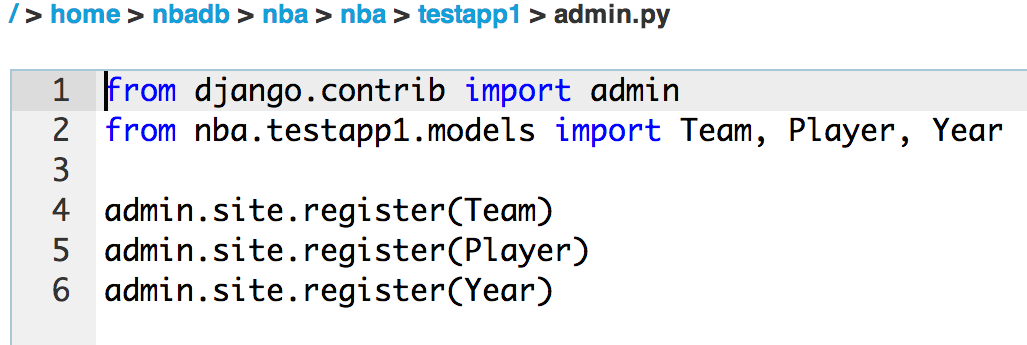
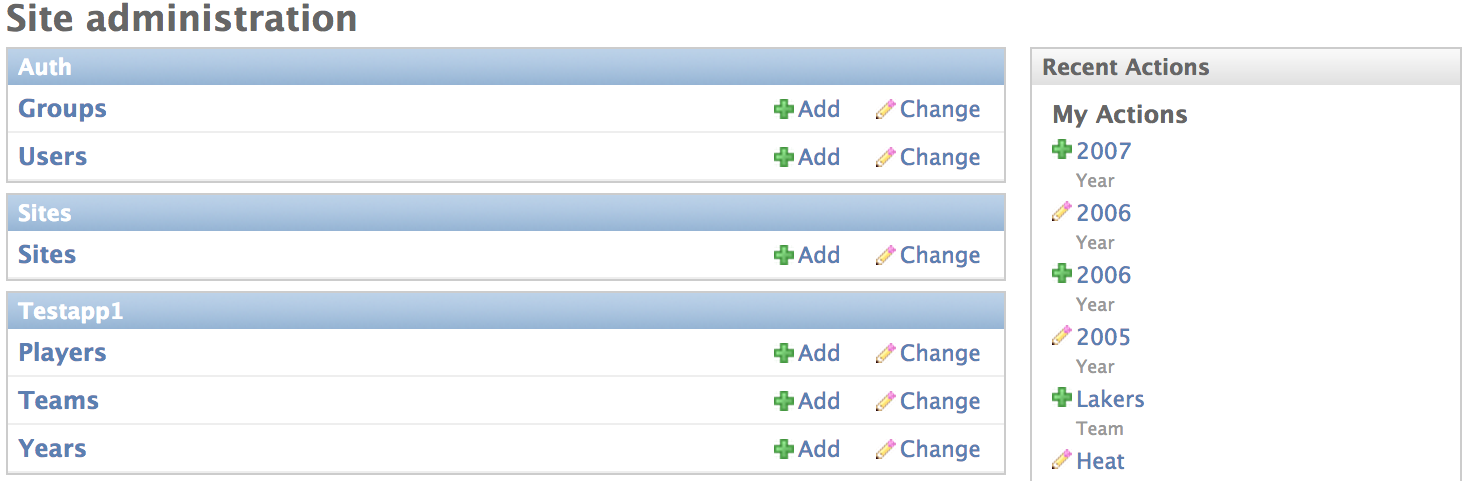
To make to static HTML pages that are served by Django on Pythonanywhere, one of the easiest ways to start is to follow the Django tutorial provided in Pythonanywhere.com (the exact link is “https://www.pythonanywhere.com/wiki/DjangoTutorial”). The Django tutorial is straightforward up to “Defining your urls” section. The tutorial instructs to fill in the first line of urls.py as “from django.conf.urls.defaults import patterns, include, url” which would result in an error for users with Django 1.6 or later versions because “django.conf.urls.defaults” has been removed in Django 1.6. The correct way to import would be “from django.conf.urls import patterns, include, url” for Django 1.6 users.

