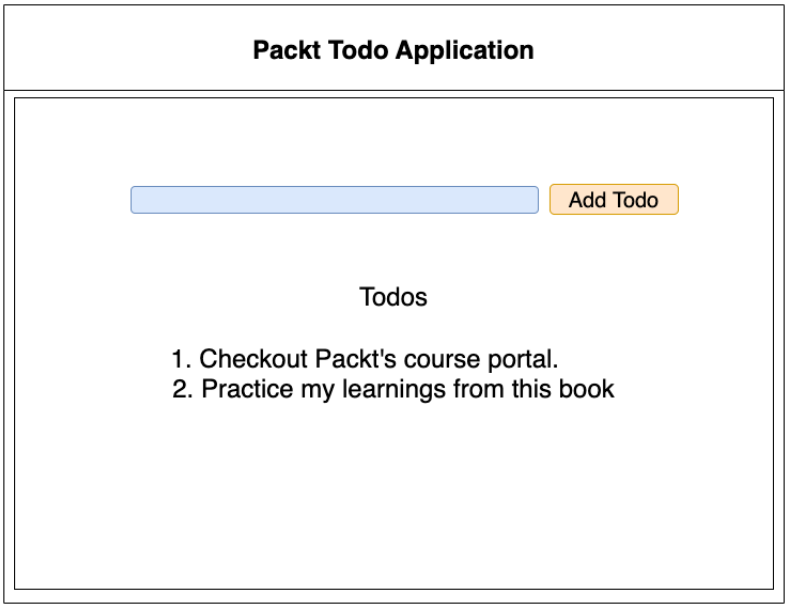


Figure 4.1 – Mockup of our homepage template

1. First, let's install the Jinja package and create the `templates` folder:

```
(venv)$ pip install jinja2
(venv)$ mkdir templates
```

2. In the newly created folder, create two new files, `home.html` and `todo.html`:

```
(venv)$ cd templates
(venv)$ touch {home,todo}.html
```

   In the preceding command block, we have created two template files:

   - `home.html` for the application's home page
   - `todo.html` for the todo page

In the mockup in *Figure 4.1*, the inner box denotes the todo template while the bigger box is the homepage template.

Before moving on to build our templates, let's configure Jinja in our FastAPI application:

1. Let's modify the `POST` route of the todo API component, `todo.py`:

```
from fastapi import APIRouter, Path, HTTPException,
status, Request, Depends
from fastapi.templating import Jinja2Templates
from model import Todo, TodoItem, TodoItems

todo_router = APIRouter()

todo_list = []

templates = Jinja2Templates(directory="templates/")

@todo_router.post("/todo")
async def add_todo(request: Request, todo: Todo =
Depends(Todo.as_form)):
    todo.id = len(todo_list) + 1
    todo_list.append(todo)
    return templates.TemplateResponse("todo.html",
    {
        "request": request,
        "todos": todo_list
    })
```

2. Next, update the `GET` routes:

```
@todo_router.get("/todo", response_model=TodoItems)
async def retrieve_todo(request: Request):
    return templates.TemplateResponse("todo.html", {
        "request": request,
        "todos": todo_list
    })

@todo_router.get("/todo/{todo_id}")
async def get_single_todo(request: Request, todo_id: int
= Path(..., title="The ID of the todo to retrieve.")):
```

```
    for todo in todo_list:
        if todo.id == todo_id:
            return templates.TemplateResponse(
            "todo.html", {
                "request": request,
                "todo": todo
            })
    raise HTTPException(
        status_code=status.HTTP_404_NOT_FOUND,
        detail="Todo with supplied ID doesn't exist",
    )
```

In the preceding code block, we have configured Jinja to look into the `templates` directory to serve the templates passed to the `templates.TemplateResponse()` method.

The `POST` method for adding a todo has also been updated to include a dependency on the input passed. Dependencies will be discussed in detail in *Chapter 6*, *Connecting to a Database.*

3. In `model.py`, add the highlighted code before the `Config` subclass:

```
from typing import List, Optional

class Todo(BaseModel):
    id: Optional[int]
    item: str

    @classmethod
    def as_form(
        cls,
        item: str = Form(...)
    ):
        return cls(item=item)
```

Now that we have updated our API code, let's write our templates. We'll start by writing the base `home.html` template in the next step.

4. In `home.html`, we'll start by declaring the document type:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible"
         content="IE=edge">
        <meta name="viewport" content="width=device-
         width, initial-scale=1.0">
        <title>Packt Todo Application</title>
        <link rel="stylesheet" href=
         "https://stackpath.bootstrapcdn.com/
         bootstrap/4.1.0/css/bootstrap.min.css"
         integrity="sha384-9gVQ4dYFwwWSjIDZ
         nLEWnxCjeSWFphJiwGPXr1jddIhOegi
         u1FwO5qRGvFXOdJZ4" crossorigin="anonymous">
        <link rel="stylesheet" href=
         "https://use.fontawesome.com/releases
         /v5.0.10/css/all.css" integrity="sha384-
         +d0P83n9kaQMCwj8F4RJB66tzIwOKmrdb46+porD/
         OvrJ+37WqIM7UoBtwHO6Nlg" crossorigin=
         "anonymous">
    </head>
```

5. The next step is to write the content for the template body. In the template's body, we'll include the name of the application under a `<header></header>` tag, and a link to the child template's `todo_container` wrapped in a block tag. The child template will be written in *step 8*.

   Include the following code just after the `</head>` tag in the `home.html` template file:

```
    <body>
        <header>
            <nav class="navar">
                <div class="container-fluid">
                    <center>
```

```
                        <h1>Packt Todo
                        Application</h1>
                    </center>
                </div>
            </nav>
        </header>
        <div class="container-fluid">
            {% block todo_container %}{% endblock %}
        </div>
    </body>
</html>
</html>
```

The highlighted code tells the parent template that the `todo_container` block will be defined by a child template. A child template containing the `todo_container` block and extending the parent template will have its content displayed there.

6. To see the changes, activate your virtual environment and start your application:

```
$ source venv/bin/activate
(venv)$ uvicorn api:app --host=0.0.0.0 --port 8000
--reload
```

7. Open `http://127.0.0.1:8000/todo` to preview the changes:

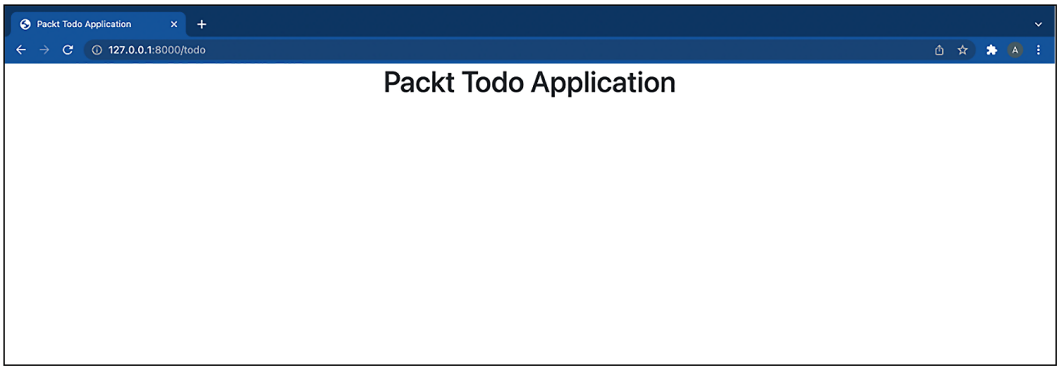


Figure 4.2 – Todo application homepage