TurboGears

tutorialspoint
SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

TurboGears is a Python web application framework, which consists of many modules. It is designed around the MVC architecture that are similar to Ruby on Rails or Struts. TurboGears are designed to make rapid web application development in Python easier and more supportable.

TurboGears is a web application framework written in Python. TurboGears follows the Model-View-Controller paradigm as do most modern web frameworks like Rails, Django, Struts, etc. This is an elementary tutorial that covers all the basics of TurboGears.

Audience

This tutorial has been designed for all those readers who want to learn the basics of TurboGears. It is especially going to be useful for all those Web developers who are required to simplify complex problems and create single database backed webpages.

Prerequisites

We assume the readers of this tutorial have a basic knowledge of web application frameworks. It will be an added advantage if the readers have hands-on experience of Python programming language. In addition, it is going to also help if the readers have an elementary knowledge of Ruby-on-Rails and Struts.

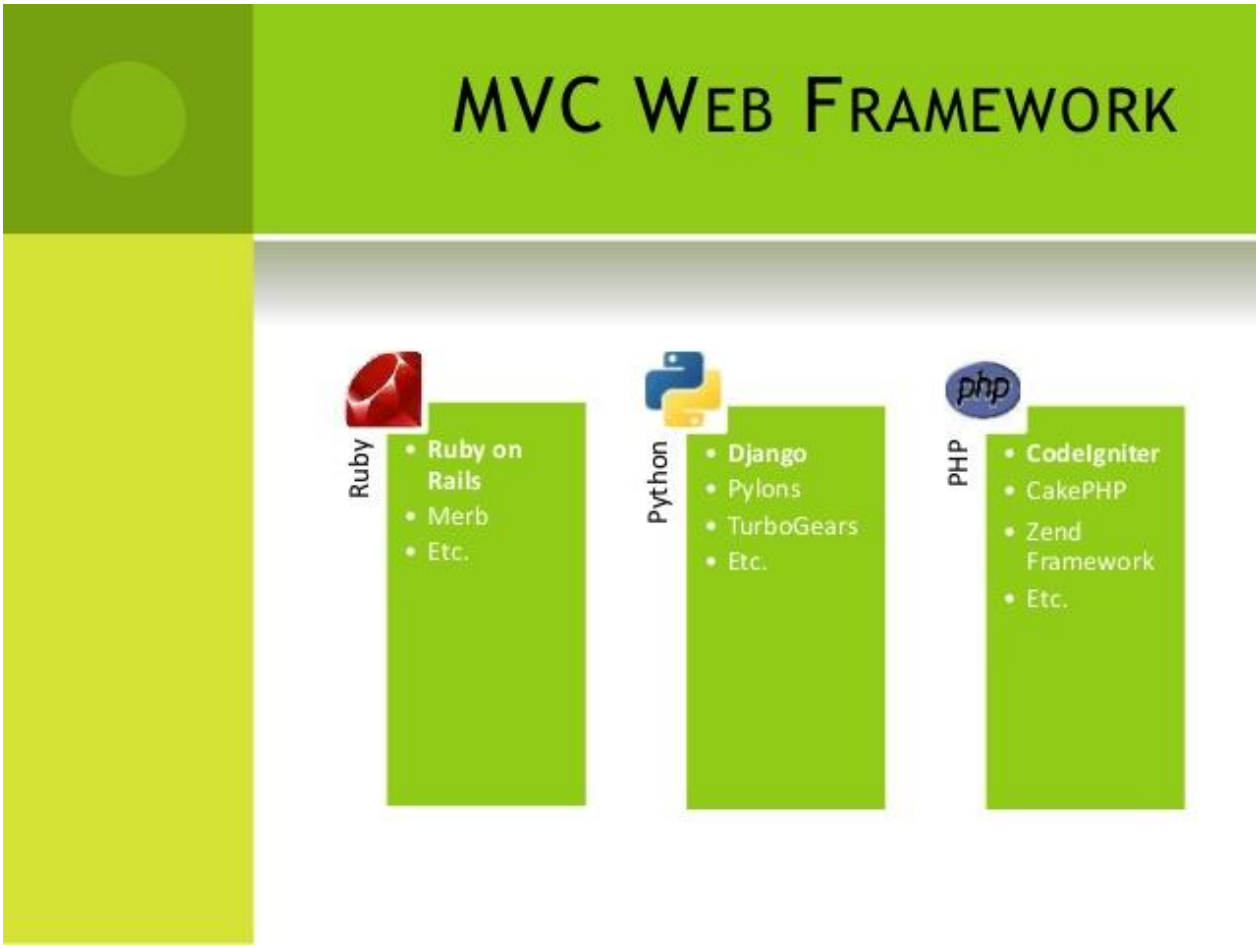
Disclaimer & Copyright

Table of Contents

1. TurboGears – Overview

What is Web Framework?

