

Django Starter

Формы

Django Starter

Introduction



Лазорык Михаил

Software developer, 3 года опыта

 mykhailo.lazoryk

 mykhailo-lazoryk



Django Starter

Формы

Django Starter

План урока

1. Рассмотрения общего понятия что такое формы
2. Создание простой формы
3. Рассмотрение полей формы
4. Валидация полей формы
5. Рассмотрение что такое виджеты
6. Добавление в форму медиа файлов

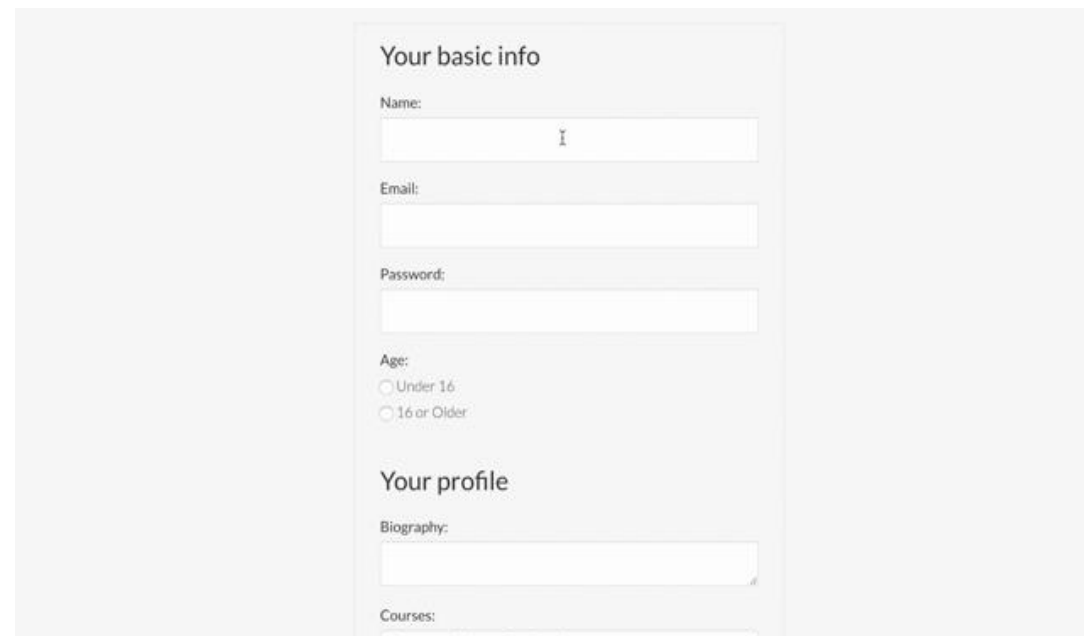
Django Starter

Рассмотрения общего понятия что такое формы

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**.



The image shows a web form with two main sections: 'Your basic info' and 'Your profile'. The 'Your basic info' section includes input fields for 'Name:', 'Email:', and 'Password:', followed by an 'Age:' section with two radio button options: 'Under 16' and '16 or Older'. The 'Your profile' section includes a 'Biography:' text area and a 'Courses:' field.

Django Starter

Рассмотрения общего понятия что такое формы

- **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.

Django Starter

Что такое Django формы

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.

Django Starter

Что такое Django формы

- 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.

Django Starter

Создание формы

- 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, этот 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.

Django Starter

Аргументы полей формы

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**
- **initial** - аргумент `initial` позволяет определять начальное значение для поля, при его отображении на незаполненной форме.

Django Starter

Аргументы полей формы

- **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.

Django Starter

Поля формы

- **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.

Django Starter

Валидация формы

Главной задачей объекта Form является проверка данных. Для этого 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.

First name: Mark (Looks good!)

Last name: Otto (Looks good!)

Username: @ Username (Please choose a username.)

City: City (Please provide a valid city.)

State: State (Please provide a valid state.)

Zip: Zip (Please provide a valid zip.)

☐ Agree to terms and conditions
You must agree before submitting.

Submit form

☐ Check this custom checkbox
Example invalid feedback text

☐ Toggle this custom radio

☐ Or toggle this other custom radio
More example invalid feedback text

Open this select menu (Example invalid custom select feedback)

Choose file... (Example invalid custom file feedback) Browse

Django Starter

Валидация формы

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 введенное 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 произойдет ошибка.

После создания экземпляра `Form`, привязки данных и их проверки, вы можете обращаться к **"чистым"** данным через атрибут **`cleaned_data`**

☒ Check this custom checkbox

☐ Toggle this custom radio

☐ Or toggle this other custom radio

One

Choose file...

Browse

Django Starter

Отображение формы

Второй задачей объекта Form является представление себя в виде HTML кода. Для этого объект надо просто “распечатать”.

```
>>> form = ContactForm()
```

```
>>> print(form)
```

```
<tr><th><label for="id_subject">Subject:</label></th><td><input id="id_subject" type="text" name="subject" maxlength="100" /></td></tr>
```

```
<tr><th><label for="id_message">Message:</label></th><td><input type="text" name="message" id="id_message" /></td></tr>
```

```
<tr><th><label for="id_sender">Sender:</label></th><td><input type="email" name="sender" id="id_sender" /></td></tr>
```

```
<tr><th><label for="id_cc_myself">Cc myself:</label></th><td><input type="checkbox" name="cc_myself" id="id_cc_myself" /></td></tr>
```

Django Starter

Отображение формы

- 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">`, а **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 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 вы можете изменить такое поведение формы.

Django Starter

Отображение формы

Метод `as_p()` представляет форму в виде последовательности тегов `<p>`, по одному на каждое поле.

```
<p><label for="id_subject">Subject:</label>
```

```
<input id="id_subject" type="text" name="subject" maxlength="100" required></p>
```

```
<p><label for="id_message">Message:</label>
```

```
<textarea name="message" id="id_message" required></textarea></p>
```

Метод `as_table()` выводит форму в виде таблицы. Этот метод используется по умолчанию:

```
<tr><th><label for="id_subject">Subject:</label></th><td><input id="id_subject" type="text" name="subject" maxlength="100"
required></td></tr>\n<tr><th><label for="id_message">Message:</label></th><td><input type="text" name="message"
id="id_message" required></td></tr>\n<tr><th><label for="id_sender">Sender:</label></th><td></td></tr>
```

Django Starter

Виджеты

- **Виджет** – это представление поля в виде HTML кода. Виджеты обеспечивают генерацию HTML и извлечение соответствующих данных из GET/POST запросов.
- 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.

Django Starter

Добавление в форму медиа файлов (практическая часть)



Django Starter

План урока

1. Рассмотрения общего понятия что такое формы
2. Создание простой формы
3. Рассмотрение полей формы
4. Валидация полей формы
5. Рассмотрение что такое виджеты
6. Добавление в форму медиа файлов

Проверка знаний

TestProvider.com



Проверьте как Вы усвоили данный материал на TestProvider.com

TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.

Django Starter

Спасибо за внимание! До новых встреч!



Лазорык Михаил
Software developer



Информационный видеосервис для разработчиков программного обеспечения

