**Методические указания по выполнению практического задания**

**Целью** занятия является практическое освоение теоретических знаний, полученных в процессе изучения темы 5 «Веб-формы в Django. Валидация. JavaScript».

**Теория для практического занятия**

Для создания формы в веб-проекте Django в среде Visual Studio 2017 необходимо:

1. Запустить Visual Studio 2017.
2. Открыть в Visual Studio 2017 ранее созданный веб-проект Django.
3. Добавить описание класса новой формы в файл **app\forms.py**.
4. Добавить разметку формы в файл шаблона HTML-страницы в папке app\templates\app.
5. В файл **app\views.py** после строк **from ... import ...** добавить строку импорта созданного класса формы:  
   **from .forms import имя\_класса\_формы**
6. В файл **app\views.py**добавить метод действия, обрабатывающий запрос к странице с формой, выполняющий создание объекта класса формы и его передачу методу**render** по ключу **‘form’** в составе словаря, передаваемого третьим аргументом метода.

Для настройки валидации полей формы необходимо:

1. Для полей, заполнение которых не обязательно, в описании класса формы передать в функцию создания поля аргумент **required=False.**
2. Для ограничения минимальной и максимальной длины текстовых полей передать в функции создания таких полей аргументы **min\_length**и**max\_length**с соответствующими значениями**.**
3. Для ограничения минимального и максимального значения числовых полей передать в функцию создания таких полей аргументы **min\_value**и**max\_value**с соответствующими значениями**.**

Для приема данных из заполненной и отправленной пользователем формы необходимо:

1. Проверить в методе действия значение атрибута **request.method***.*Если его значение**‘POST’**, то создать экземпляр класса формы, передав в функцию создания объекта формы атрибут **request.POST**.
2. Если на шаге 1 объект формы был создан, вызвать метод **form.is\_valid()**. Если метод вернет True, получить данные формы из словаря **form.cleaned\_data**по ключам, являющимся текстовыми строками, содержащими имена полей класса формы.
3. Если метод **form.is\_valid()** вернул False, передать объект form методу render для дальнейшего редактирования содержания формы пользователем.
4. Если на шаге 1 атрибут **request.method** не имел значение **‘POST’**, создать новый экземпляр класса формы и передать его методу **render**для отображения пустой формы, которую должен будет заполнить пользователь.

Для передачи набора данных в шаблон страницы для отображения пользователю необходимо:

1. Создать в методе действия переменную-словарь вызовом функции **dict()**и сохранить в ней по ключам набор данных, который будет передаваться в шаблон страницы.
2. Включить созданный словарь как значение, записанное по определенному ключу, в словарь, который передаётся в методе действия методу**render**.

Для отображения в составе веб-страницы данных, переданных из метода действия необходимо:

Включить в вёрстку страницы переменные, содержащие значения из словаря с набором данных, переданным из метода действия. Например, если ключ словаря был **‘data’**, а ключ значения, содержащего имя пользователя - **‘name’**, то для вставки этого имени в шаблон страницы используется переменная **{{ data.name }}**.

Для создания шаблона страницы, отображающей либо форму, либо другие данные, переданные из метода действия, необходимо:

Использовать в шаблоне страницы оператор **if** для проверки наличия данных формы в переданном в шаблон словаре: **{% if form != None %}**. После этого может размещаться верстка формы, завершаемая**{% else %}**, после которого до **{% endif %}** будет размещена вёрстка данных, передаваемых в отсутствие данных формы.

**Задание 1**

Цель задания – освоить технологию создания форм и обработки данных, передаваемых через формы, в веб-проекте Django в среде Visual Studio 2017.

1. Добавить в проект страницу для отзывов пользователей сайта – pool.html.
2. Добавить соответствующий пункт, например, «Обратная связь», для перехода на страницу отзывов, в меню сайта.
3. Сформировать состав формы отзыва пользователя сайта, включающий в себя ряд вопросов к пользователю о его оценках различных сторон сайта и его пожеланиях. Форма должна включать в себя поля ввода, флажки, радиокнопки, список, блок ввода текста.
4. Описать в файле forms.py класс формы, содержащий требуемые поля формы. Настроить валидацию полей формы, соответствующую их назначению.
5. Добавить в файл views.py метод действия, который будет создавать объект формы и принимать данные из формы.
6. В случае успешного приема данных передавать из метода действия на страницу отзыва список принятых данных и выводить на этой странице благодарность за заполнение отзыва и оформленный список принятых от пользователя данных.

