Формы

№ урока:	7	**Курс:**	Django Starter

Средства обучения:	Персональный компьютер с установленными: Python 3.8.2 Django 3.0.4

Обзор, цель и назначение урока

Цель данного урока - ознакомиться с основами работы Django форм, научиться создавать собственные формы, настраивать и подключать внедренные в Django формы. Также в процессе урока будет рассмотрено как с помощью Django форм можно упростить и ускорить разработку приложения. Дополнительно будут рассмотрены поля форм, виджеты и работа с медиа файлами.

Изучив материал данного занятия, учащийся сможет:

- Создавать собственные формы с правильными полями.
- Использовать существующие формы для ускорения работы.
- Работать с медиа файлами в рамках форм.
- Использовать виджеты.

Содержание урока

1) Рассмотрения общего понятия что такое формы
2) Рассмотрения форм в рамках Django приложения
3) Создание простой формы
4) Рассмотрение полей формы
5) Валидация полей формы
6) Рассмотрение что такое виджеты и как их можно использовать вместе с формами
7) Добавление в форму медиа файлов

Резюме

- Если вы планируете создавать сайты и приложения, которые принимают и сохраняют данные от пользователей, вам необходимо использовать формы. Django предоставляет широкий набор инструментов для этого.
- Форма в HTML – это набор элементов в <form>...</form>, которые позволяют пользователю вводить текст, выбрать опции, изменять объекты, контролировать страницы и так далее, а потом отправлять эту информацию на сервер.
- Некоторые элементы формы - текстовые поля ввода и чекбоксы - достаточно простые и встроены в HTML. Некоторые – довольно сложные, состоят из диалогов выбора даты, слайдеров и других контролов, которые обычно используют JavaScript и CSS.
- GET и POST – единственные HTTP методы, которые используются для форм.
  Форма авторизации в Django использует POST метод. При отправке формы браузер собирает все данные формы, кодирует для отправки, отправляет на сервер и получает ответ.
- Не следует использовать GET запросы для формы с паролем, т.к. пароль появится в URL, а следовательно - в истории браузера и журналах сервера. Также он не подходит для отправки большого количества данных или бинарных данных, например, изображения.
- GET удобен для таких вещей, как форма поиска, т.к. URL, который представляет GET запрос, можно легко сохранить в избранное или отправить по почте.

CyberBionic systematics

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Формы Django могут упростить и автоматизировать большую часть этого процесса, и могут сделать это проще и надежнее, чем код, написанный большинством программистов. Django позволяет:
    - подготовить данные для отображения в форме;
    - создать HTML формы для данных;
    - получить и обработать отправленные формой данные.
- Сердце всего механизма – класс Form. Как и модель в Django, которая описывает структуру объекта, его поведение и представление, **Form** - описывает форму, как она работает и показывается пользователю.
- Как поля модели представляют поля в базе данных, поля формы представляют HTML <input> элементы.
- Поля формы сами являются классами. Они управляют данными формы и выполняют их проверку при отправке формы. Например, DateField и FileField работают с разными данными и выполняют разные действия с ними.
- Поле формы представлено в браузере HTML “виджетом” - компонентом интерфейса. Каждый тип поля представлен по умолчанию определенным классом Widget, который можно переопределить при необходимости.
- Создание формы происходит через класс Form, который импортируется из пакета django.forms.
- Экземпляр Form содержит метод is_valid(), который выполняет проверку всех полей формы. Если все данные правильные, это метод:
    - вернет True;
    - добавит данные формы в атрибут cleaned_data.
- Данные формы отправляются обратно в Django и обрабатываются представлением, обычно тем же, которое и создает форму. Это позволяет повторно использовать часть кода.
- При создании класса Form наиболее важной деталью является определение полей формы. Каждое поле обладает собственной логикой проверки вводимых данных наряду с дополнительными возможностями.
- Каждый конструктор класса Field принимает эти аргументы. Некоторые классы Field принимают дополнительные аргументы. Перечисленные ниже аргументы принимаются всеми полями:
    - required - по умолчанию каждый класс Field предполагает значение обязательным. Таким образом, если вы передадите ему пустое значение, т.е. None или пустую строку (""), то метод clean() вызовет исключение ValidationError
    - label - аргумент label позволяет вам определить “видимую людьми” метку для этого поля. Оно используется, когда Field отображается на форме.
    - label_suffix - суффикс, который будет добавлен к надписи для текущего поля.
    - initial - аргумент initial позволяет определять начальное значение для поля, при его отображении на незаполненной форме.
    - widget - аргумент widget позволяет указать класс Widget, который следует использовать при отображении поля.
    - help_text - аргумент help_text позволяет указать описание для поля. Если вы укажете help_text, он будет показан около поля при отображении формы с помощью вспомогательных методов Form (например, через as_ul()).
    - error_messages - аргумент error_messages позволяет изменить стандартные сообщения об ошибках, которые выдает поле. Создайте словарь с ключами тех сообщений, которые вы желаете изменить.
    - validators - аргумент validators позволяет указать список функций, осуществляющих проверку поля.
    - localize - аргумент localize включает локализацию для данных формы, как на входе, так и на выходе.

