# **Django Prepare Template**

Create Template

After creating Models, with the fields and data we want in them, it is time to display the data in a web page.

Start by creating an HTML file named all\_members.html and place it in the /templates/ folder:

my\_tennis\_club/members/templates/all\_members.html:

<!DOCTYPE html>

<html>

<body>

<h1>Members</h1>

<ul>

{% for x in mymembers %}

<li>{{ x.firstname }} {{ x.lastname }}</li>

{% endfor %}

</ul>

</body>

</html>

Do you see the {% %} brackets inside the HTML document?

They are Django Tags, telling Django to perform some programming logic inside these brackets.

You will learn more about Django Tags in our [Django Tags chapter](https://www.w3schools.com/django/django_template_tags.php).

Modify View

Next we need to make the model data available in the template. This is done in the view.

In the view we have to import the Member model, and send it to the template like this:

my\_tennis\_club/members/views.py:

from django.http import HttpResponse

from django.template import loader

from .models import Member

def members(request):

mymembers = Member.objects.all().values()

template = loader.get\_template('all\_members.html')

context = {

'mymembers': mymembers,

}

return HttpResponse(template.render(context, request))

[Run Example »](https://www.w3schools.com/django/showdjango.php?filename=demo_prepare)

The members view does the following:

* Creates a mymembers object with all the values of the Member model.
* Loads the all\_members.html template.
* Creates an object containing the mymembers object.
* Sends the object to the template.
* Outputs the HTML that is rendered by the template.

The Result

We have created an example so that you can see the result:

[Run Example »](https://www.w3schools.com/django/showdjango.php?filename=demo_prepare)

If you have followed all the steps on your own computer, you can see the result in your own browser:

Start the server by navigating to the /my\_tennis\_club/ folder and execute this command:

py manage.py runserver

In the browser window, type [127.0.0.1:8000/members/](http://127.0.0.1:8000/members/) in the address bar.

## Details Template

The next step in our web page will be to add a Details page, where we can list more details about a specific member.

Start by creating a new template called details.html:

my\_tennis\_club/members/templates/details.html:

<!DOCTYPE html>

<html>

<body>

<h1>{{ mymember.firstname }} {{ mymember.lastname }}</h1>

<p>Phone: {{ mymember.phone }}</p>

<p>Member since: {{ mymember.joined\_date }}</p>

<p>Back to <a href="/members">Members</a></p>

</body>

</html>

## Add Link in all-members Template

The list in all\_members.html should be clickable, and take you to the details page with the ID of the member you clicked on:

my\_tennis\_club/members/templates/all\_members.html:

<!DOCTYPE html>

<html>

<body>

<h1>Members</h1>

<ul>

{% for x in mymembers %}

<li><a href="details/{{ x.id }}">{{ x.firstname }} {{ x.lastname }}</a></li>

{% endfor %}

</ul>

</body>

</html>

## Create new View

Then create a new view in the views.py file, that will deal with incoming requests to the /details/ url:

my\_tennis\_club/members/views.py:

from django.http import HttpResponse

from django.template import loader

from .models import Member

def members(request):

mymembers = Member.objects.all().values()

template = loader.get\_template('all\_members.html')

context = {

'mymembers': mymembers,

}

return HttpResponse(template.render(context, request))

def details(request, id):

mymember = Member.objects.get(id=id)

template = loader.get\_template('details.html')

context = {

'mymember': mymember,

}

return HttpResponse(template.render(context, request))

The details view does the following:

* Gets the id as an argument.
* Uses the id to locate the correct record in the Member table.
* loads the details.html template.
* Creates an object containing the member.
* Sends the object to the template.
* Outputs the HTML that is rendered by the template.

## Add URLs

Now we need to make sure that the /details/ url points to the correct view, with id as a parameter.

Open the urls.py file and add the details view to the urlpatterns list:

my\_tennis\_club/members/urls.py:

from django.urls import path

from . import views

urlpatterns = [

path('members/', views.members, name='members'),

path('members/details/<int:id>', views.details, name='details'),

]

[Run Example »](https://www.w3schools.com/django/showdjango.php?filename=demo_add_link_details)

If you have followed all the steps on your own computer, you can see the result in your own browser: [127.0.0.1:8000/members/](http://127.0.0.1:8000/members/).

If the server is down, you have to start it again with the runserver command:

py manage.py runserver