**Разбор задач, пошаговые примеры**

Рассмотрим пример решения задачи, подобной заданию 1 практического занятия по теме 5. В примере создаётся и обрабатывается форма анкеты пользователя сайта.

1. Из теории в подразделе **Теория для практического занятия** следует, что первоначально необходимо:

a. Запустить на своем компьютере **Visual Studio 2017**.

b. Далее в меню **Файл** выбрать команду **Открыть**, далее выбрать **Решение или проект..**. В появившемся окне «**Открыть проект**» перейти в папку ранее созданного проекта, выбрать там файл с расширением .sln и нажать кнопку «Открыть».

c. Далее в окне «Обозреватель решений» выбрать и открыть для редактирования файл **app\forms.py**. Этот файл следует дополнить описанием класса создаваемой формы. На рисунке 1 представлен текст класса модели формы анкеты пользователя сайта, содержащей следующие поля: имя, город, специальность, пол, вопрос о частоте использования интернета, e-mail, согласие на отправку новостей сайта, короткое резюме.

Комментарий к тексту класса формы из примера на рисунке 1.

Поля *name*, *city*, *job*, *gender* являются однострочными текстовыми полями ввода и создаются как экземпляры класса **forms.CharField**. Аргумент **label** задаёт тестовую подпись поля, **min\_length** и **max\_length** – ограничения на длину вводимого текста, используемые при валидации.

Поле *gender* задаёт две радиокнопки и создаётся как экземпляр класса **forms.ChoiceField**. Аргумент **widget** задает виджет **forms.RadioSelect**, который обеспечивает отображение этого поля как набора радиокнопок. Аргумент **choices** содержит список кортежей, задающих пары “значение”-“надпись” для радиокнопок одной группы.Эти параметры указываются как текстовые строки. Значения полей могут содержать не только числа, как в примере, но и другой текст. Аргумент **initial=1** задает выбранную по умолчанию кнопку, в данном случае - первую кнопку группы.

Поле **internet** задаёт раскрывающийся список и создаётся как экземпляр класса **forms.ChoiceField** без указания виджетов. Через аргумент **choices** задается кортеж кортежей, содержащий пары “значение”-“надпись” для отдельных строк списка. Аргумент **initial=1** задает выбор по умолчанию для первой строки списка.

Поле **notice** задаёт флажок и создаётся как экземпляр класса **forms.BooleanField**. В метод действия для этого поля передаётся логическое значение **True** или **False** в зависимости от того, выбран флажок или сброшен. Аргумент **required=False** необходим, когда у пользователя должна быть возможность не выбирать флажок. Поле **email** создается как экземпляр класса **forms.EmailField**. Это тип обеспечивает валидацию поля ввода текста в соответствии со стандартным форматом e-mail. Аргумент **min\_length** задаёт дополнительное условие валидации – ограничение минимальной длины.

Поле **message** является экземпляром класса **forms.CharField**, как и другие поля ввода, но для него в качестве значения атрибута **widget** указан экземпляр класса **forms.Textarea**, что обеспечивает создание многострочного поля ввода. При создании экземпляра **forms.Textarea** передается аргумент **attrs**, содержащий словарь параметров, задающих число текстовых строк и столбцов.