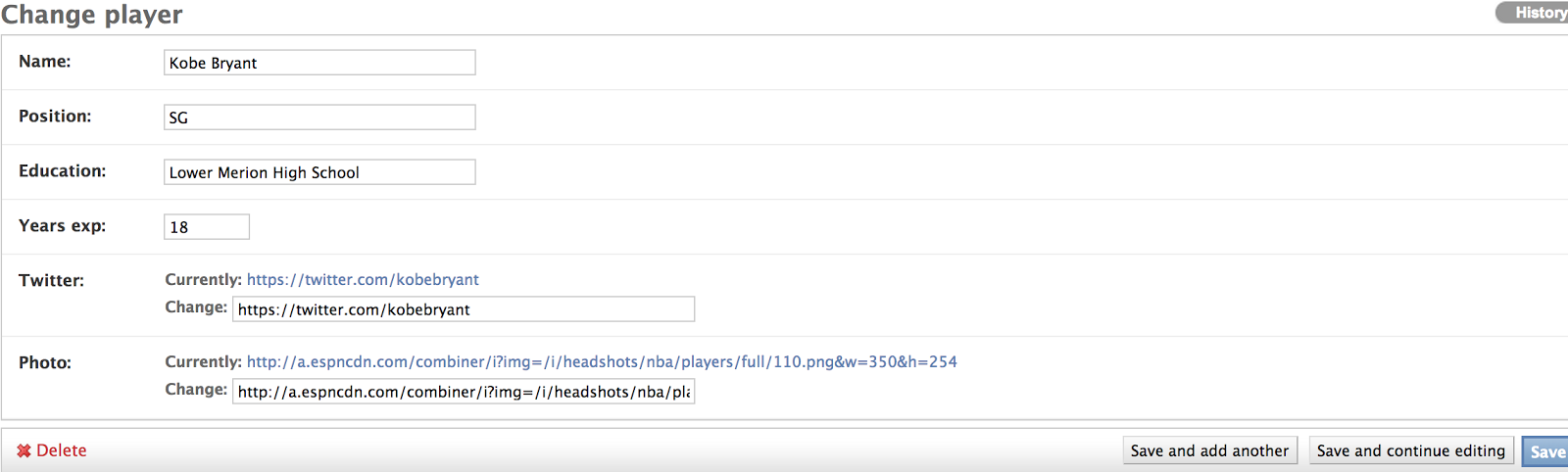
After finishing the tutorial, the user should have all the necessary ingredients for building basic html pages in his pythonanywhere.com directory. The user should now be able to access into “http://user-id.pythonanywhere.com/admin/” (“http://nbadb.pythonanywhere.com/admin/” in our case) and administer the Django-powered website. The first thing the administrator would want to do is adding class objects, which are defined in the models.py file, into the website. To do this, models.py file should be in the app directory along with views.py. Also, an admin.py file with information about the models.py class objects should be in the same directory as well.



By adding an admin.py file to the app directory as shown above, the administrator can now add class objects to the website as shown below.



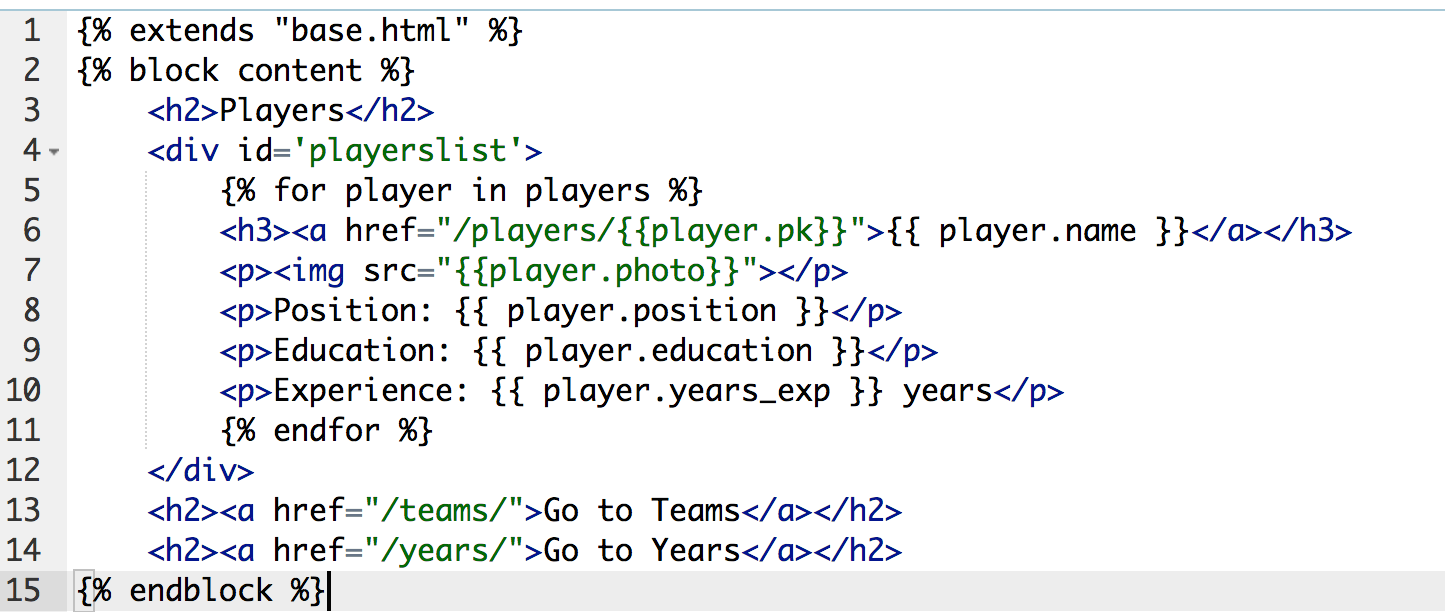
The classes that our model.py file define are “Players,” “Teams,” and “Year,” as shown above. By clicking the “Add” and “Change” buttons, the administrator can manipulate informations about the class objects (shown below) in the website.



