skypro

# Права доступа. Шпаргалка

**Права доступа (разрешения, пермишенны, Permissions)** — представляют собой систему правил, которые контролируют, какие действия могут выполнять пользователи в веб-приложении.

## Модули, отвечающие за права доступа в Django

**Модуль** `django.contrib.auth` — это основной модуль Django для управления пользователями и правами доступа. Он предоставляет средства для аутентификации, управления пользователями и группами, а также управления разрешениями.

**Модуль** `django.contrib.contenttypes` — позволяет работать с универсальными отношениями, то есть создавать связи между моделями различных типов.

## Настройка прав доступа для пользователя

В Django разрешения создаются автоматически для каждой модели при выполнении миграций.

- `add_modelname` — разрешение на добавление объекта модели.
- `change_modelname` — разрешение на изменение объекта модели.
- `delete_modelname` — разрешение на удаление объекта модели.
- `view_modelname` — разрешение на просмотр объекта модели.

## Изменение и расширение прав доступа

1. Удаление разрешений — можно удалить конкретное разрешение у пользователя, если оно больше не требуется.
2. Добавление новых разрешений — можно добавить новые разрешения для расширения возможностей пользователя.

```
# Удаление разрешения
user.user_permissions.remove(permission)

# Добавление нового разрешения
new_permission = Permission.objects.get(codename='change_modelname')
user.user_permissions.add(new_permission)
```

Пример:

```
from users.models import CustomUser
from django.contrib.auth.models import Permission

user = CustomUser.objects.get(username='example')
permission = Permission.objects.get(codename='add_modelname')
user.user_permissions.add(permission)
```

## Ограничение прав доступа к страницам

**Примеси (mixins) в Django** — представляют собой классы, которые добавляют определенные функциональные возможности к другим классам, например к классам представлений. Примеси помогают избежать дублирования кода и упрощают управление правами доступа.

1. Для FBV **-контроллеров:**

```
from django.contrib.auth.decorators import login_required, permission_
required

@login_required
@permission_required('app_label.add_modelname', raise_exception=True)
def my_view(request):
    # Код представления
    pass
```

2. Для CBV **-контроллеров:**

```
from django.contrib.auth.mixins import LoginRequiredMixin, PermissionR
equiredMixin
from django.views.generic import View

class MyView(LoginRequiredMixin, PermissionRequiredMixin, View):
    permission_required = 'app_label.add_modelname'
    # Код представления
    pass
```

## Настройка прав доступа для групп

### Создание группы

```
from django.contrib.auth.models import Group

# Создаем новую группу «Редакторы»
editors_group = Group.objects.create(name='Editors')
```

### Назначение разрешений группе

```
from django.contrib.auth.models import Permission

# Получаем разрешения
add_permission = Permission.objects.get(codename='add_modelname')
change_permission = Permission.objects.get(codename='change_modelname')

# Назначаем разрешения группе
editors_group.permissions.add(add_permission, change_permission)
```

### Добавление пользователя в группу

```
from users.models import CustomUser

# Получаем пользователя
user = CustomUser.objects.get(username='example_user')

# Добавляем пользователя в группу «Редакторы»
user.groups.add(editors_group)
```

# Управление правами доступа и группами через админку

## Управление пользователями

1. Создание и редактирование пользователей:
    - Перейдите в раздел **Users**.
    - Вы можете создать нового пользователя, нажав кнопку **Add user**, или редактировать существующего пользователя, кликнув по его имени.
    - При создании или редактировании пользователя вы можете указать его основные данные (имя пользователя, пароль, электронная почта) и задать статус (активный, суперпользователь и т. д.).
2. Назначение разрешений пользователю:
    - При редактировании пользователя вы можете найти раздел **Permissions**.
    - В этом разделе вы можете назначить пользователю индивидуальные разрешения, выбрав их из списка **User permissions**.

## Управление группами

1. Создание и редактирование групп:
    - Перейдите в раздел **Groups**.
    - Вы можете создать новую группу, нажав кнопку **Add group**, или редактировать существующую группу, кликнув по ее имени.
    - При создании или редактировании группы вы можете задать имя группы и назначить ей разрешения.
2. Назначение разрешений группе:
    - При редактировании группы в разделе **Permissions** вы можете выбрать разрешения, которые будут присвоены всем пользователям, входящим в эту группу.
    - Выберите нужные разрешения из списка **Permissions** и сохраните изменения.

## Создание кастомных прав доступа для модели

Для создания кастомных прав доступа в Django используется атрибут `permissions` в метаклассе `Meta` вашей модели.

```
from django.db import models

class Student(models.Model):
    FIRST_YEAR = 'first'
    SECOND_YEAR = 'second'
    THIRD_YEAR = 'third'
    FOURTH_YEAR = 'fourth'

    YEAR_IN_SCHOOL_CHOICES = [
        (FIRST_YEAR, 'Первый курс'),
        (SECOND_YEAR, 'Второй курс'),
        (THIRD_YEAR, 'Третий курс'),
        (FOURTH_YEAR, 'Четвертый курс'),
    ]

    first_name = models.CharField(max_length=150, verbose_name='Имя')
    last_name = models.CharField(max_length=150, verbose_name='Фамилия')
    year = models.CharField(
        max_length=6,
        choices=YEAR_IN_SCHOOL_CHOICES,
        default=FIRST_YEAR,
        verbose_name='Курс'
    )

    def __str__(self):
        return f'{self.first_name} {self.last_name}'

    class Meta:
        verbose_name = 'студент'
        verbose_name_plural = 'студенты'
        ordering = ['last_name']
        permissions = [
            ("can_promote_student", "Can promote student"),
            ("can_expel_student", "Can expel student"),
        ]
```

Атрибут `permissions` принимает список кортежей, где каждый кортеж состоит из двух элементов: **кодового имени разрешения и описания разрешения.**

## Дополнительная проверка прав доступа в контроллере

Метод `has_perm` проверяет наличие у пользователя конкретного права и возвращает `True` или `False`. В контроллерах вы можете использовать этот метод для выполнения различных проверок.

```
from django.views.generic import View
from django.shortcuts import get_object_or_404, redirect
from django.contrib.auth.mixins import LoginRequiredMixin
from django.http import HttpResponseForbidden
from students.models import Student

class PromoteStudentView(LoginRequiredMixin, View):
    def post(self, request, student_id):
        student = get_object_or_404(Student, id=student_id)

        if not request.user.has_perm('students.can_promote_student'):
            return HttpResponseForbidden("У вас нет прав для перевода
студента.")

        # Логика перевода студента на следующий курс
        student.year = next_year(student.year)
        student.save()

        return redirect('students:student_list')
```

## Проверка прав доступа в шаблонах

Django предоставляет функционал `perms`, который позволяет проверять права доступа пользователя в шаблонах. Возвращает `True`, если пользователь имеет указанное право, и `False` в противном случае.

### Пример использования perms

```
<h1>Студенты</h1>

<ul>
    {% for student in students %}
        <li>{{ student.first_name }} {{ student.last_name }}</li>

        {% if perms.students.can_promote_student %}
            <button>Перевести студента</button>
        {% endif %}

        {% if perms.students.can_expel_student %}
            <button>Исключить студента</button>
        {% endif %}
    {% endfor %}
</ul>
```