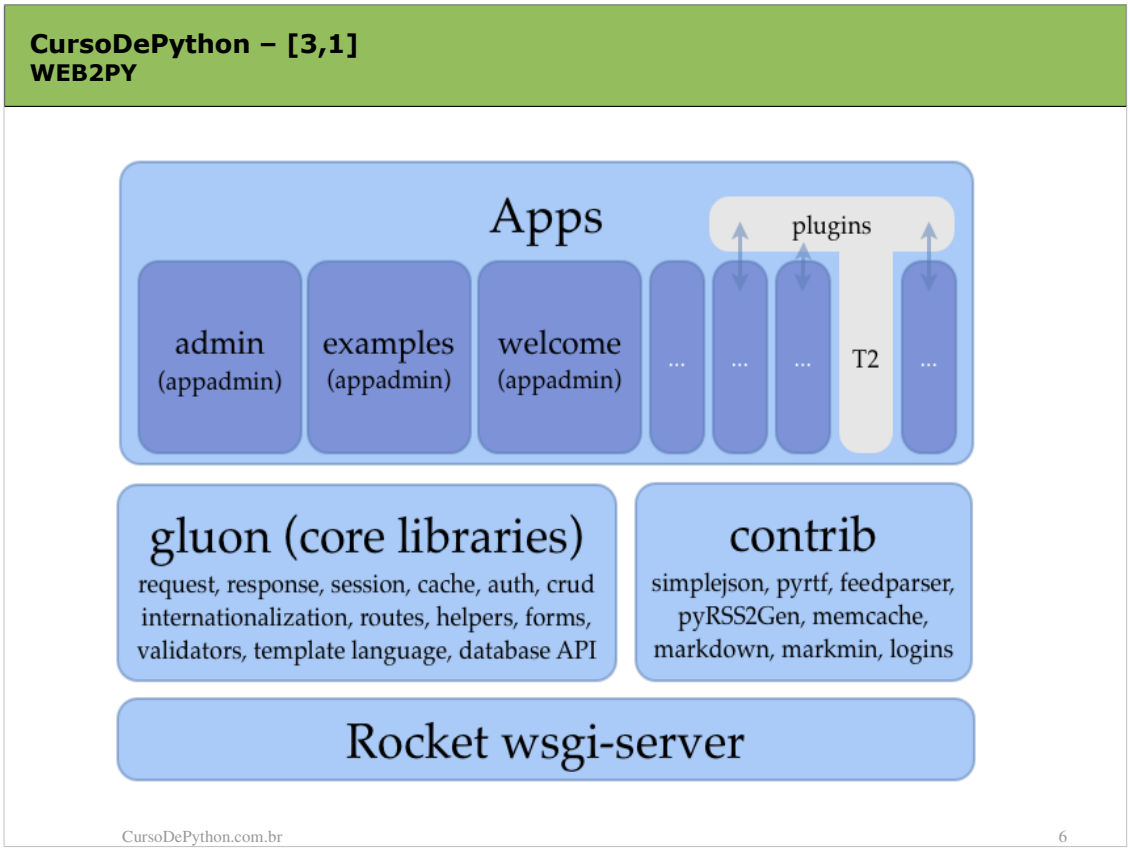
O que é web2py?

web2py é um framework para desenvolvimento Web escrito em Python, software livre e gratuito, que tem como um de seus principais objetivos proporcionar agilidade no desenvolvimento de aplicações web seguras, baseadas em bancos de dados.

O framework segue o modelo MVC (Model-View-Controller), que permite melhor organização do código. Ele também é autocontido, ou seja, tudo o que você precisa para desenvolver alguma aplicação está nele, basta baixar e descompactar para começar - nada de configurações!

Com o foco em permitir que o desenvolvedor pense apenas na aplicação que está desenvolvendo, o web2py possui integração com mais de 10 sistemas de banco de dados e vários subsistemas, como: 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.), dentre outros.

O web2py leva em consideração todas as questões referentes à segurança da aplicação web, e isso significa que o framework se preocupa em tratar as vulnerabilidades aplicando práticas bem estabelecidas.



gluon

gluon foi um dos nomes do framework antes de se tornar web2py, para manter a compatibilidade com as versões anteriores o nome do pacote foi, e sempre será mantido.

Gluon é o nucleo do framework, dentro deste pacote estão todas as bibliotecas e módulos que fazem parte do framework.

