Django 8: Django REST Framework

IN608: Intermediate Application Development Concepts

Last Session’s Content

- Security in Django
  - XSS protection
  - CSRF protection
  - SQL injection protection
  - Clickjacking protection
  - Secret key

Today’s Content

- Django REST framework
  - Permissions
  - Serialization
  - Viewsets
  - Routers

Django REST Framework

Django REST Framework

- Install Django REST framework

```
# Windows
...\> pipenv install djangorestframework

# Linux or macOS
$ pipenv install djangorestframework
```

Django REST Framework

- View `Pipfile` & `Pipfile.lock`

```
[[source]]
name = "pypi"
url = "https://pypi.org/simple"
verify_ssl = true

[dev-packages]
selenium = "*"

[packages]
django = "*"
django-crispy-forms = "*"
djangorestframework = "*"

[requires]
python_version = "3.7"
```

Django REST Framework

- `mvt/settings.py`
- Resource: https://www.django-rest-framework.org/#installation

```
INSTALLED_APPS = [
    'polls.apps.PollsConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'crispy_forms',
    'rest_framework',
]
```

Permissions

- `mvt/settings.py`
- Default permission policy may be set globally using the `DEFAULT_PERMISSION_CLASSES` setting
- `IsAdminUser` permission class will deny permission to any user unless `user.is_staff` is `True`
  - Suitable if you want your API to only be accessible to trusted admins
- `IsAuthenticated` permission class will deny permission to any unauthenticated user
- Resource: https://www.django-rest-framework.org/api-guide/permissions

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

Serialization

- In the `polls/` directory, create a file called `serializers.py`
- App structure:

```
polls/
    __init__.py
    admin.py
    apps.py
    forms.py
    migrations/
        __init__.py
    models.py
    serializers.py
    static/
        polls/
            style.css
    templates/
        polls/
            base.html
            detail.html
            index.html
            results.html
    tests.py
    urls.py
    views.py
```

Serialization

- `polls/serializers.py`
- `ModelSerializer`
  - Automatically generates a set of fields based on the model
  - Automatically generates validators for the serializer
  - Includes default implementations of `.create()` & `.update()`
- Resource: https://www.django-rest-framework.org/api-guide/serializers

```
from rest_framework import serializers
from .models import User, Question, Choice

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username', 'email', 'first_name', 'last_name']

class QuestionSerializer(serializers.ModelSerializer):
    class Meta:
        model = Question
        fields = ['id', 'question_text', 'pub_date']

class ChoiceSerializer(serializers.ModelSerializer):
    class Meta:
        model = Choice
        fields = ['id', 'question', 'choice_text', 'votes']
```

Viewsets

- `polls/views.py`
- `from rest_framework import viewsets`
- `from .serializers import UserSerializer, QuestionSerializer, ChoiceSerializer`
- `ModelViewSet` - provides all actions
- `ReadOnlyModelViewSet` - only provides read-only actions, i.e, `.list()` & `.retrieve()`
- Resource: https://www.django-rest-framework.org/api-guide/viewsets

