# Bootcamp Training

## Django

Authorized & published by Summitworks Technologies Inc

SummitWorks™
GLOBAL SOLUTION ARCHITECTS

- **Views**

# Introduction

# Use of Context object in views and templates

**If you do not provide "context_object_name", your view may look like this:**

```
<ul>
   {% for publisher in object_list %}
      <li>{{ publisher.name }}</li>
   {% endfor %}
</ul>
```

**But if you provide like {"context_object_name": "publisher_list"}, then you can write view like:**

```
<ul>
   {% for publisher in publisher_list %}
      <li>{{ publisher.name }}</li>
   {% endfor %}
</ul>
```

**That means you can change the original parameter name(object_list) into any name through "context_object_name" for your view.**

# Views

Function based views were quite simple, and was very hard to extend or customize them.

To address those issues, the class-based views was created.

Now, views are always functions. Even class-based views.

When we add them to the **URL conf using the View.as_view()** class method, it returns a function.

```
class View:
    @classonlymethod
    def as_view(cls, **initkwargs):
        """Main entry point for a request-response process."""
        for key in initkwargs:
            # Code omitted for clarity
```

# Generic display views

- The two following generic class-based views are designed to display data. On many projects they are typically the most commonly used views.

  - **DetailView**
  - **ListView: While this view is executing, self.object_list will contain the list of objects (usually, but not necessarily a queryset) that the view is operating upon.**

https://docs.djangoproject.com/en/2.2/ref/class-based-views/generic-display/#detailview

# Any Queries