Além de gluon também temos o pacote contrib, onde ficam as bibliotecas de terceiros, contribuidores e vindas de outros frameworks que foram aproveitadas no web2py.

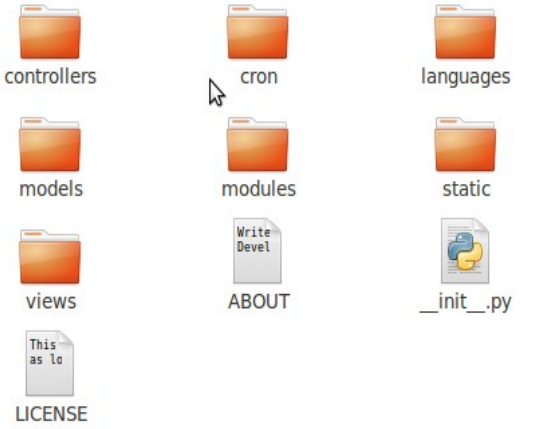
Acima desses dois pacotes principais rodamos nossas aplicações e plugins.

E toda esta estrutura é servida por um webserver, que no caso do web2py pode ser qualquer um com suporte a WSGI, mas para facilitar, o framework já vem com um framework interno.

CursoDePython – [3,1]
APLICAÇÃO WEB2PY - MVC

- **Models**
  - definição de modelos de dados.
- **Controllers**
  - Programação do controle de fluxo.
- **Modules**
  - Classes, funções (a programação dos objetos da aplicação)
- **Views**
  - Interface da aplicação, apresentação, estilos, layouts.
- **Static**
  - Arquivos estáticos (não Python) css, js, imagens.
- **Languages**
  - Traduções da aplicação
- **Cron**
  - Agendador de tarefas

- **Databases**
  - Arquivos de controle do banco de dados
- **Uploads**
  - Arquivos enviados pelos usuários da aplicação
- **Sessions**
  - Arquivos de controle de sessões físicas
- **Cache**
  - Cache de disco
- **Errors**
  - tickets/logs de erro



CursoDePython.com.br 7

[**models**]
Modelos – É o local onde definimos e gerenciamos os modelos de dados da aplicação, efetuamos conexões com bancos de dados e definimos a modelagem das tabelas, constantes, variáveis e configurações de acesso global.

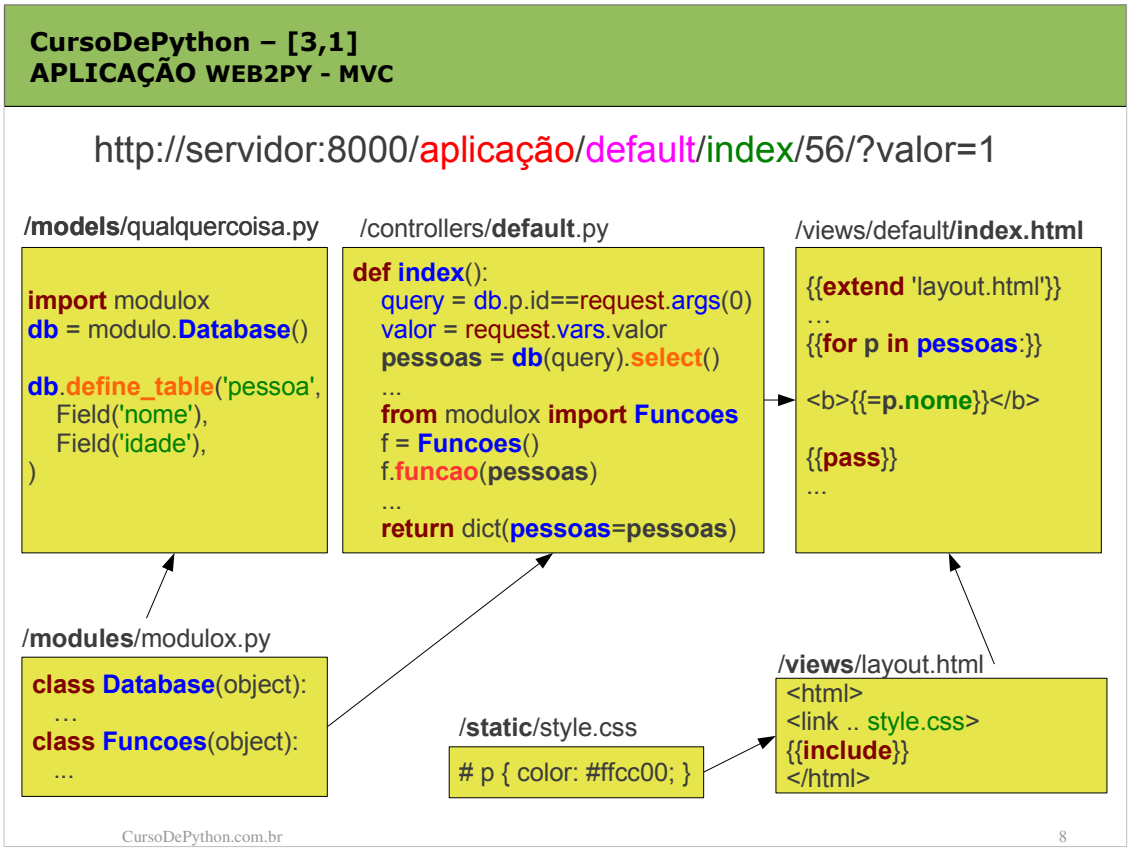
[**controllers**]
Controladores – São as ações da aplicação. Aqui definimos as regras de negócio e as validações de tempo de execução; o controller é quem recebe a entrada de dados, invoca os objetos do modelo de dados, efetua as validações e envia como resposta uma visão.