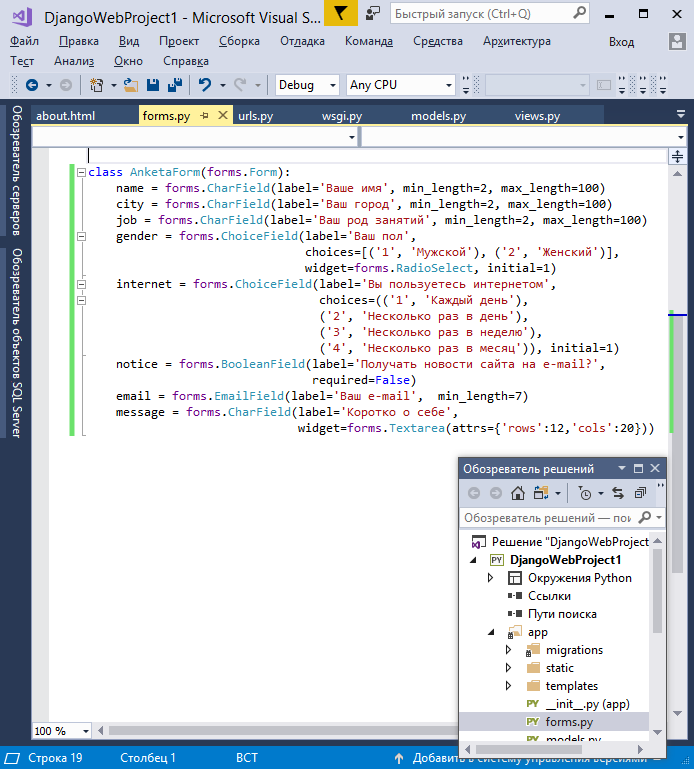


Рисунок 1 – Описание класса формы

1. Из теории в подразделе **Теория для практического занятия** следует, что после создания класса формы следует добавить разметку формы в файл шаблона HTML-страницы в папке **app\templates\app**. Добавление новой веб-страницы в проект Django рассматривалось в самостоятельной работе 4. Для этого требуется:

a) Добавить файл шаблона страницы в папку **app\templates\app.** В примере добавляется файл **anketa.html**. Разметка в файле построена таким образом, чтобы отображать форму в том случае, когда в шаблон передаётся переменная формы **form**, не равная **None** (пустому значению). В противном случае подразумевается, что переменная **data**, переданная в шаблон страницы, содержит поля для отображения результатов отправки формы, и выполняется отображение этих данных. Разметка шаблона страницы анкеты приведена на рисунке 2.

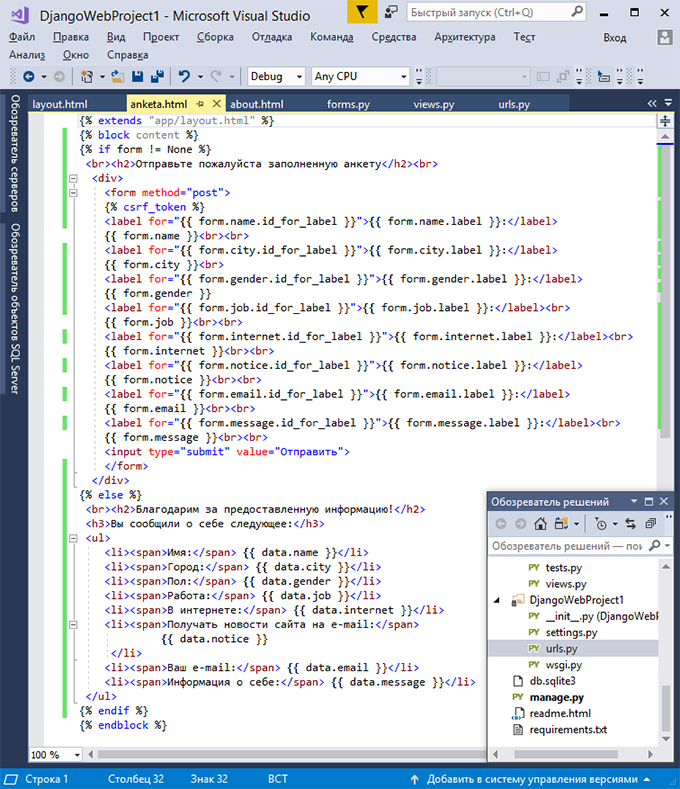


Рисунок 2 – Разметка шаблона страницы, содержащей форму

б) Добавить в файл **views.py** строку импорта класса формы.  
В данном примере это строка   
*from .forms import AnketaForm* Затем следует добавить в файл **views.py** метод, который будет обрабатывать запросы к добавленной странице.В примере добавляется метод *def anketa(request)*. Добавить в файл **views.py** метод, который будет обрабатывать запросы к добавленной странице.В примере добавляется метод *def anketa(request)*. Текст метода приведен на рисунке 3. В начале кода метода создаются словари **gender** и **internet**, используемые в дальнейшем для получения текста выбранных пользователем полей по их значению, передаваемому из формы. Далее, если в метод передана заполненная форма методом POST, выполняется валидация формы, и, в случае успешного результата, данные полей формы переносятся в словарь **data**. Если валидация пройдёт неудачно, заполненная форма будет передана обратно в шаблон страницы для редактирования. Если же форма не была отправлена методом POST, то создаётся пустая новая форма для заполнения её пользователем. Затем вызывается метод **render**, которому в последнем аргументе передаётся словарь, содержащий переменные **form** и **data**.

