8. Next, let's write the todo template in `todo.html`:

```
{% extends "home.html" %}

{% block todo_container %}
<main class="container">
    <hr>
    <section class="container-fluid">
        <form method="post">
            <div class="col-auto">
                <div class="input-group mb-3">
                    <input type="text" name="item"
                     value="{{ item }}" class="form-
                     control" placeholder="Purchase
                     Packt's Python workshop course"
                     aria-label="Add a todo"
                     aria-describedby="button-addon2" />
                    <button class="btn btn-outline-
                     primary" type="submit" id=
                     "button-addon2" data-mdb-ripple-
                     color="dark">
                        Add Todo
                    </button>
                </div>
            </div>
        </form>
    </section>
    {% if todo %}
        <article class="card container-fluid">
            <br/>
            <h4>Todo ID: {{ todo.id }} </h4>
            <p>
                <strong>
                    Item: {{ todo.item  }}
                </strong>
```

```
            </p>
        </article>
    {% else %}
        <section class="container-fluid">
            <h2 align="center">Todos</h2>
            <br>
            <div class="card">
                <ul class="list-group list-group-
                 flush">
                    {% for todo in todos %}
                        <li class="list-group-item">
                        {{ loop.index }}. <a href=
                        "/todo/{{loop.index}}"> {{
                        todo.item }} </a>
                    </li>
                {% endfor %}
            </ul>
        </div>
    {% endif %}
</section>
</main>
{% endblock %}
```

In the previous code block, the todo template is inheriting the homepage template. We also defined the `todo_container` block, whose content will be displayed in the parent template.

The todo template is used by both routes for retrieving all the todos and for a single todo. As a result, the template renders different content depending on the route used.

In the template, Jinja checks to see whether a todo variable is passed using the `{% if todo %}` block. The todo detail is rendered if a todo variable is passed, otherwise, it renders the content in the `{% else %}` block, which is the list of todos.

9. Refresh the web browser to view the recent changes:

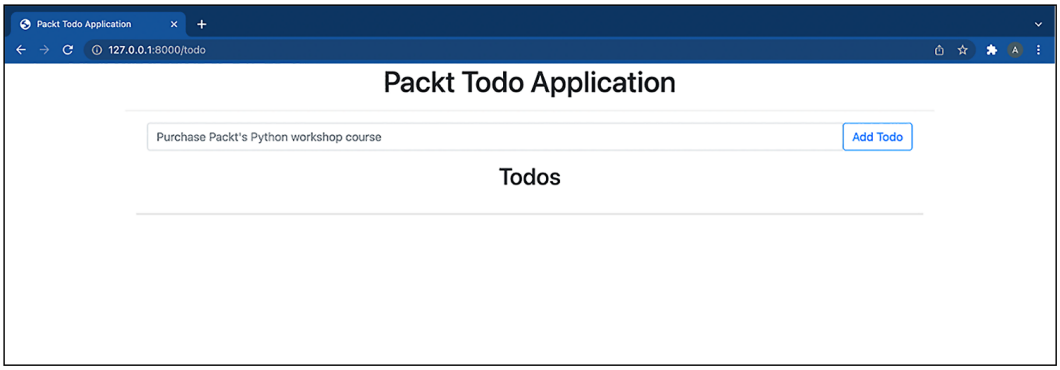

Figure 4.3 – Update todo homepage

10. Let's add a todo to verify that the homepage works as expected:

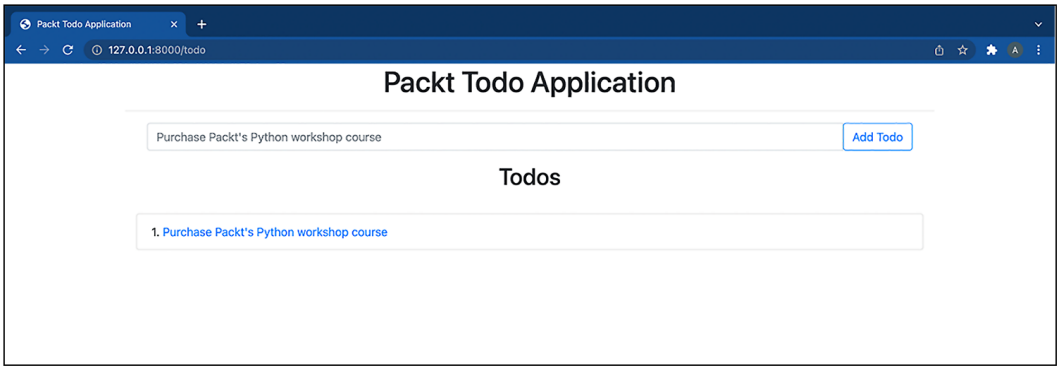

Figure 4.4 – List of todos displayed

11. The todo is clickable. Click on the todo and you should see the following page:

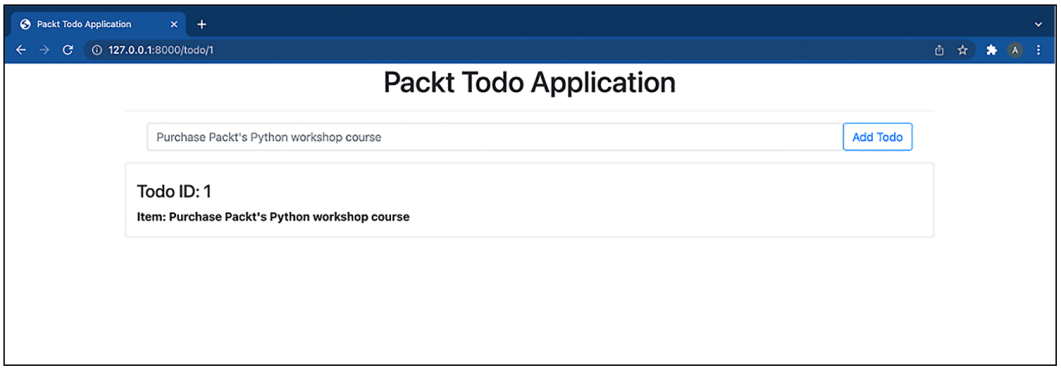

Figure 4.5 – Single todo page

We have successfully added a template to our FastAPI application.

Summary

In this chapter, we learned what templating is, the basics of the Jinja templating system, and how to use it in FastAPI. We made use of the basics learned in the first section of this chapter to decide what content to render. We also learned what template inheritance is and how it works using the homepage and todo templates as examples.

In the next chapter, you will be introduced to structuring applications in FastAPI. In this chapter, you will be building a planner application using the knowledge from this and earlier chapters. You will first be introduced to how applications are structured before proceeding to build the planner application.