[**views**]
Visões – São as visões que servirão para apresentar os dados do model, invocados e tratados pelo controller. São criadas a partir de templates que podem responder conteúdo no formato HTML, RSS, XML e JSON, entre outros.

[**languages**]
Linguagens – Local onde definimos arquivos de linguagem que permitem a internacionalização das aplicações.

[**static**]
Estáticos – Neste diretório, inserimos arquivos que não necessitam de processamento, como estruturas estáticas de layout, imagens, arquivos de estilo CSS e JavaScript.

[**modules**]
Módulos - Aqui colocaremos bibliotecas e módulos Python que não fazem parte do web2py, mas que podem ser importados e utilizados nas aplicações.



Mapeamento de URLS

O mapeamento de URLs (Dispatching) no web2py é orientado a ações dos controladores, cada ação de um controlador é acessível através de uma URL no seguinte formato:

http://servidor:porta/aplicação/controlador/ação/argumentos?variáveis

http://servidor:porta
Endereço IP e porta do servidor, em desenvolvimento 127.0.0.1:8080

/aplicação
Pasta da aplicação web2py, ao ser chamada executa o código dos modelos

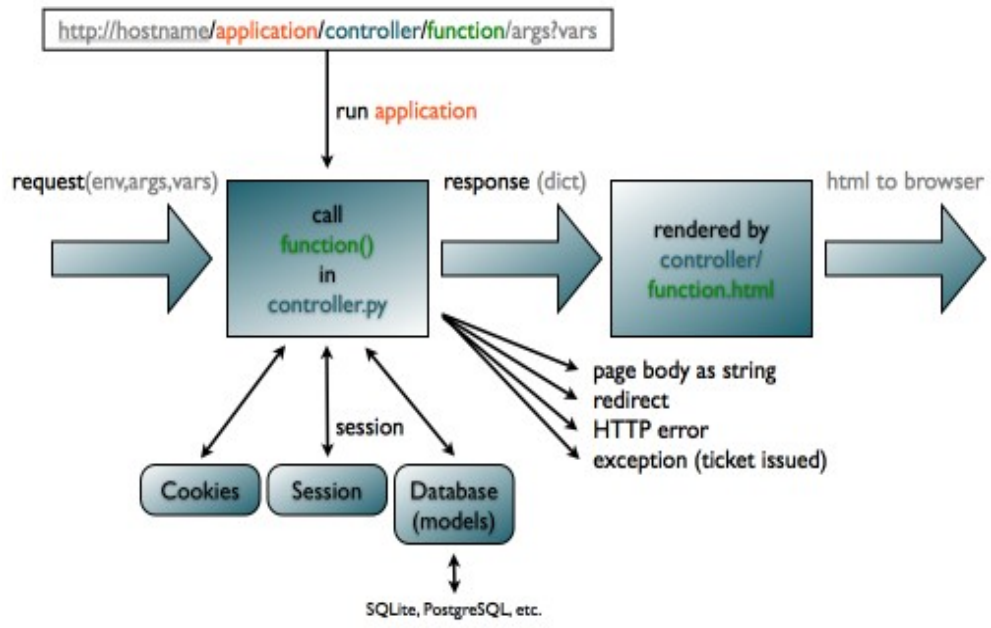
/controlador
Módulo controlador

/ação
Ação de um controlador, ao ser chamada executa o código da função correspondente, recebe os parâmetros e variáveis

/argumentos
Argumentos passados para o controlador são tratados como POST

?variáveis
Variáveis são tratadas como GET (QueryString)

CursoDePython – [3,1]
ROTEAMENTO



Controladores são a porta de entrada de uma aplicação web2py.