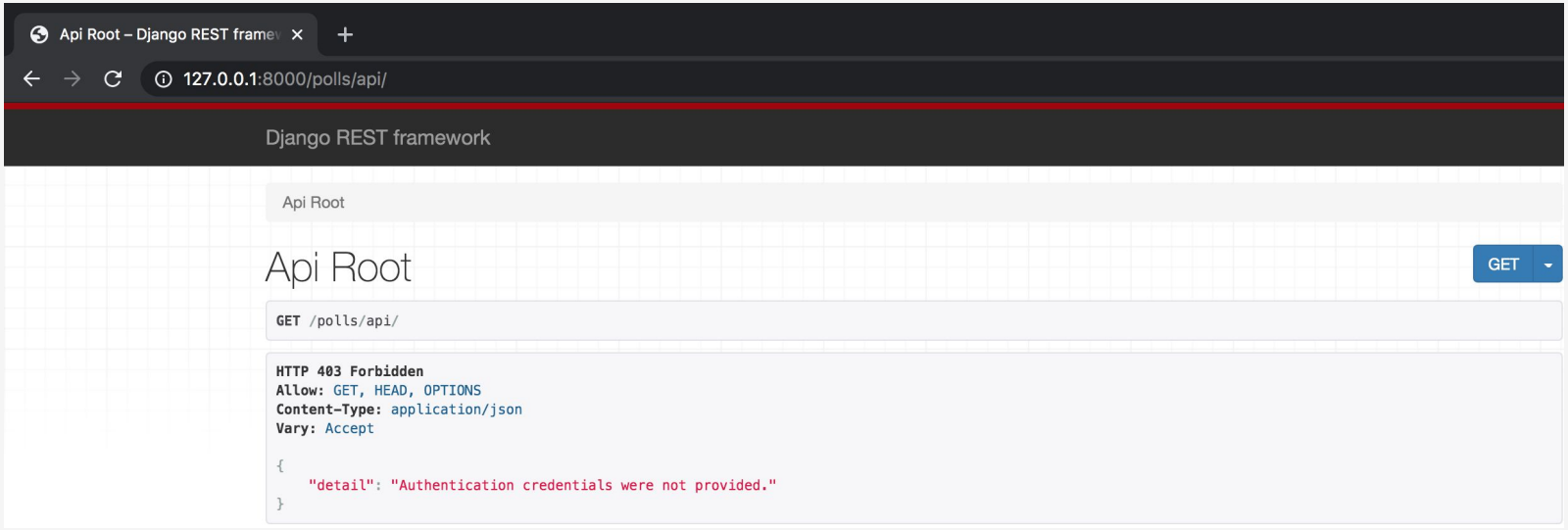
```
class UserViewSet(viewsets.ReadOnlyModelViewSet):
    queryset = User.objects.all()
    serializer_class = UserSerializer

class QuestionViewSet(viewsets.ModelViewSet):
    queryset = Question.objects.all()
    serializer_class = QuestionSerializer

class ChoiceViewSet(viewsets.ModelViewSet):
    queryset = Choice.objects.all()
    serializer_class = ChoiceSerializer
```

Routers

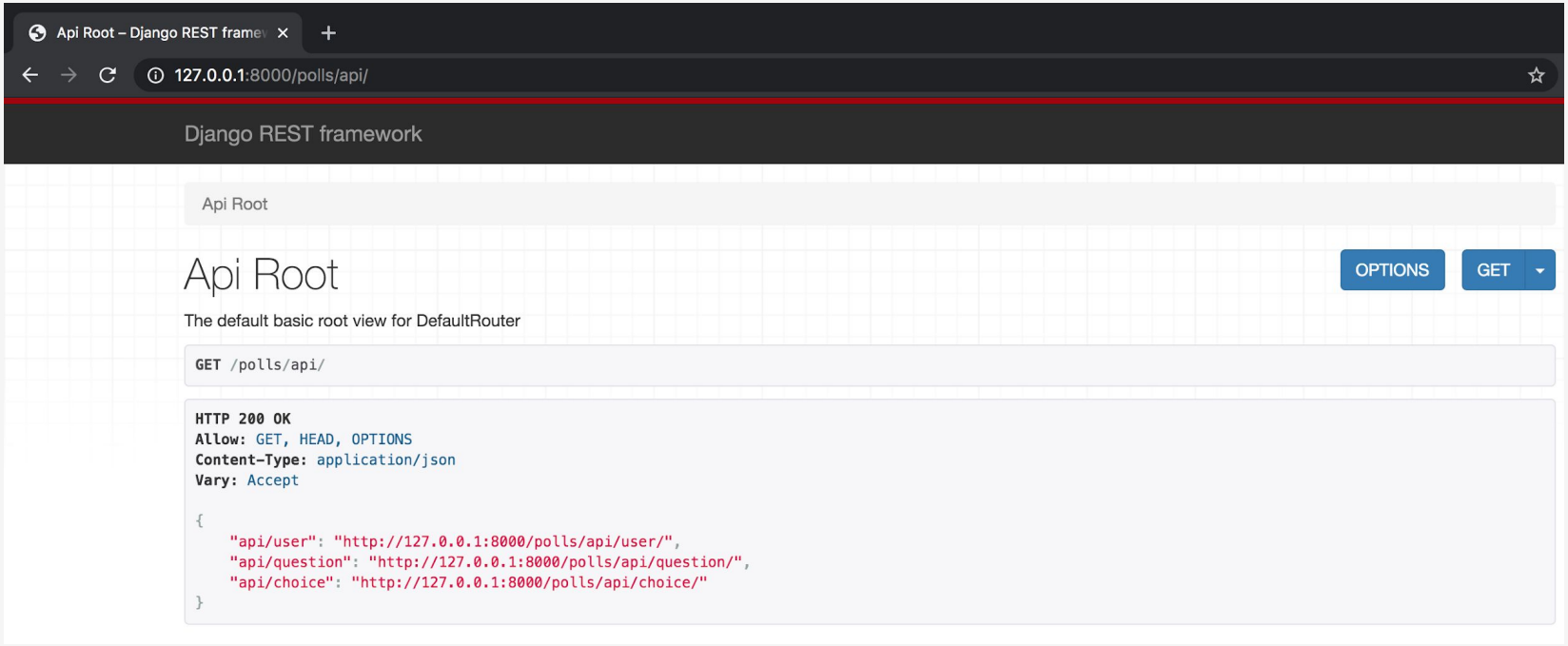
- `polls/urls.py`
- `DefaultRouter()` - returns a response containing hyperlinks to all the views
- `register()` - prefix & viewset
- Resource: https://www.django-rest-framework.org/#installation

```
from django.contrib.auth import views as auth_views
from django.urls import include, path
from rest_framework import routers
from . import views

app_name = 'polls'

router = routers.DefaultRouter()
router.register(r'api/user', views.UserViewSet)
router.register(r'api/question', views.QuestionViewSet)
router.register(r'api/choice', views.ChoiceViewSet)

urlpatterns = [
    path('api/', include(router.urls), name='api'),
    path('accounts/signup/', views.SignupView.as_view(), name='signup'),
    path('accounts/login/', auth_views.LoginView.as_view(), name='login'),
    path('accounts/logout/', auth_views.LogoutView.as_view(), name='logout'),
    path('', views.IndexView.as_view(), name='index'), # /polls/
    path('<int:pk>/', views.DetailView.as_view(), name='detail'), # /polls/2/
    path('<int:pk>/results/', views.ResultsView.as_view(), name='results'), # /polls/2/results/
    path('<int:question_id>/vote/', views.vote, name='vote'), # /polls/2/vote/
]

urlpatterns += router.urls
```