Web Application Framework or simply Web Framework represents a collection of libraries and modules, which enables a web application developer to write applications, without having to bother about low level details such as protocols, thread management, etc.



What is TurboGears?

TurboGears is a web application framework written in Python. Originally created by Kevin Dangoor in 2005, its latest version TurboGears (ver 2.3.7) is managed by a group of developers led by Mark Ramm and Florent Aide.

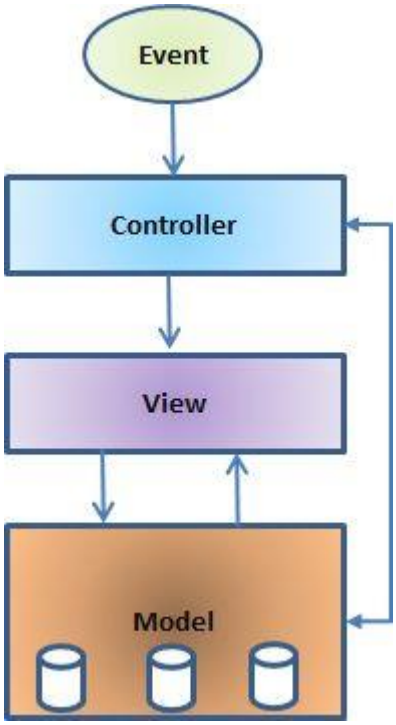
TurboGears follows the Model-View-Controller paradigm as do most modern web frameworks like Rails, Django, Struts, etc.

Model View Controller

MVC is a software design pattern for developing web applications. A Model View Controller pattern is made up of three parts:

- **Model** - The lowest level of the pattern is responsible for maintaining data.
- **View** - This is responsible for displaying all or a portion of data to the user.
- **Controller** - Software Code that controls the interactions between the Model and View.

MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here, the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows:



The Model

The Model is responsible for managing the data of the application. It responds to the request from the view and it also responds to instructions from the controller to update itself.

The View

A presentation of data in a particular format, triggered by a controller's decision to present the data. They are script based templating systems very easy to integrate with AJAX technology.

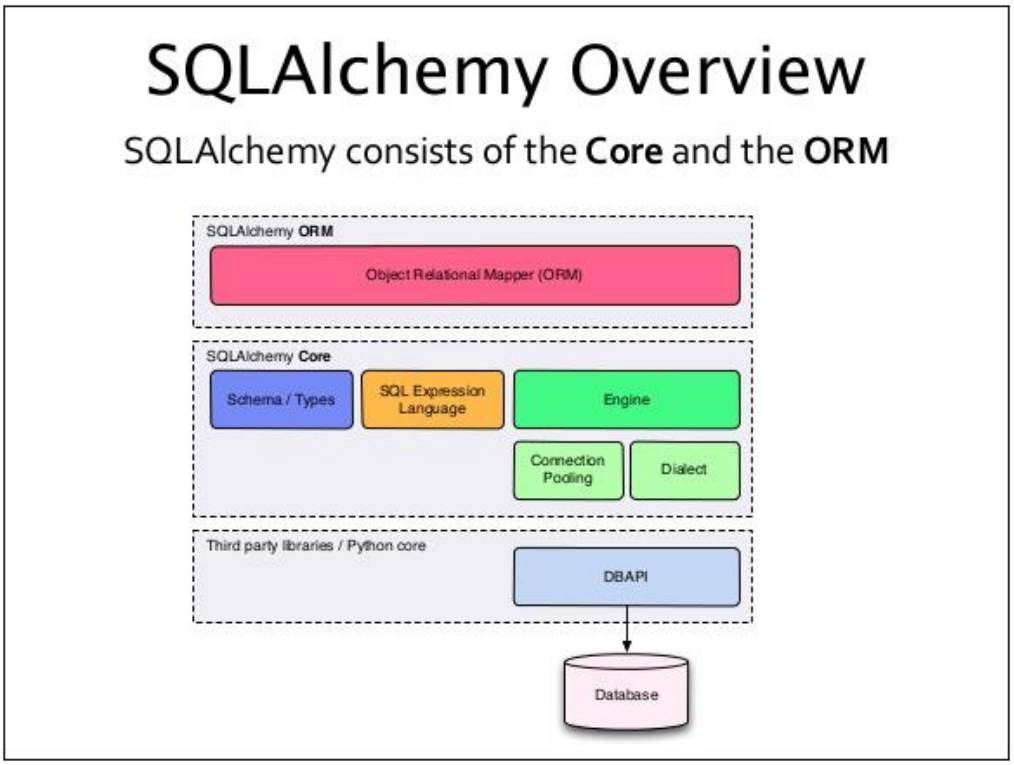
The Controller

The controller is responsible for responding to the user input and perform interactions on the data model objects. The Controller receives the input, it validates the input and then performs the business operation that modifies the state of the data model.