Quando um usuário efetua uma requisição(request) no navegador o web2py irá mapear esta requisição de acordo com o esquema de mapeamento de URLs definido no framework, este esquema de mapeamento já vem por padrão de acordo com a imagem acima.

Vamos pegar por exemplo a seguinte URL:

http://127.0.0.1:8080/a/c/f.html

O web2py irá procurar um arquivo de configuração de mapeamento de URLs(routes.py), se este arquivo for encontrado, então será aplicado o modelo definido nele, caso este arquivo não exista, então o web2py irá utilizar o mapeamento padrão.

Lendo do final para o ínicio:
A função **f()** definida no controlador **c.py** que faz parte da aplicação **a**, e que retorna conteúdo no formato **html**

CursoDePython – [3,1]
HTML HELPERS

Considere o seguinte código:

```
DIV('isto',  'é',  'um',  'teste',  _id='123',  _class='minhaclasse')
```

Que será renderizado como:

```
<div id="123" c l a s s="minhaclasse">istoéumteste< /div>
```

DIV, por exemplo, é uma classe helper, um tipo de classe que pode ser usada para criar HTML programaticamente.
Em uma classe helper, os argumentos são interpretados como objetos contidos dentro da tag HTML. Os argumentos nomeados que iniciam com um underline serão interpretados como atributos. Alguns helpers recebem argumentos nomeados que não iniciam com underline; estes argumentos são específicos para cada tipo de TAG

Um do mais importantes é o helper URL, que automatiza o processo de criação de URLs de acordo com as rotas.

```
A(”LINK”, _href=URL('default','index',args=['Bruno',2],vars={'valor':20}))
```

será renderizado como:

```
<a href='http://server:porta/aplicacao/default/index/Bruno/2/?valor=20'>LINK</a>
```


TAGS

A, B, BEAUTIFY, BODY, BR, CENTER, CODE, DIV, EM, EMBED, FIELDSET, FORM, H1, H2, H3, H4, H5, H6,
HEAD, HR, HTML, I, IFRAME, IMG, INPUT, LABEL, LEGEND, LI, LINK, OL, UL, MARKMIN, MENU, META,
OBJECT, ON, OPTION, P, PRE, SCRIPT, OPTGROUP, SELECT, SPAN, STYLE, TABLE, TAG, TD, TEXTAREA,
TH, THEAD, TBODY, TFOOT, TITLE, TR, TT, URL, XHTML, XML, xmlescape, embed64

