

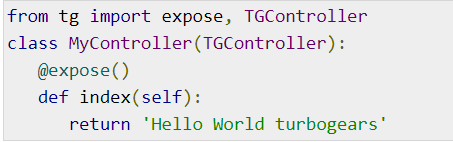
TurboGears

Web-FrameWork

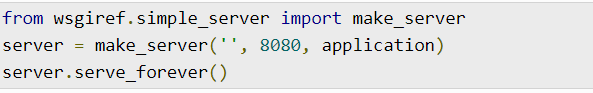
Prepared by Shubham Panchal

(20CE070)

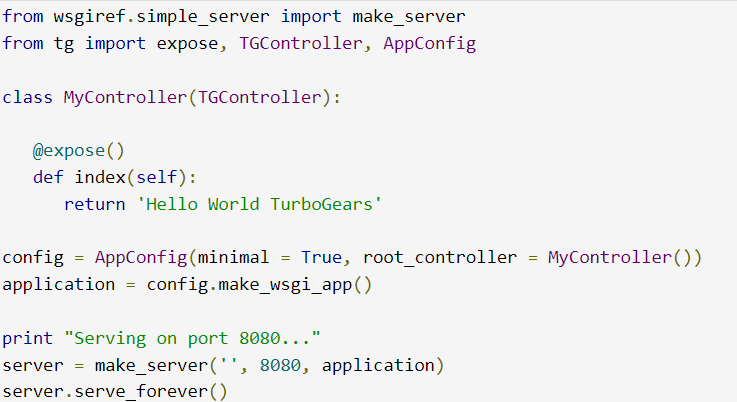
* **TURBOGEARS-Web Framework**
  + TurboGears is a Python web application framework, which consists of many modules. TurboGears are designed to make rapid web application development in Python easier and more supportable. TurboGears is a web application framework written in Python. TurboGears follows the Model-View-Controller pedagogy as do most modern web frameworks like Rails, Django, Struts, etc.
  + TurboGears 2 is built on top of the experience of several next generation web frameworks including TurboGears 1, Django, and Rails. All of these frameworks had limitations that frustrated us, and TG2 was built as an answer to that frustration.
* **Feature**
  + Starts as a microframework and scales up to a full stack solution
  + Code that is as natural as writing a function
  + A powerful and flexible **Object Relational Mapper (ORM)** with real multi-database support
  + Support for Horizontal data partitioning (aka, sharding)
  + A new widget system to make building AJAX heavy apps easier
  + Support for multiple data-exchange formats
  + Built in extensibility Pluggable Applications and standard Web-server gateway interface (**WSGI**)components
  + Designer friendly template system great for programmers
* **Installing Turbogears**
  + (tgenv)$ pip install TurboGears2
* **HELLO WORLD**
  + 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.
  + To create a TurboGears application we need a configurator (Application Configurator) and a RootController. The configurator will set up and create the TurboGears application itself, while the controller will be in charge of dispatching the received requests and taking actions.
  + For convenience, we will use a **MinimalApplicationConfigurator** which provides a minimal set of components that your application will surely need to work like routing.
  + For our first application we are going to define a controller with an index method that just tells ***Hello World turbogears***. Just create a new tgapp.py file and declare your RootController.

****

* Next, set the application’s configuration and declare the application object. The **AppConfig** class constructor here takes two parameters – minimal attribute set to true and the controller class.
* The **make\_wsgi\_app()** function here constructs an application object.
* In order to serve this application, we now need to start the HTTP server. As mentioned earlier, we shall use the **simple\_server** module in **wsgiref** package to set up and start it. This module has a **make\_server()** method which requires port number and application object as arguments.



* 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 −



**END**