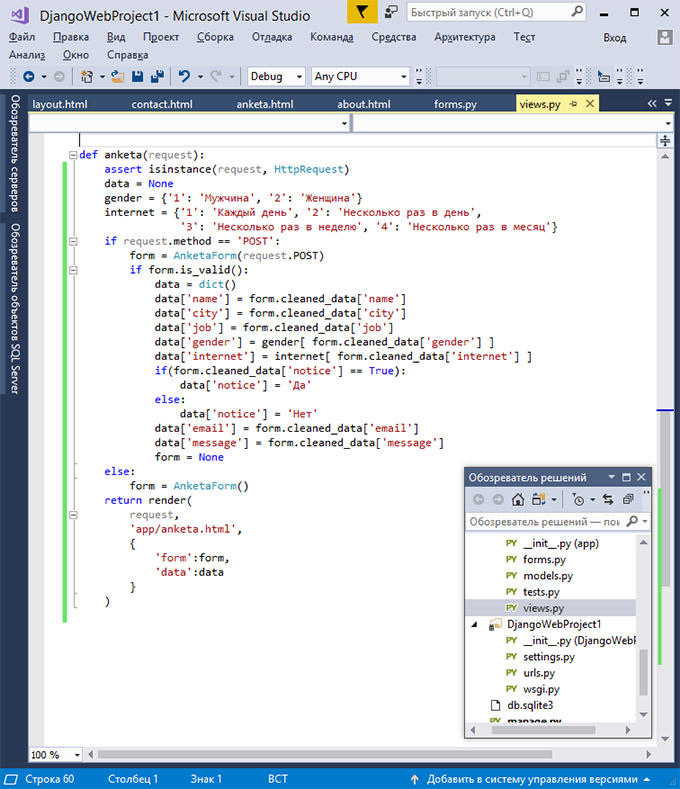


Рисунок 3 – Код метода, обрабатывающего запрос страницы с формой

в) В файл **templates\app\layout.html** добавить в список пунктов меню (тег **<ul>**) строку для добавляемой веб-страницы. В рассматриваемом примере это будет строка

<li><a href="{% url 'anketa' %}">Анкета</a></li>

г) В файл **urls.py** добавить URL для добавляемой веб-страницы. В рассматриваемом примере это будет строка

url(r'^anketa$', app.views.anketa, name='anketa'),

На рисунке 4 представлен вид страницы с анкетой при переходе на неё пользователем через меню сайта. При оформлении страницы использованы стилевые правила из файла **site.css**.

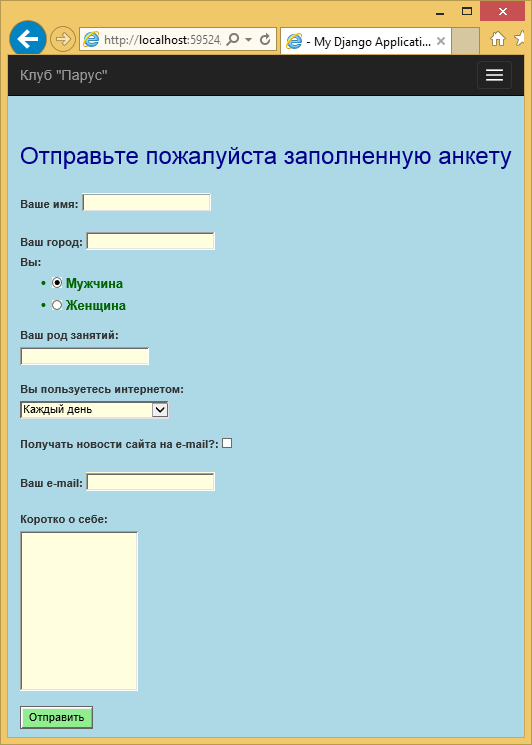


Рисунок 4 – Вид загруженной страницы с формой

На рисунке 5 приведен пример отображения информации на странице после успешной отправки анкеты на сервер.

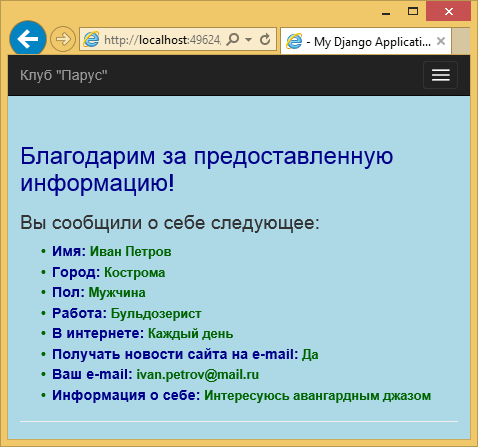


Рисунок 5 – Отображение информации об успешном приеме анкеты