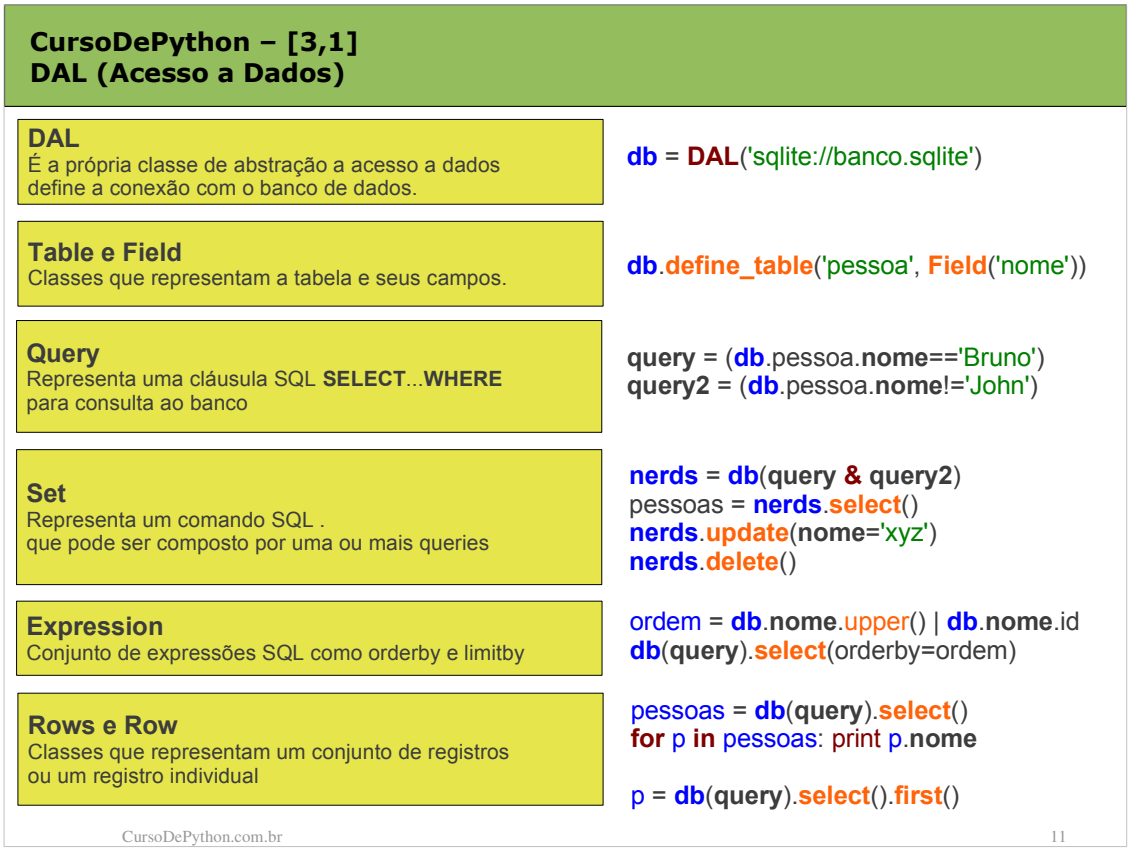
As tags podem ser agrupadas para gerar códigos HTML mais complexos:

DIV(B(I("hello ", "<world>"))) , _class="myclass")

é renderizado como:

<div c l a s s="myclass"> <b> <i>hello <world> < /i> < /b> < /div>

http://www.web2py.com/book/default/chapter/07



A camada de acesso à dados - DAL

O web2py possui uma camada de abstração de banco de dados(DAL), é uma API que mapeia objetos Python em objetos de banco de dados como queries, tabelas, e registros.

A DAL gera códigos SQL dinâmicamente em tempo real utilizando sempre o dialeto SQL referente ao banco de dados em uso, dessa forma você não precisa escrever código SQL ou aprender comandos SQL de um banco de dados específico, e sua aplicação será portável para diferentes bancos de dados, Atualmente os bancos de dados suportados são:

SQLite (que já vem com Python e com web2py),
PostrgreSQL,
MySQL,
Oracle,
MSSQL,
FireBird,
DB2,
Informix,
Ingres.
CUBRID,
MongoDB e CouchDB

O web2py também é capaz de se conectar ao Google BigTable no Google App Engine (GAE).

CursoDePython – [3,1]
Definição de modelo de dados e migrations

definir ou alterar a lista de campos ou tipos de campo em um modelo, provoca uma migração automática, ou seja, web2py gera SQL para alterar a tabela de acordo.

Se a tabela não existir ele é criada. Ações de migração são registradas no arquivo Sql.log acessível através da interface administrativa. A migração pode ser desligada por tabele em uma base de dados, passando o parâmetro migration = False no método define_table.

```
db.define_table('empresa',
    Field('CNPJ','string', length=18, unique=True),
    Field('atividade','text'),
    Field('logotipo','upload'),
    migrate = True
)
```

Relacionamento **UM** para **MUITOS**

Para todas as tabelas a DAL cria automaticamente o campo 'id' como um auto-incremento.

