College of Engineering, Construction and Living Sciences
Bachelor of Information Technology
IN608: Intermediate Application Development Concepts
Level 6, Credits 15

Practical 21 React 6: Forms

Due Date: 27/10/2020 at 5pm

In this practical, you will complete a series of tasks covering today's lecture. This practical is worth 1% of the final mark for the IN608: Intermediate Application Development Concepts course.

Before you start, in your practicals repository, create a new branch called **21-practical**.

Task 1

Create a Django project called `backend`. `cd` to `backend`, create a virtual environment & install **Django**. Create an app called `practical21backend`. Please ensure you configure your app in `backend/settings.py`.

```
# backend/settings.py

INSTALLED_APPS = [
    'practical21backend',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

Create a model to define how the `Todo` items should be stored in the database, open the `practical21backend/models.py` file & update it with the following:

```
# practical21backend/models.py

from django.db import models
```

```
class Todo(models.Model):
    title = models.CharField(max_length=120)
    description = models.TextField()
    completed = models.BooleanField(default=False)

def __str__(self):
    return self.title
```

We have created a `Todo` model, we need to create a migration file & apply the changes to the database, so let's run these commands:

```
python manage.py makemigrations
python manage.py migrate
```

Install the `djangorestframework` & `django-cors-headers` using `Pipenv`:

```
pipenv install djangorestframework django-cors-headers
```

We need to add `rest_framework` & `corsheaders` to the list of installed applications, so update the `INSTALLED_APPS` & `MIDDLEWARE` sections in `backend/settings.py`:

```
# backend/settings.py

INSTALLED_APPS = [
    'practical21backend',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'corsheaders',
    'rest_framework',
]

MIDDLEWARE = [
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

CORS_ORIGIN_WHITELIST = [
  'http://localhost:3000'
]
```

`django-cors-headers` is a python library that will help in preventing the errors that we would normally get due to `CORS`. rules. In the `CORS_ORIGIN_WHITELIST` snippet, we whitelisted `http://localhost:3000/` because we want the React application to interact with the Djnago REST API.

Create a new `practical21backend/serializers.py` file & update it with the following code:

```
from rest_framework import serializers
from .models import Todo

class TodoSerializer(serializers.ModelSerializer):
    class Meta:
        model = Todo
        fields = ['id', 'title', 'description', 'completed']
```

In the code snippet above, we specified the model to work with & the fields we want to be converted to **JSON**.

Open `practical21backend/views.py` file & update it with the following code:

```
from django.shortcuts import render
from rest_framework import viewsets
from .serializers import TodoSerializer
from .models import Todo

class TodoView(viewsets.ModelViewSet):
    serializer_class = TodoSerializer
    queryset = Todo.objects.all()
```

The `viewsets` base class provides the implementation for **CRUD** operations by default, what we had to do was specify the serializer class & the `queryset`.

Open `backend/urls.py` file & update it with the following code:

```
from django.contrib import admin
from django.urls import path, include
from rest_framework import routers
from practical21backend import views

router = routers.DefaultRouter()
router.register(r'todos', views.TodoView, 'todo')

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include(router.urls))
]
```

This is the final step that completes the building of the **API**, we can now perform **CRUD** operations on the `Todo` model.

Task 2

Create a new **React** application called `frontend`.

```
npx create-react-app frontend
```

Install the following packages:

```
npm install bootstrap reactstrap axios
```

In `src/index.js`, declare `import 'bootstrap/dist/css/bootstrap.min.css'`. This will allow you to use **Bootstrap** in your application.

Open the `src/index.css` file & replace the styles with the following:

```
body {
  margin: 0;
  padding: 0;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
    "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

.todo-title {
  cursor: pointer;
}

.completed-todo {
  text-decoration: line-through;
}

.tab-list > span {
  padding: 5px 8px;
  border: 1px solid #000000;
  margin-right: 5px;
  cursor: pointer;
}

.tab-list > span.active {
  background-color: #000000;
  color: #ffffff;
}
```

Create a new directory called `components`. In the `components` directory, copy `Modal.js` which is given to you in the `21-react-6-forms` directory. Replace `App.js` with the `App.js` file given to you in the same directory. Comments have been provided in these two files.

In package.json, add `"proxy": "http://localhost:8000"` above the `"dependencies"`. The proxy will help in tunnelling API requests to `http://localhost:8000` where the `Django` application will handle them.

Once you have done this, run both applications:

```
python manage.py runserver
npm start
```

Task 3

- Django application
  - Create five unit tests covering the `Todo` model, `TodoSerializer` serializer & API
- React application
  - Convert the **CSS** in `src/index.css` to **Sass**
  - Ensure the `title` & `description` form inputs are required
  - Display an appropriate message, if there are no `Todo` items in the list

Task 4

Please answer the following question in the comment section once you make a pull request:

- Describe three reasons why an application's frontend & backend should be separated.

Resources

- CORS