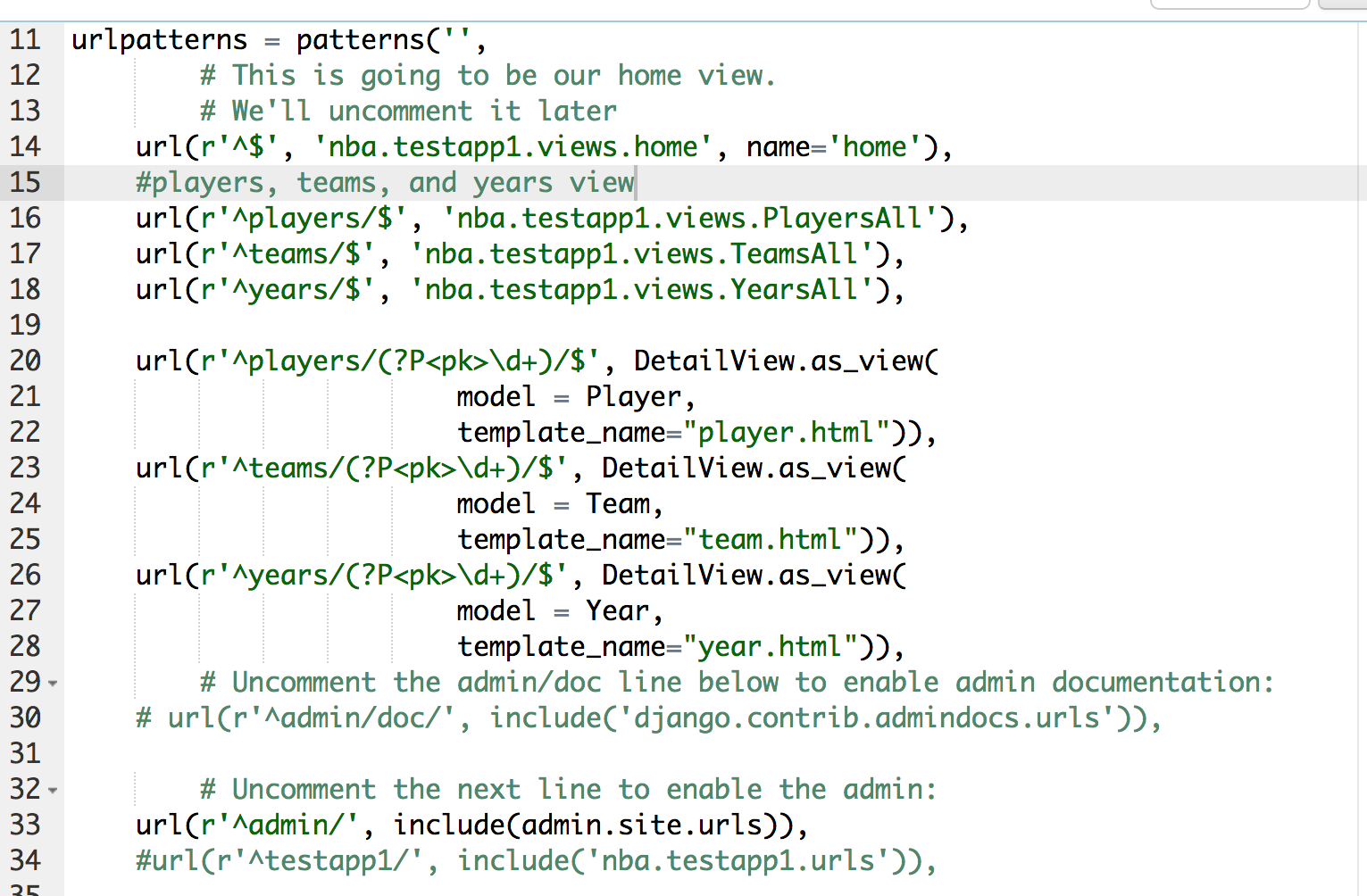
Once the necessary objects are added to the website, it is time to make the first html page of the website. The “home.html” was already made from the tutorial. In order to enable other html files to work and present them in the web, views.py and urls.py files have to be modified to handle each files. For the views.py file, “render\_to\_response” function was mainly used to handle the class objects (Documentations and more information about “render\_to\_response” function can be found in “https://docs.djangoproject.com/en/dev/topics/http/shortcuts/”). Our group’s views.py file so far looks as follows:



Apart from the request for “home.html,” all the requests in view.py returns a HttpResponse object, which makes it possible to display the appropriate html page such as “playersall.html,” which is displayed below.



The basic layout is the same for all the other html files (teamsall.html and yearsall.html): They all display the class objects’ information that the administrator added. At the bottom of the html block are the links to other html pages. The files also have links that present a more detailed view (using DetailView) of a certain object (for example, line 6 of “playersall.html”).



The urls.py file (from line 20 to 28), as present above, shows how DetailView was implemented (a full documentation and explanation can be found in “https://docs.djangoproject.com/en/dev/ref/class-based-views/generic-display/”).