Podemos definir o id da seguinte maneira: Field('meu_id','id') para isso migrate deve ser False

```
db.define_table('pessoa',
    Field('nome','string', length=50, unique=True),
    Field('idade','integer'),
    Field('foto','upload'),
    Field('empresa', db.empresa),
    migrate = True
)
```

CursoDePython.com.br 12

Tipos de dados

A DAL suporta os seguintes tipos de dados:

Field(name, 'string')
Field(name, 'text')
Field(name, 'password')
Field(name, 'blob')
Field(name, 'upload')
Field(name, 'boolean')
Field(name, 'integer')
Field(name, 'double')
Field(name, 'time')
Field(name, 'date')
Field(name, 'datetime')
Field(name, db.referenced_table) # reference field

Atributos

length (apenas para o tipo string, padrão 32)
default (padrão None)
required (padrão False)
notnull (padrão False)
unique (padrão False)
requires (um validador ou uma lista de validadores para os formulários)
comment (Comentários para formulários)
widget (Para formulários)
represent (Para formulários)
readable (Determina se o campo é visível no formulário, padrão True)
writable (Determina se o campo é alteravel no formulário, padrão True)
update (valor padrão caso o registro seja atualizado)
uploadfield (Para criação de um campo BLOB para upload, padrão None)
authorize (Para autenticar em caso de download protegido)
autodelete (padrão False, caso True, apaga a imagem assim que sua referência no banco for apagada)
label (Rótulos para os formulários)

CursoDePython – [3,1]
FORMULÁRIOS

FORM é um helper que retorna um formulário html puro e permite que seja efetuada a validacão.

```
form = FORM('Seu nome:', INPUT(_name='name'), INPUT(_type='submit'))
```

SQLFORM é um helper que cria um formulário baseado em uma tabela de nosso modelo de dados, faz a validação e insere os dados na tabela automaticamente.

```
form = SQLFORM(db.pessoa, formstyle='divs')
```

CUSTOM FORMS é uma maneira para customizar nas views os formulários criados automaticamente.

```
{{=form.custom.begin}}
    <b>Nome</b><div>{{=form.custom.widget.name}}</div>
{{=form.custom.end}}
```

CRUD é um helper que automatiza a criação de formulários para consulta, inserção, edição, busca e remoção de registros.

```
form = crud.create(db.pessoa)
updateform = crud.update(db.pessoa, id_do_registro)
searchform = crud.search(db.pessoa)
```

CursoDePython.com.br 13

FORMS

Os formulários possuem um método que avaliam os campos e fazem a validacão considerando os validators definidos, além de executar operações de bancos de dados. é o método accepts()

```
def display_form():
    form=FORM('Your name:',
              INPUT(_name='name', requires=IS_NOT_EMPTY()),
              INPUT(_type='submit'))

    if form.accepts(request.vars, session):
        response.flash = 'form accepted'
    elif form.errors:
        response.flash = 'form has errors'
    else:
        response.flash = 'please fill the form'
    return dict(form=form)
```

A lista completa de métodos e opções pode ser encontrada em:
http://www.web2py.com/book/default/chapter/07

CursoDePython – [3,1]
VALIDATORS

Validadores são classes usadas para validar campos de entrada em formulários (incluindo os gerados através do banco de dados).

Exemplo de uso de um validador na criação de um formulário.

```
INPUT(_name='marca',  requires=IS_NOT_EMPTY(error_message='Você deve informar a marca!') )
```

Aqui um exemplo de uso de validador na base de dados:

```
banco.define_table('carros',  Field('marca') )
banco.carros.marca.requires = IS_NOT_EMPTY( )
```

Os validadores sempre são atribuídos através do atributo requires da classe Field. Um campo pode ter um ou mais validadores. Múltiplos validadores são formados com uma lista

```
bancos.carros.marca.requires =   [IS_NOT_EMPTY( ) , IS_NOT_IN_DB(banco, 'carros.marca') ]
```

Os validadores são invocados através da função accepts em um **FORM** ou qualquer outro **HTML** helper que contenha um **FORM**. São invocados na exata ordem em que são listados.

Validadores de formulário

IS_ALPHANUMERIC
Este validador checa se um campo contém um caracter dentro do range a-z, A-Z ou 0-9

IS_DATE
Este validador checa se o campo contém uma data válida, de acordo com o formato especificado

```
requires = IS_DATE(format='%Y-%m-%d',
                   error_message='O formato deve ser AAAA-MM-DD')
```

IS_IN_SET
Checa se o valor está contido em um conjunto (set):

```
requires = IS_IN_SET(['a', 'b', 'c'],
                     zero='escolha um valor',
                     error_message='o valor deve ser a, b ou c'
                     )
```