CyberBionic systematics

CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Поля форм:
    - BooleanField - возвращает: True или False языка Python.
    - CharField - используется для ввода строки.
    - ChoiceField - выбор из списка.
    - DateField – дата.
    - DurationField - отрезок времени.
    - EmailField - проверяет, что полученное значение является правильным адресом электронной почты, используя достаточно сложное регулярное выражение.
    - FileField - возвращает объект UploadedFile, который оборачивает содержимое файла и его имя в единый объект.
    - FloatField - проверяет, что полученное значение является числом с плавающей точкой.
    - IntegerField - Используется для хранения ID.
    - ImageField - проверяет, что данные файла были связаны с формой, а затем, что файл является изображением, формат которого поддерживается библиотекой Pillow.
- Главной задачей объекта Form является проверка данных. У заполненного экземпляра Form вызовите метод is_valid() для выполнения проверки и получения её результата.
- Обратитесь к атрибуту errors для получения словаря с сообщениями об ошибках. В этом словаре, ключами являются имена полей, а значениями – списки юникодных строк, представляющих сообщения об ошибках. Сообщения хранятся в виде списков, так как поле может иметь множество таких сообщений.
- Каждое поле в классе Form отвечает не только за проверку, но и за нормализацию данных. Это приятная особенность, так как она позволяет вводить данные в определенные поля различными способами, всегда получая правильный результат.
  Например, класс DateField нормализует введённое значение к объекту datetime.date. Независимо от того, передали ли вы строку в формате '1994-07-15', объект datetime.date или число в других форматах, DateField всегда преобразует его в объект datetime.date, если при этом не произойдёт ошибка.
  После создания экземпляра Form, привязки данных и их проверки, вы можете обращаться к “чистым” данным через атрибут cleaned_data.
- Второй задачей объекта Form является представление себя в виде HTML кода. Для этого объект надо просто “распечатать”.
- Для гибкости, выводимый код не включает в себя ни теги <table> и </table>, ни теги <form> and </form>, ни тег <input type="submit">. Не забывайте их добавлять.
- Каждый тип поля имеет стандартное HTML представление. Тип CharField``представлен как ``<input type="text">, a EmailField как <input type="email">. Тип BooleanField представлен <input type="checkbox">. Следует отметить, что эти представления достаточно гибкие, так как вы можете влиять на них, указав для поля виджет.
- Атрибут name каждого тега совпадает напрямую с именем атрибута в классе.
- Текстовая метка каждого поля, т.е. 'Subject:', 'Message:' и 'Cc myself:' генерируется из имени поля, конвертируя символы подчеркивания в пробелы и переводя первый символ в верхний регистр. Также, вы можете явно назначить текстовую метку для поля.
- Каждая текстовая метка выводится с помощью тега <label>, который указывает на соответствующее поле формы с помощью атрибута id. Атрибут id генерируется путём добавления префикса 'id_' к имени поля. Атрибуты id и теги <label> включаются в HTML представление формы по умолчанию, но можете изменить такое поведение формы.
- Метод as_p() представляет форму в виде последовательности тегов <p>, по одному на каждое поле.
- Метод as_table() выводит форму в виде таблицы. Этот метод используется по умолчанию.
- **Виджет** – это представление поля в виде HTML кода. Виджеты обеспечивают генерацию HTML и извлечение соответствующих данных из GET/POST запросов.

CyberBionic systematics
CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Не следует путать виджеты с полями формы. Поля формы обеспечивают логику проверки вводимой информации и используются в шаблонах. Виджеты же отвечают за рендеринг форм на веб-странице и обработку переданных данных. Тем не менее, виджеты следует назначать на поля формы.
- При добавлении поля на форму, Django использует стандартный виджет, наиболее подходящий к отображаемому типу данных. Для того, чтобы узнать какой виджет использует интересующий вас тип поля, обратитесь к built-in fields.
- Django с помощью модуля django.forms.widgets обеспечивает представление для всех базовых HTML виджетов, а также некоторые часто используемые группы виджетов, включая виджеты для ввода текста, различные чекбоксы и селекторы, загрузку файлов и обработку сложных значений.

Закрепление материала

- Что такое Django Forms (формы)?
- Какие есть типы полей у форм?
- Что такое виджеты?
- Какие есть встроенные формы?
- Как происходит валидация формы?
- Как добавить форму на страницу шаблона?

Дополнительное задание

Задание
Создать форму для отправки отзыва на продукт для интернет-магазина. Форма должна вмещать:

- Файл с картинкой.
- Почту пользователя.
- Описание.
- Выбор оценки.
- Негативный или позитивный отзыв.
- Номер телефона.

Самостоятельная деятельность учащегося

Задание 1
Изучить и понять все преимущества и недостатки инструментов, которые были рассмотрены на уроке.

Задание 2
Создать несколько форм для авторизации пользователя.

Задание 3
Найти информацию что такое ModelForm и создать несколько таких форм.

Рекомендуемые ресурсы

Официальная документация Django:
https://www.djangoproject.com/
https://docs.djangoproject.com/en/3.0/topics/forms/modelforms/

CyberBionic systematics
CyberBionic Systematics ® 2020
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com