




# Welcome to the CoGrammar

## Django 4

**The session will start shortly...**

**Questions? Drop them in the chat. We'll have dedicated moderators answering questions.**



# Software Engineering Session Housekeeping

---

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.  
**(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions](#)

## Software Engineering Session Housekeeping cont.

---

- For all **non-academic questions**, please submit a query:  
[www.hyperiondev.com/support](http://www.hyperiondev.com/support)
- Report a **safeguarding** incident:  
[www.hyperiondev.com/safeguardreporting](http://www.hyperiondev.com/safeguardreporting)
- We would love your **feedback** on lectures: [Feedback on Lectures](#)

# Skills Bootcamp Progression Overview

## ✓ Criterion 3: Course Progress

- **Completion:** All mandatory tasks, including Build Your Brand and resubmissions by study period end
- **Interview Invitation:** Within 4 weeks post-course
- **Guided Learning Hours:** Minimum of 112 hours by support end date (10.5 hours average, each week)

## ✓ Criterion 4: Demonstrating Employability

- **Final Job or Apprenticeship Outcome:** Document within 12 weeks post-graduation
- **Relevance:** Progression to employment or related opportunity

# Learning Outcomes

- Explain what Permissions are in Django
- Create Permissions in their Django projects.
- Assign Permissions to a user in their projects.
- Explain what Groups are in Django.
- Create Groups in their Django projects.
- Assign Permissions to Groups.
- Assign Group to a user to apply Permissions of Group to the user.

**SKILLS  
FOR LIFE**

**SKILLS BOOTCAMPS**



Department  
for Education

# CoGrammar

## Permissions

April 2024



# Data Security

- Web applications deal with **lots of data**.
- Some data is **sensitive** and we **don't** want malicious actors to **gain access** to that data.
- We use ideas such as **encryption** and **hashing** to protect the data we store.
- This allows us to protect data even when malicious actors have gain access to it.
- We want to block malicious actors from accessing the data in the first place.

# Permissions

- Our applications have **multiple tables** in our databases and we would like to **limit** who has **access** where.
- Permission will allow us to do just that.
- We can set a **permission** for a **specific table** and only allow a **user** with the **correct** permissions to **access** or change the table.
- When having a table **Users** we can create a permission "**can edit users**" then only **users with** this **permission** can edit users in our table.



# Django Permissions

- Luckily for us we don't have to build this permission system from scratch, although you can.
- Django has a **built-in** permissions system we can make use of.
- We can use **permission** with our **models** to **limit access** to the data within that model's database table.

# Django Permissions

- Having “`django.contrib.auth`” within our **installed apps** we get four default permissions for each of our models.
  - Add
  - Change
  - Delete
  - View
- These **permissions** are **created** when calling:
  - `python manage.py migrate`

# Django Permissions

- Assuming you have an application with an app\_label **blog\_app** and a **model** named **Blog**, to test for basic permissions you should use:
  - **add:** `user.has_perm('blog_app.add_blog')`
  - **change:** `user.has_perm('blog_app.change_blog')`
  - **delete:** `user.has_perm('blog_app.delete_blog')`
  - **view:** `user.has_perm('blog_app.view_blog')`

# Custom Permissions

- We can create our own permissions as well.

```
from myapp.models import BlogPost
from django.contrib.auth.models import Permission
from django.contrib.contenttypes.models import ContentType

content_type = ContentType.objects.get_for_model(BlogPost)
permission = Permission.objects.create(
    codename="can_publish",
    name="Can Publish Posts",
    content_type=content_type,
)
```

# Adding Permissions

To add specific permissions to a user we first have to get the permissions.

```
content_type = ContentType.objects.get_for_model(BlogPost)
permission = Permission.objects.get(
    codename="can_publish",
    content_type=content_type,
)
```

# Adding Permissions

Then we can take the permission and add it to our user.

```
user.user_permissions.add(permission)
```

Finally we can check if the user has the correct permissions.

```
user.has_perm("myapp.change_blogpost")
```



# Permissions

We can restrict user from accessing certain views based on their permissions using the `permission_required()` decorator.

```
from django.contrib.auth.decorators import permission_required

@permission_required("polls.add_choice")
def my_view(request):
```

**SKILLS  
FOR LIFE**

**SKILLS BOOTCAMPS**



Department  
for Education

# CoGrammar

Groups

April 2024

# Groups

- Now that we have permissions we would like to give **multiple users** the **same** set of permissions.
- It would be very tedious to have to go to each individual user and add each permission.
- This is where groups come into play.
- We can create a **group** and **give** it a **set** of **permissions**.
- When we then want to **assign** those **permissions** to a **user** we can just **assign** them to the **group**

# Groups

- We saw that we had the option to **create** permission using **code** or we could make use of the Django **admin site**.
- The **same** applies for **groups** we can **create** and **assign** permissions to a group using **code** or we could do it through the **admin site**.
- The Group Model class has 2 attributes:
  - Name
  - Permissions

# Creating Groups

```
from django.contrib.auth.models import Group, Permission
from django.contrib.contenttypes.models import ContentType

from .models import BlogPost

new_group, created = Group.objects.get_or_create(name='publishers')

content_type = ContentType.objects.get_for_model(BlogPost)
permission = Permission.objects.get(
    codename="change_blogpost",
    content_type=content_type,
)
new_group.permissions.add(permission)
```

# Summary

- **Data security:** We have a responsibility to protect the data of our users. We have to limit access we provide users within our applications.
- **Permissions:** We can create and add permissions to our web application to limit users access to certain parts of the program as well as their access to the data contained within our program.
- **Groups:** We can use groups to apply a broad set of permissions to a user belonging to a specific group.



# Questions and Answers



# Thank you for attending



Department  
for Education

CoGrammar

