Digital Career Institute

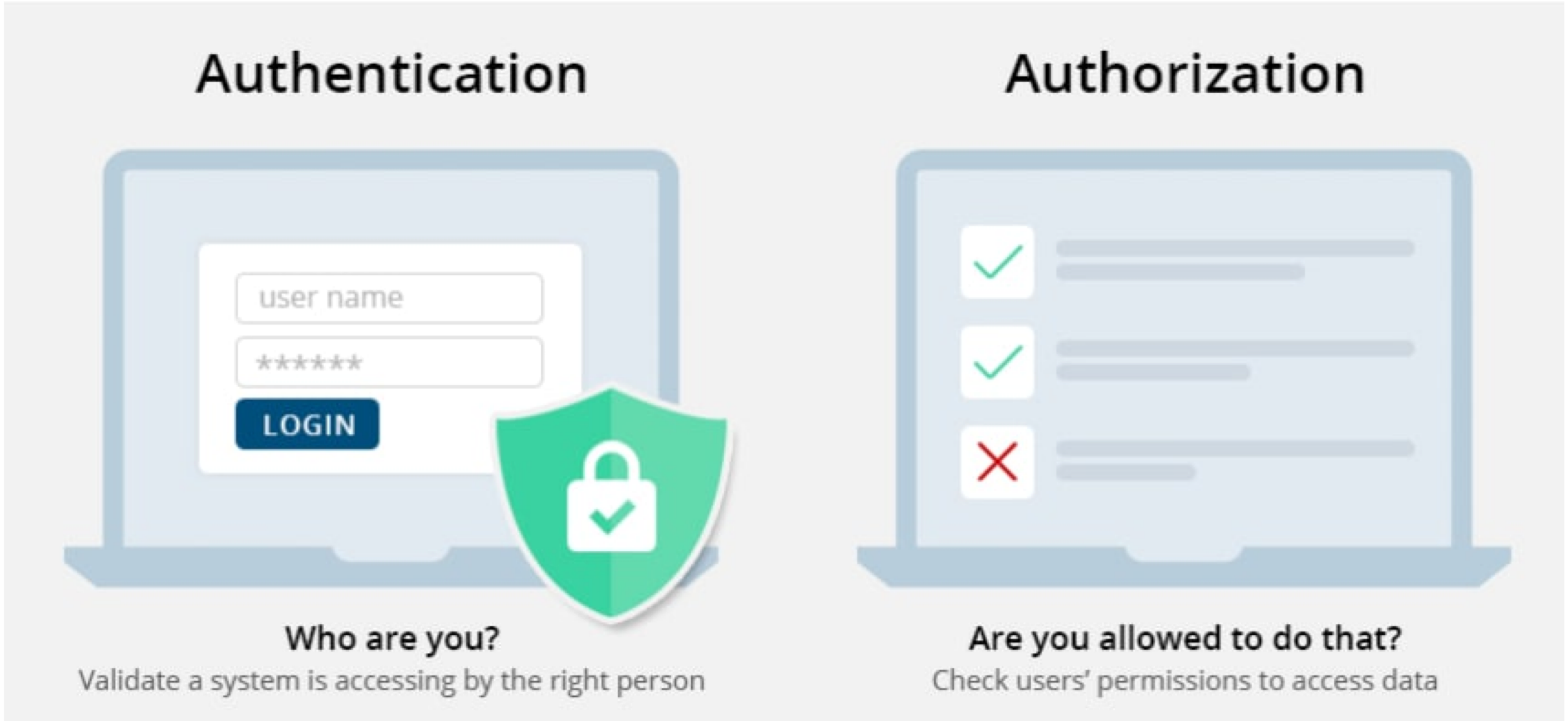
Create REST API

Digital Career Institute

Permission (Authorization) and Authentication

Digital Career Institute

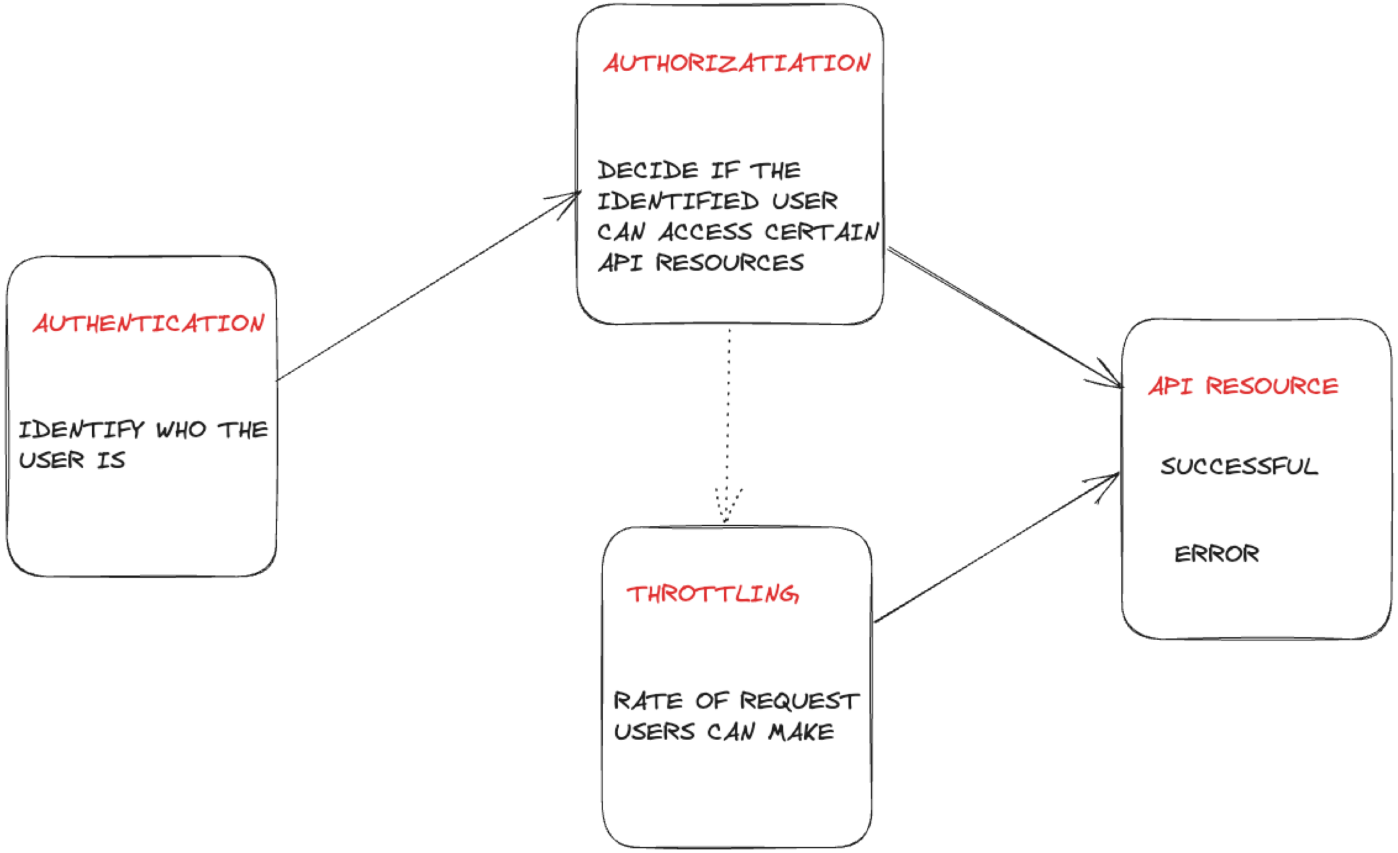
Permission & authentication


Authentication
Authorization
user name
******
LOGIN
Who are you?
Validate a system is accessing by the right person
Are you allowed to do that?
Check users' permissions to access data

Permission & authentication

- In **Django REST Framework (DRF)**, **authentication** refers to the process of verifying the identity of a user or client making a request to an **API**. It is the mechanism used to ensure that only authorized users or clients are allowed to access protected resources.
- **Permissions**, on the other hand, determine whether a user has the right to perform a certain action within an application. **Permissions** can be used to restrict access to specific pages or views within an application, and to restrict access to specific data or functionality within the application.