Lista completa em: http://www.web2py.com/book/default/chapter/07

CursoDePython – [3,1]
Acesso e Autenticação

```
from gluon.tools import Auth
auth = Auth(db)
auth.define_tables(username=False)
```

```
@auth.requires_login()
def minhapagina():
    dados = 'segredo'
    return dict(dados=dados)
```

```
@auth.requires_membership('grupo')
def minhapagina():
    dados = 'segredo'
    return dict(dados=dados)
```

```
@auth.requires(auth.user_id==1 or request.client=='127.0.0.1')
def minhapagina():
    dados = 'segredo'
    return dict(dados=dados)
```

CursoDePython.com.br 15

O web2py inclui um poderoso e customizável sistema de controle de acesso baseado em papéis (RBAC). Este sistema é implementado na classe Auth que necessita e implementa as seguintes tabelas:

auth_user
armazena o nome do usuário, o endereço de e-mail, a senha e a situação do registro do usuário (pendente, aceito ou bloqueado)

auth_group
armazena os grupos ou os papéis para os usuários. Por padrão cada novo usuário criado possui seu próprio grupo, porém, cada usuário pode fazer parte de múltiplos grupos, e cada grupo pode conter diversos usuários.

auth_membership
efetua o relacionamento entre usuários e grupos em uma estrutura m u i t o s-p a r a-m u i t o s.

auth_permission
efetua o relacionamento entre grupos e permissões.
Uma permissão é identifica por um nome e opcionalmente por uma tabela e um registro. Por exemplo, membros de um certo grupo podem executar update em um registro específico de uma tabela.

auth_event
armazena o log do sistema de autenticação.

CursoDePython – [3,1]
MAIL

O web2py possui uma classe especializada no envio de emails, esta classe funciona com qualquer servidor smtp e já vem pré configurada para trabalhar com o servidor do gmail.

```
from gluon.tools import Mail
mail = Mail()

mail.settings.server = 'smtp.gmail.com:587'
mail.settings.sender = 'seuemail@gmail.com'
mail.settings.login = 'seuemail@gmail.com:suasenha'
```

```
mail.send(
    to = 'destino@email.com',
    cc=['alguem@dominio.com'],
    bcc=['pessoa@oculta.com'],
    reply_to='alguem@dominio.com ',
    subject='Assunto do email',
    message=["texto puro", "<b>texto em html</b>"],
    attachments = [Mail.Attachment('/pasta/arquivo.zip', content_id='zip')]
    )
```

CursoDePython – [3,1]
VIEWS

Views são as estruturas de hipertexto que criam a interface de apresentação da aplicação.

Podem ser feitas nos formatos: html, xml, rss, json, pdf, xls, xlsx, atom ou customizadas.

Tudo o que é escrito entre os caracteres {{ e }} é executado como código Python, qualquer coisa fora destas tags será apenas apresentado sem executar.

Views podem herdar um layout atraves da instrução {{extend}} podem embutor um arquivo html externo com {{include}} e possuem blocos {{block …}} {{end}}

```
def index():
    ...
    return dict(var=value)
```

```
{{extend 'layout.html'}}
…
{{for i in range(10):}}

<b>{{=var}}</b>

{{pass}}
…

{{=DIV(var)}}
```

CursoDePython.com.br 17

Assim como nos models e controllers o web2py utiliza a própria linguagem Python para a marcação de template das views. Nas views é possível escrever qualquer tipo de código de marcação, como HTML, XML, CSS, JavaScript e embutir código Python no meio, e isso faz com que você tenha uma maior flexibilidade para gerar o conteúdo da maneira que quiser, podendo por exemplo programar o formato de saída de uma view em .csv, .rss ou até mesmo .xls do Microsoft Excel.

O web2py executa as views em uma cópia separada do ambiente de execução da aplicação, e para isso, transforma todo o código de marcação da view em um módulo com puro código Python, por este motivo a execução da views é rápida.

O código da view não precisa seguir regras de endentacão do Python
A marcação para escape é feita dentro de {{ e }}
Blocos de código começam nas linhas {{}} terminadas em :
Blocos de código teminam onde encontram a instrução {{pass}}
Nos casos em que a construção do bloco for clara, não será preciso usar o {{pass}}

Tudo o que está entre {{...}} será executado como código Python. Além disso, o web2py especifica algumas palavras-chave para ajudar no processo de criação de views, são elas:

{{=expressão_Python}}: executa expressão_Python e imprime o resultado no local;
{{extend 'arquivo.html'}}: utiliza a view arquivo.html como base para essa view;
{{include}}: indica onde outras views que utilizam essa como base serão adicionadas.
{{pass}}: marca o término de um bloco de código