In the part “Django Template Language” the language syntax of the Django template system are explained. Django’s template language is designed to strike a balance between power and ease. It’s also designed to feel comfortable for those used to working with HTML. Code examples from this unit help you to notice that if you have any exposure to other text-based template languages, such as Smarty or CheetahTemplate, you should feel right at home with Django’s templates. The template system is meant to express presentation, not program logic. So, a template is simply a text file. It can generate any text-based format (HTML, XML, CSV, etc.). A template contains variables, which get replaced with values when the template is evaluated, and tags, which control the logic of the template. Many tags, filters and syntax are supported by default, but although you can add your own extensions to the template language as needed. In this part filters and tags possibilities, their using and features are described more detail about. Special play is devoted to template inheritance, custom libraries and automatic HTML escaping (about different cases of using, how to turn it off and how to implement it in best way for each case). Also there are many useful notes about special cases and differences between versions.

In unit “Django Template Language : For Python Programmers” the Django template system it is explained from a technical perspective – how it works and how to extend it. There are described basics principles of template structure and its parts – block tags and variables. A lot of detail information about both steps of using template system process – compiling Template object and calling the render() method are there. Also you can investigate many small and easy examples, some of examples are not very easy to implement but easy to understand. They show, for instance, how invalid variables are handled, how to use built-in variables and Context objects. There are some useful tips about debug process and many other interesting notes. Python API, using sub-directories, main loaders types and alternative template language also are described there.

In part “Request And Response Objects” it is told more detailed about requests and special features of their realization with Django. Detailed overview of HttpRequest objects, their attributes, methods, differences between versions, parameters for requests, built-in properties and objects are proposed for your attention there. There are also lot information about passing iterators, setting header fields, HttpResponse subclasses, which is divided into paragraphs. There are declared most useful and expected novelties from Django 1.6. And in the last paragraph performance consideration is described. In this part there are not so many examples as it was in previous units, but there are quit full descriptions of all properties and methods.

If you know how to write simplest views you are ready for code optimization and using more Django advantages. So, unit “Class-Based Views” is really helpful for you this. There are short, but example-illustrated, descriptions of base and generic views, threads, class-based views, HTTP methods and specifications of all this “wonder”. Also it this part it is included detailed description of using mixins and handling forms with class-based views. Special place is given for decorating class-based views and possibilities of generic views of objects. Also there are a large part of basic information about dynamic filtering and template contexts, a few about models and their interaction with requests, for example, using AJAX. As it was said before, all information is supported by useful examples.

And the last part of overviewed professional literature calls “Using Mixins With Class-Based Views”. Its main purpose is to show and approve that Django provided a number of mixins that guaranteed more discrete functionality, than Django’s built-in class-based views. You can find overview of two main built-in Django's generic class-based views there: DetailView and ListView and different cases of their using with. Also there are described few variants of really good

solutions for typical problems, which you should solve during development of event the simplest web application.

This text, a small part of Django original documentation, was really useful for me, because I need to study Django for my diploma paper and want to use this Python framework for feature work projects. It's also one of most interesting direction of personal developing for me, don't looking for this is one of the leading web technologies in the world. Django gives to developers tools for quick and quality creation of complicate web tool, with both – client and server sides written in one programming language with addition of design decoration. It's really easy to start even if you haven't any experience of working with Python. And full documentation is enable for reading in many comfort formats. Besides clean documentation, which is easy to understand and supported by many examples, there are six parts of general tutorials for beginners and they are really helpful.