Permission & authentication


AUTHENTICATION
IDENTIFY WHO THE USER IS
AUTHORIZATIATION
DECIDE IF THE IDENTIFIED USER CAN ACCESS CERTAIN API RESOURCES
THROTTLING
RATE OF REQUEST USERS CAN MAKE
API RESOURCE
SUCCESSFUL
ERROR

Permission & authentication

- Pemission Checks are always run at the very start of the view, before any other code is allowed to proceed. It uses the authentication information in the **request.user** and **request.auth** properties to determine if the incoming request should be permitted
- The simplest style of permission would be to grant access to any authenticated user and deny unauthorized users (**IsAuthenticated** or **IsAuthenticatedOrReadOnly**), it can also go beyond by categorizing access based on different classes of users.

Permission & authentication

- In **DRF** authentications and permissions are needed for many reasons :
    - Security
    - Access control
    - Tracking and auditing
    - Compliance
    - Flexibility
- Permissions and authentications are set by specifying them in a list via our class-based views
    - permission_classes (list of builtin or customized)
    - authentication_classes (list of builtin or customized)

How Permissions Are Determined

- Permissions in Rest framework are always defined as a list of permission **builtin** or **customized** classes in the settings file (optionally in the views.py file)
- Before running the main body of the view, each permission in the list is checked. If any permission check fails, an appropriate exception is raised and the main code / body wont run.

Setting The Permission Policy

- The default permission policy is set globally using the **DEFAULT_PERMISSION_CLASSES** setting in the **REST_FRAMEWORK** setting

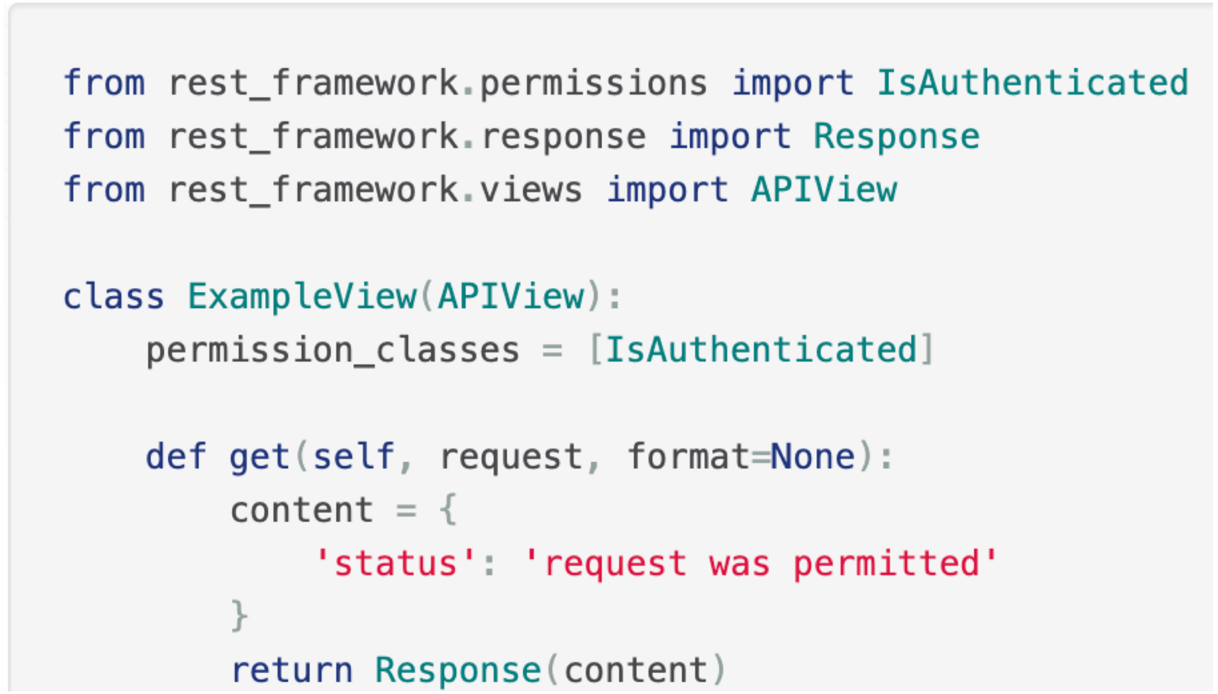
```
REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ]
}
```

