Module 4 Cheat Sheet: Consolidate and Deploy Your Django App

Package/ Method	Description	Code Example

ListView:	Displays a list of objects.	**class MyListView(ListView):
		  model = MyModel
		
		  template_name = 'my_template.html'
		
		  context_object_name = 'object_list' # default: object_list**

DetailView	Displays details of a single object.	**class MyDetailView(DetailView):
		  model = MyModel
		
		  template_name = 'my_template.html'
		  context_object_name = 'object' # default: object
		  pk_url_kwarg = 'my_model_id' # default: pk**

CreateView	Displays a form to create a new object.	**class MyCreateView(CreateView):
		  model = MyModel
		
		  template_name = 'my_template.html'
		  fields = '__all__' # or specify a list of fields**

UpdateView	Displays a form to update an existing object.	**class MyUpdateView(UpdateView):
		  model = MyModel
		
		  template_name = 'my_template.html'
		  fields = '__all__' # or specify a list of fields
		  pk_url_kwarg = 'my_model_id' # default: pk**

DeleteView	Displays a confirmation page to delete an object.	**class MyDeleteView(DeleteView):
		  model = MyModel
		
		  template_name = 'my_template.html'
		  success_url = '/success-url/'
		  pk_url_kwarg = 'my_model_id' # default: pk**

Basic View Function	Function-based view that returns “Hello, World!”
	
	From Django.http import HttpResponse
		**def my_view(request):
		  # Your view logic here
		  return HttpResponse("Hello, World!")**

Render a Template	Function-based view to render a template with context.
	
	From django.shortcuts import render
		**def my_template_view(request):
		  context = {'variable': value}
		
		  return render(request, 'my_template.html', context)**

Redirect to a URL	Function-based view to redirect to a specific URL. From django.shortcuts import redirect	**def my_redirect_view(request):** ** return redirect('url_name_or_path')**
Handle a Form Submission	Function-based view to handle form submission. From django.shortcuts import render	**def my_form_view(request):** ** if request.method == 'POST':** ** # Process the form data here** ** else:** ** # Display the form** ** return render(request, 'my_form_template.html', context)**
Handle URL Parameters	Function-based view that accesses URL parameters.	**def my_param_view(request, param):** ** # Access the 'param' value from the URL**
Protecting Views (Restrict Access) using @login_required Decorator	Function-based view protected with login_required decorator. From django.contrib.auth.decorators import login_required	**@login_required** **def my_protected_view(request):** ** # Your view logic here**

		Add the following link to the <head> section of your base template (usually base.html):
Bootstrap CSS	Link to include Bootstrap CSS in the base template.	**<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">**
		Include the Bootstrap JavaScript library at the end of the <body> section to enable certain features (for example, dropdowns, modals):
Bootstrap JavaScript	Script tag to include Bootstrap JavaScript library.	**<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.min.js"></script>**
Bootstrap classes and components	Create visually appealing and responsive web pages without having to write CSS styles manually.	**Click Me**

Configuration – Static files	Django settings for static files configuration.	**In your Django settings (settings.py), define the following settings:** **STATIC_URL = '/static/studio/edx.org-next/' # URL to access static files** **STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')] # Directory to look for static files**
Configuration – Installed apps	Defines a list of all the applications installed in the project.	**Add 'django.contrib.staticfiles' to your INSTALLED_APPS in settings.py:** **INSTALLED_APPS = [** **# ...** **'django.contrib.staticfiles',** **# ...** **]**

Configuration – App Dirs	A configuration option used within the TEMPLATES setting. When set to TRUE, Django will look for template files within the app directories.	**Make sure the APP_DIRS setting is set to True in the TEMPLATES list. This allows Django to look for static files within the apps' directories.** **TEMPLATES = [** **{** **# ...** **'APP_DIRS': True,** **# ...** **},** **]**
Usage – Static content	Code to style the HTML templates and provide interactivity to web pages.	**<link href="{% static 'your_app/css/style.css' %}" rel="stylesheet">** **<script src="{% static 'your_app/js/script.js' %}"></script>** **<img src="{% static 'your_app/img/logo.png' %}" alt="Logo">**
Collecting static files	When deploying your project, you need to collect all static files into a single location.	**python manage.py collectstatic** **STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')**