Pyxer

Yet another Python framework

(C)opyright by Dirk Holtwick, Germany
dirk.holtwick@gmail.com
http://www.pyxer.net

Table of Contents

Introduction

The **Pyxer** Server is a very simple Python Web Framework that aims to makes starting a new project as easy as it can be. It still works respecting the MVC concept but the files can be mixed in one directory. For a high end solution you should consider using Pylons, TurboGears, Django and similar.

This work is inspired by http://pythonpaste.org/webob/do-it-yourself.html

Technical background

The Google App Engine (GAE) in version 1.1 offers a very restricted Python environment and the developer as to ship arround a lot of problems. Pyxer helps on this point by also providing solutions that work with the great WSGI Framework Paster by Ian Bicking. So you get the best from both sides: GAE and Paster. To achieve this some other great third party tools are used like WebOb and VirtualEnv (also by Ian Bicking) and for templating the html5lib is used.

Installation

Install Pyxer using easy_install:

```
$ easy_install pyxer
```

All required packages (webob, html5lib, beaker) should be installed automatically if needed.

If you want to use Google AppEngine install it separately too.

Quick start

Create a new project

At first set up a new Pyxer project using the Pyxer command line tool like this:

```
$  pyxer init myexample
```

In the newly created directory "myexample" you will find a directory structure like this one (under windows "bin" will be called "Scripts"):

```
bin/
public/
lib/
```

Place your files in the "public" directory.

Start the server

To start the server you may choose between the Paster-Engine:

```
$ xpaster serve
```

or the GAE-Engine:

```
$ xgae serve
```

Or use Pyxer command line tool again to use the default engine (which is Paster):

```
$ pyxer serve
```

"Hello World"

For a simple "Hello World" just put an "index.html" file into the "public" directory with the following content:

```
Hello World
```

This works just like a static server. To use a controller put a file "__init__.py" into that directory with the following content:

```
@controller
def index():
    return "Hello World"
```

About controllers

Controller, templates and static files are placed in the same directory (usually "public"). First **Pyxer** looks for a matching controller. A controller is defined in the "__init__.py" file and decorated by using **`@controller`** which is defined in **`pyxer.base`**.

```
from pyxer.base import *

@controller
def index():
    return "Hello World"
```

This example can be called like `/` or `/index`. To use a Genshi template with this file you may use the `render()` function or just return `None` (that is the same as not returning anything) and the matching template will be used, in this case `index.html`. The available object in the template are the same as used by Pylons: `c` = context, `g` = globals and `h` = helpers.

```
from pyxer.base import *

@controller
def index():
    c.title = "Hello World"
```

XXX

1. #Looks for a controller (foo/bar/__init__py:bar)
   1. If the controller returns a dictionary this will be applied to template (step 2)
2. Looks for the template (foo/bar.html)

Templating

Pyxer offers yet another templating language that is very close to Genshi and Kit. As the former did not work with Google AppEngine at the moment of birth of **Pyxer** the new templating tools had been implemented. It is based on html5lib.

Variables and expressions

The default templating works similar to most known other templating languages. Variables and expressions are realized like `$<varname>` (where `<varname>` may contain dots!) and `${<expression>}`:

```
Hello ${name.capitalize()}, you won $price.
$item.amount times $item.name.
```

Commands

Commands are the same as used by Python. The only difference is, that you have to tell that a block ends by calling the command `end` (not needed for `else` and `elif`):

```
<% def address(name): %>
  ABCDEF Company
  <% if name: %>
    Your contact: $name
  <% else: %>
    Unknown contact!
  <% end %>
<% end %>
<% for name in ['Tom', 'Fred']: %>
  ${ address(name) }
<% end %>
```

Includes and extends

You can insert content of other templates by using “include”:

```
<% include(filename) %>
```

More sophisticated is the extension of a file.

```
<% extends('b.html') %>
<% def content(): %>
  Here goes the content
<% end %>
```

File “a.html”

```
<% content() %>
```

File “b.html”

HTML Attriutes

These are also known form many templating languages like Genshi. They are used like this:

```
<div py:if=”name.startswith('tom ')”>Welcome $name</div>
```

xxx

XXX Engines

Pyxer uses support different so called "engines" to publish a project. Most of them need own configurations and a well prepare environment to work fine. These are very specific to each of these engines and **Pyxer** tries to make the setup as easy as possible

Paster

```
$ xpaster serve --reload
```

Google Appengine

```
$ xgae serve
```