# Разработка программного обеспечения на языке Python



## Лекция 5. Модель пользователя

Посмотрите видеоуроки и ответьте на контрольные вопросы после лекции

## Модель пользователя


00:00 / 02:48

В данной теме рассмотрим модель пользователя, подключаемую к проекту джанго по умолчанию и позволяющую создавать и редактировать пользователей.

Django содержит систему идентификации пользователей, предоставляя работу с пользователями, правами и группами пользователей.

**Система идентификации пользователей состоит из следующих элементов:**

- Пользователи,
- Права: Булево значение , определяющее имеет ли пользователь права на различные действия.
- Группы: предоставляют возможность назначить множество прав нескольким пользователям.

Система идентификации пользователей в Django представлена приложением django.contrib.auth. Данное приложение устанавливается и подключается к нашему проекту джанго сразу же при его создании. Список установленных приложений находится в файл settings.py в в INSTALLED_APPS.

```
INSTALLED_APPS = [
    'example_app',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

**Модель пользователя** применяется в Django для создания в веб-приложении личного кабинета пользователей, разграничении прав и возможностей пользователя, а также для того, чтобы привязывать контент к пользователям. Например, в некотором блоге могут быть созданы статьи, которые в схеме базы данных будут соотноситься с написавшими их пользователями. При этом можно выдать права на редактирования только автору статьи.

Таким образом, по умолчанию в проект подключается приложение, связанное с авторизацией пользователей. При этом в БД создаются таблицы пользователи и группы.

**Таблица User содержит следующие поля:**

- **Username** Обязательное поле. Имена пользователей могут содержать только буквы, цифры и подчеркивание.
- **first_name и last_name** Необязательные строковые поля длиной до 30 символов.
- **Email** Необязательное поле, хранит адрес электронной почты.
- **Password** В нем хранится Хеш пароля и метаданные о нем. (Django не сохраняет открытый пароль.)
- Также к атрибутам модели пользователя относятся следующие поля.
- **is_staff** Указывает, имеет ли пользователь доступ к интерфейсу администратора.
- is_active. Указывает, следует ли считать эту учетную запись пользователя активной. Рекомендуется вместо удаления учетных записей установить этот флаг в False.
- **is_superuser** указывает, что данный пользователь является администратором с максимальным набором возможностей на сайте.
- **last_login и date_joined – соответственно в**ремя последнего входа пользователя и создания аккаунта.
- is_authenticated Это способ определить, был ли пользователь аутентифицирован. То есть зашел ли он в свой аккаунт в настоящий момент.

Модели классов, описывающих таблицы БД в джанго, кроме атрибутов, могут содержать также методы. Они предназначены для описания действий с определенной строкой – записью в таблице. Класс User также содержит методы, которые позволяют получать доступ или менять значение атрибутов для конкретного пользователя.
**Рассмотрим некоторые методы:**

- методы get_username, get_full_name, get_short_name Возвращают значение атрибутов имени пользователя.
- Метод set_password Устанавливает пароль.
- Метод get_all_permissions Возвращает набор строк разрешений, которые имеют пользователи.
- Метод email_user отправляет электронное письмо пользователю.

Модель User использует менеджер, который предоставляет дополнительные методы, предназначенные для выполнения действий с таблицей – в основном добавления новой записи. Метод create_user создает и сохраняет новый объект User с параметрами username, email и пароль, равными переданным аргументам Метод create_superuser также создает запись пользователя, но с правами администратора.

Для того, чтобы использовать таблицу пользователь, требуется в файле с описанием моделей импортировать данную модель из модуля django.contrib.auth.models. После чего она может быть применена в схеме базы данных нашего проекта. Например, можем в таблице добавить поля, ссылающиеся на таблицу User.

models.py

```python
from django.db import models
from django.contrib.auth.models import User


class Article(models.Model):
    name = models.CharField(max_length=50)
    text = models.CharField(max_length=500)
    author = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return str(self.name)+" "+str(self.author.name)
```

Самый простой способ создать пользователя - вызывать метод create user. В качестве параметров данного метода обязательно требуется задать логин пользователя (username) и пароль. Остальные параметры указываются по желанию. Для того, чтобы задать значение некоторым необязательным полям класса – можно обратится к данному полю как атрибуту созданного объекта класса User.

create_user(username, email=None, password=None)

```python
from django.contrib.auth.models import User
user = User.objects.create_user('user1', 'user1@mail.ru', 'user1password')

user.is_staff = True
user.save()
```

В данном примере после создания объекта мы задали ему атрибут is_staff=true. После чего с помощью метода save() сохранили запись о пользователе в базу данных.

Команда manage.py changepassword *username* позволяет изменить пароль пользователя через консоль.

```python
from django.contrib.auth.models import User

u = User.objects.get(username__exact='user1')
u.set_password('new password')
u.save()
```

Подведем итоги, в данной теме мы рассмотрели модель пользователя, ее основные атрибуты и методы, а также способ добавления ее к проекту. В следующей теме изучим способы расширить данную модель путем добавления новых полей в таблицу.

Панель администратора Django

ПРЕДЫДУЩИЙ ЭЛЕМЕНТ КУРСА

◄ Задание 8. Вывод данных

Перейти на...

СЛЕДУЮЩИЙ ЭЛЕМЕНТ КУРСА

Задание 9. Личный кабинет администратора ►

**Контакты** +7(391) 206-27-05
info-ms@sfu-kras.ru