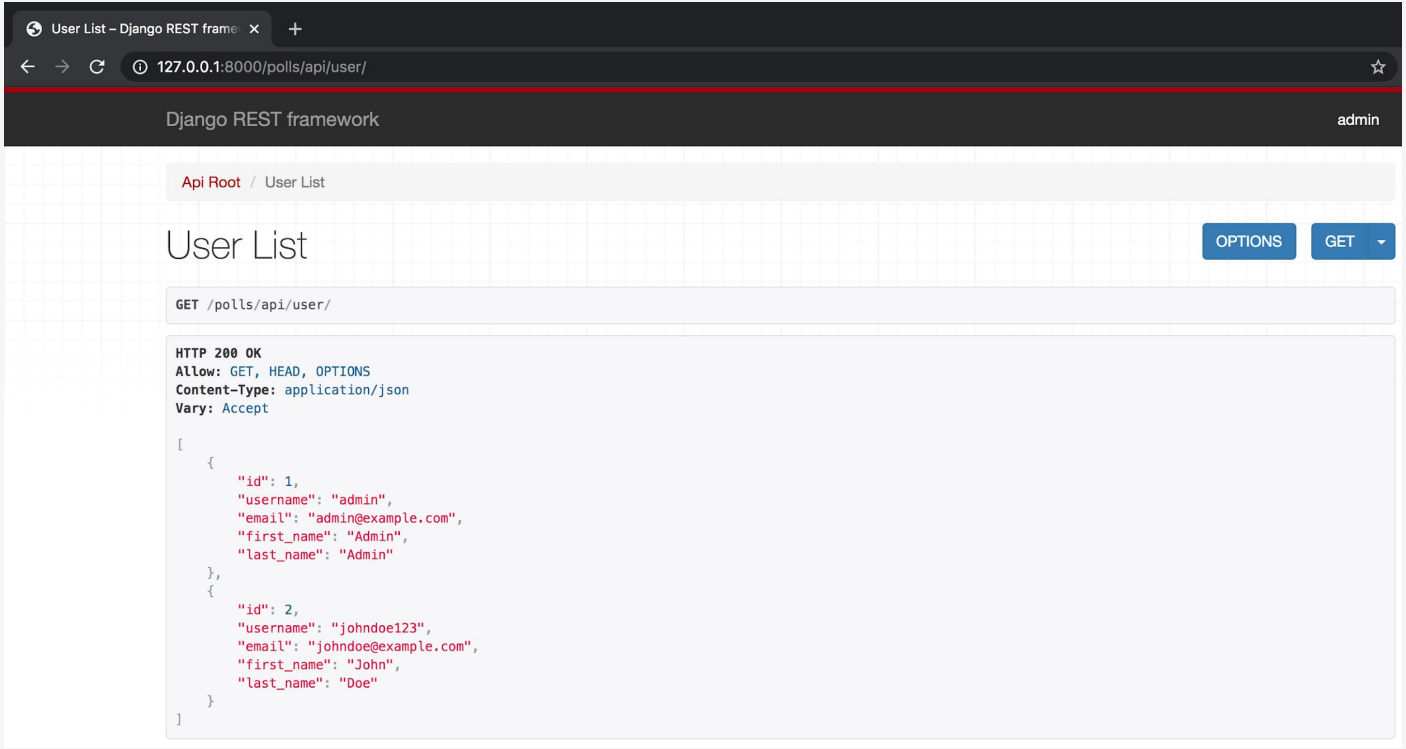
Django REST Framework



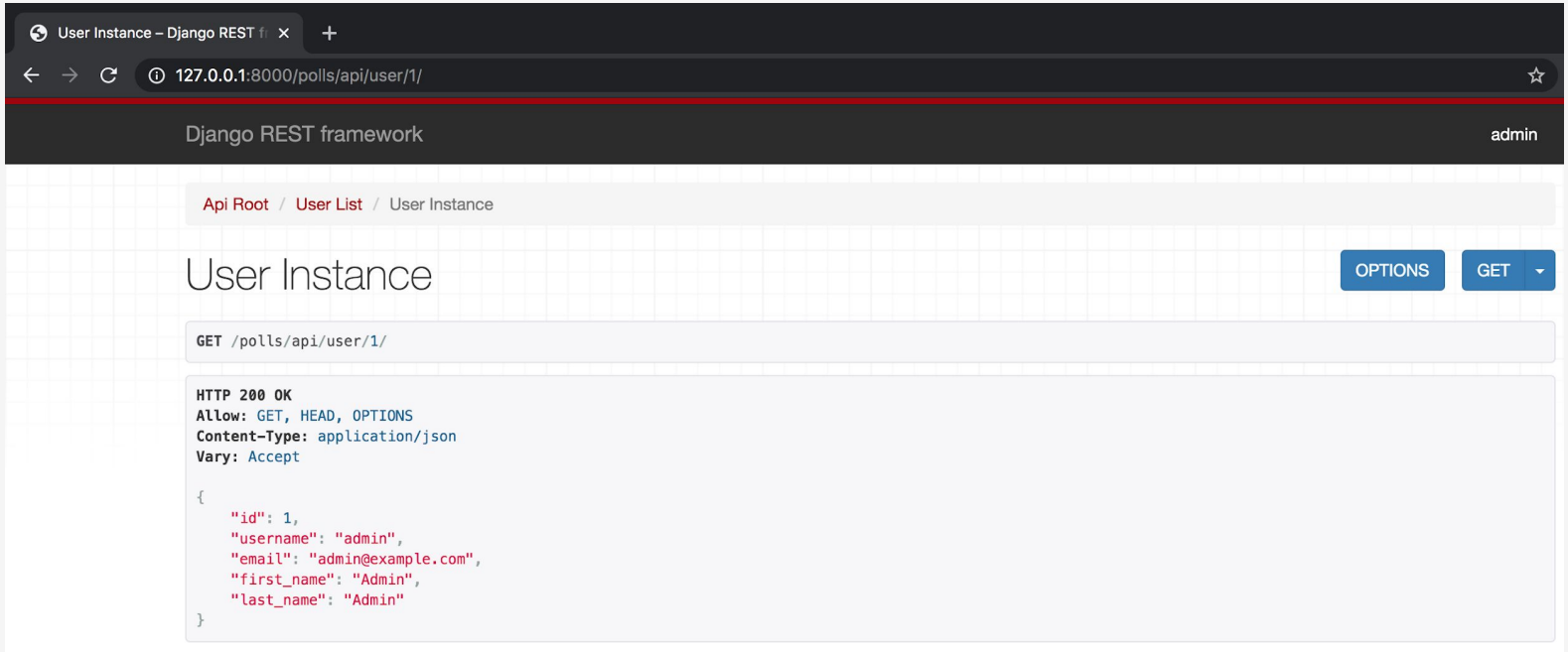
Django REST Framework



Django REST Framework

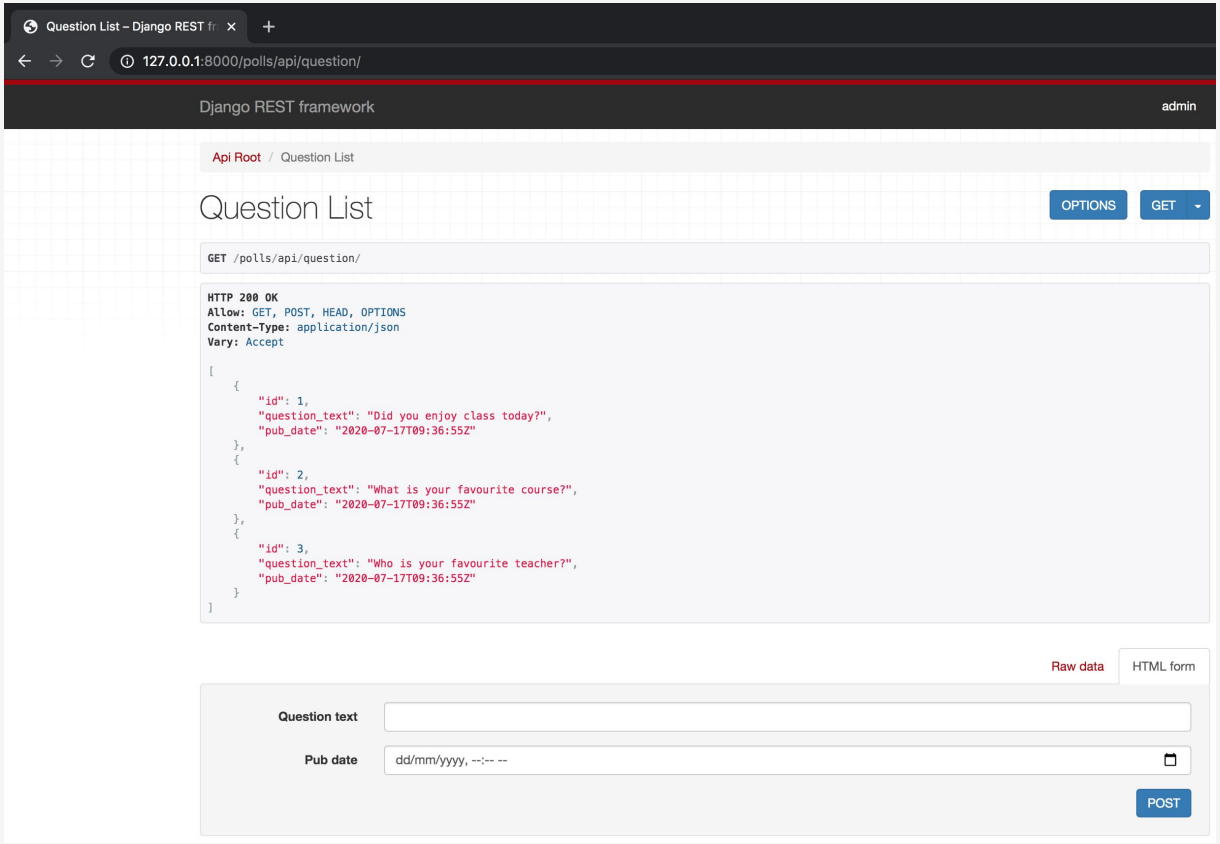


Django REST Framework



Django REST Framework

Django REST Framework



Programming Activity

- Checkout to master - `git checkout master`
- Create a new branch called 14-practical - `git checkout -b 14-practical`
- Open the file `14-practical.pdf` and work on the tasks described