- If not specified, the settings default to.

```
'DEFAULT_PERMISSION_CLASSES': [
   'rest_framework.permissions.AllowAny',
]
```

Setting The Permission Policy

- The permission policy can also be set per view



```python
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response
from rest_framework.views import APIView

class ExampleView(APIView):
    permission_classes = [IsAuthenticated]

    def get(self, request, format=None):
        content = {
            'status': 'request was permitted'
        }
        return Response(content)
```

Setting The Permission Policy

```python
from rest_framework.decorators import api_view, permission_classes
from rest_framework.permissions import IsAuthenticated
from rest_framework.response import Response

@api_view(['GET'])
@permission_classes([IsAuthenticated])
def example_view(request, format=None):
    content = {
        'status': 'request was permitted'
    }
    return Response(content)
```

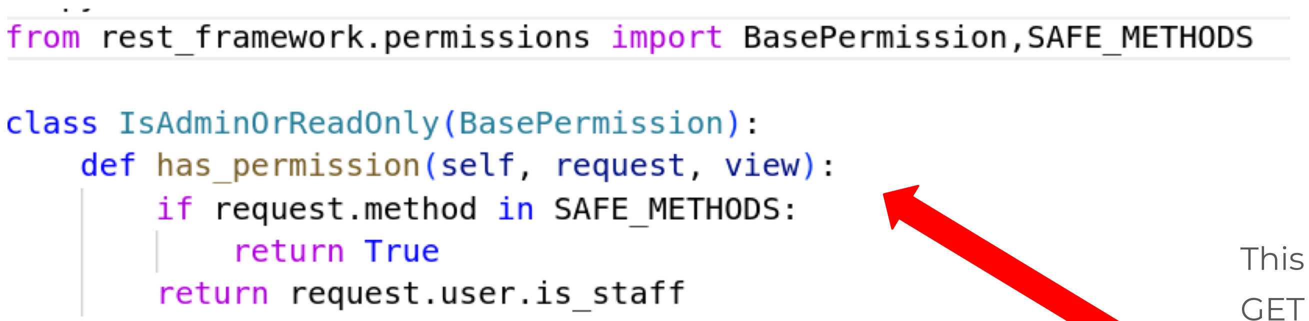
Built-in permissions

Permission Class	Description
AllowAny	Allows anyone to access the view, regardless of authentication status.
IsAuthenticated	Allows only authenticated users to access the view.
IsAdminUser	Allows only users with the "is_staff" flag set to True to access the view.
IsAuthenticatedOrReadOnly	Allows authenticated users to perform any action on the view, but allows unauthenticated users to only read the view.
DjangoModelPermissions	Uses the Django permission system to determine access control for the view based on the user's permissions.
DjangoObjectPermissions	Uses the Django permission system to determine access control for individual objects in the view.