TurboGears is built on top of a number of libraries and tools. These tools have changed between different versions of TurboGears. The components of current version (ver 2.3.7) are listed below.

SQLAlchemy

It is an open source SQL kit that provides Object relation mapping (ORM) for Python code.



Genshi

This templating engine is used to construct the front-end of TG applications. A web templating system combines a template with a certain data source to render dynamic web pages.

ToscaWidgets

It is a widget library for generating HTML forms with server side controls. Tosca also acts as a middleware to connect with JavaScript widgets and toolkits.

Gearbox

It provides a set of commands to manage projects and server TurboGears applications. TurboGears applications can be deployed on any WSGI compliant web server.

The Web Server Gateway Interface (WSGI) has been adopted as a standard for Python web application development. WSGI is a specification for universal interface between web server and web applications. The wsgiref package is a reference implementation of WSGI. It is used to add WSGI support to web TurboGears web framework. The simple_server module in this package implements a simple HTTP server that serves WSGI applications. We shall be using it to test applications developed during this tutorial.

2. TurboGears – Environment

Prerequisite

Python 2.6 or higher. Earlier versions of TurboGears were not compliant with Python 3.X. Latest version claims to work well on Python 3.X. However, official documentation of TurboGears is still based on Python 2.7 environment.

The following command **installs virtualenv –**

```
pip install virtualenv
```

This command needs **administrator** privileges. Add **sudo before pip** on Linux/Mac OS. If you are on Windows, log in as Administrator. On Ubuntu virtualenv may be installed using its package manager.

```
Sudo apt-get install virtualenv
```

Once installed, the new virtual environment is created in a folder.

```
mkdir newproj
cd newproj
virtualenv venv
```

To activate corresponding environment, on **Linux/OS X**

```
venv/bin/activate
```

On **Windows**

```
venv\scripts\activate
```

We are now ready to **install TurboGears** in this environment. A minimal installation of TurboGears is done by following command:

```
pip install TurboGears2
```

The above command can be run directly without virtual environment for system wide installation.

To install TurboGears along with development tools, use following command:

```
pip install tg.devtools
```

3. TurboGears – First Program

TurboGears has a minimal mode that makes it possible to create single file applications quickly. Simple examples and services can be built quickly with minimal set of dependencies.

Application class in a TG application is inherited from **TGController** class. Methods in this class are available for access by **@expose** decorator from **tg** module. In our first application, **index()** method is mapped as root of our application. The TGController class also needs to be imported from **tg** module.

```
from tg import expose, TGController

class MyController(TGController):

    @expose()

    def index(self):

        return 'Hello World turbogears'
```

Next, set the application's configuration and declare application object. **AppConfig** class constructor here takes two parameters – minimal attribute set to true and the controller class.

```
config = AppConfig(minimal=True, root_controller=RootController())

application = config.make_wsgi_app()
```

The **make_wsgi_app()** function here constructs application object.

In order to serve this application, we now need to start the HTTP server. As mentioned earlier, we shall use **simple_server** module in **wsgiref** package to set up and start it. This module has **make_server()** method which requires port number and application object as arguments.

```
from wsgiref.simple_server import make_server

server = make_server('', 8080, application)

server.serve_forever()
```

It means that our application is going to be served at port number 8080 of localhost.

The following is the complete code of our first TurboGears application **–**

app.py

```
from wsgiref.simple_server import make_server

from tg import expose, TGController, AppConfig

class MyController(TGController):
```

```
    @expose()
    def index(self):
        return 'Hello World TurboGears'

config = AppConfig(minimal=True, root_controller=MyController())
application = config.make_wsgi_app()

print "Serving on port 8080..."
server = make_server('', 8080, application)
server.serve_forever()
```

Run the above script from Python shell.

```
Python app.py
```

Enter http://localhost:8080 in browser's address bar to view 'Hello World TurboGears' message.

The **tg.devtools** of TurboGears contains Gearbox. It is a set of commands, which are useful for management of more complex TG projects. Full stack projects can be quickly created by the following Gearbox command:

```
gearbox quickstart HelloWorld
```

This will create a project called **HelloWorld**.

End of ebook preview

If you liked what you saw…

Buy it from our store @ **https://store.tutorialspoint.com**