Customised permissions

Digital Career Institute

- We can also create our own customised permission in our project

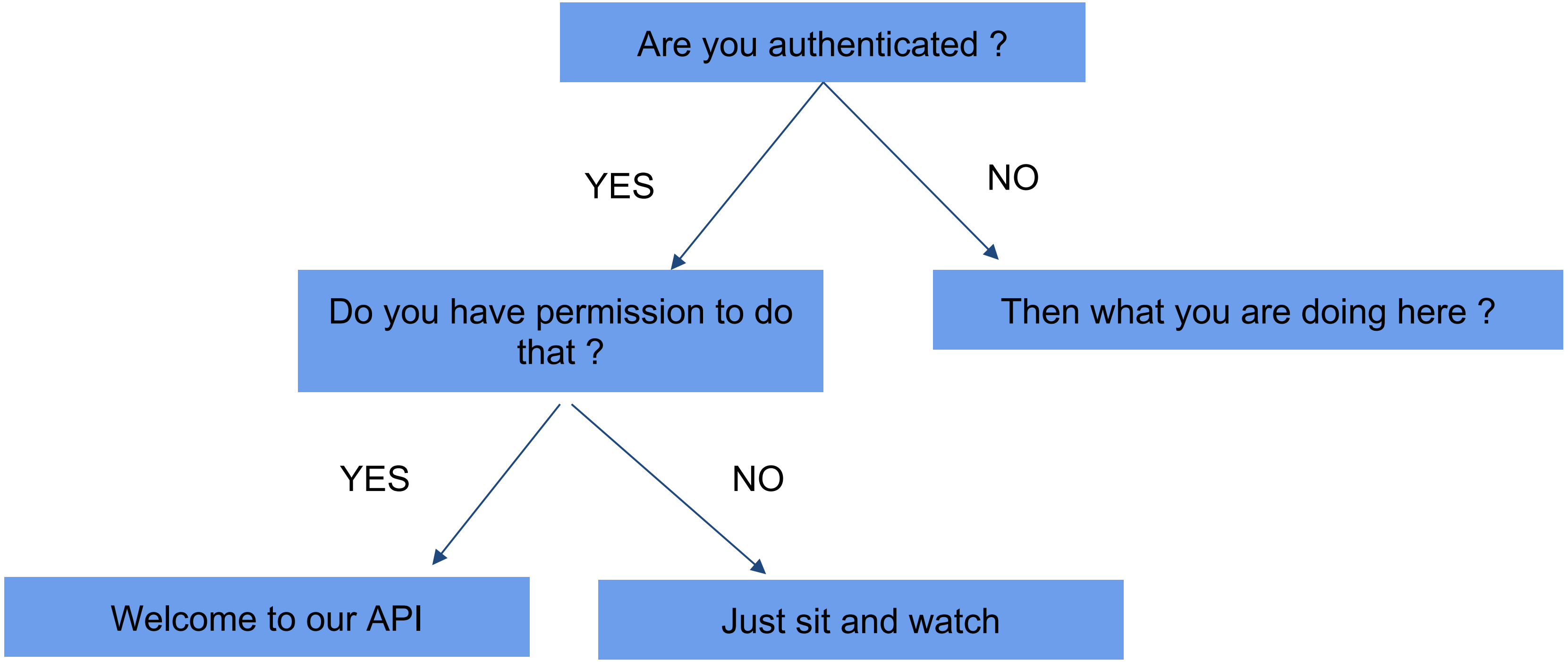


```python
from rest_framework.permissions import BasePermission,SAFE_METHODS

class IsAdminOrReadOnly(BasePermission):
    def has_permission(self, request, view):
        if request.method in SAFE_METHODS:
            return True
        return request.user.is_staff
```

This permission allows every to GET the APIs but only admin staff can POST,PUT and DELETE, by subclassing the **BasePermission** and implementing the **has_permission** method

To summarize



At the core of the lesson

Lessons learned:

- Connection between authentications and Authorization
- What Authorization Means in DRF and the flow for it
- Built-in permissions
- Custom permissions

References

https://www.django-rest-framework.org/api-guide/authentication/

https://www.django-rest-framework.org/api-guide/permissions/

THANK YOU

Contact Details
DCI Digital Career Institute gGmbH

Digital Career Institute
DCI