**Activating Admin Interfaces, Using Admin Interfaces, Customizing Admin Interfaces, Reasons to use Admin Interfaces. Form Processing, Creating Feedback forms, Form submissions, custom validation, creating Model Forms, URLConf Ticks, Including Other URLConfs**

1. **Discuss Migration of Database with an example.**
2. **Create a simple Django project called urls\_dispatcher\_example with two application(articles and Blog).**
3. **Explain steps of Configuring URLs in Django.**
4. **Discuss Django Form Submission.**

**Django Admin Interface**

The Django admin interface is a built-in feature of the Django web framework that provides a web-based interface for managing application data. It allows developers and administrators to perform CRUD (Create, Read, Update, Delete) operations on models without having to write any custom code for these basic operations. This interface is highly customizable and can be tailored to fit the specific needs of a project.

**Key Features of Django Admin Interface**:

1. **Automated Admin Interface**:
   * Automatically generates a user-friendly interface for all models registered with it.
   * Provides forms for creating, updating, and deleting records.
2. **User Authentication and Authorization**:
   * Integrates with Django’s authentication system to manage user permissions.
   * Allows setting different permission levels for different users (e.g., superusers, staff).
3. **Data Management**:
   * Supports listing, searching, filtering, and ordering data.
   * Enables bulk actions (e.g., deleting multiple records at once).
4. **Customization**:
   * Admin classes (ModelAdmin) can be used to customize the admin interface for specific models.
   * Supports customizing form fields, list displays, and other aspects of the interface.
5. **Extensibility**:
   * Allows adding custom actions, views, and templates to the admin interface.
   * Supports plugins and third-party extensions to enhance functionality.
6. **Security**:
   * Access to the admin interface is restricted to authenticated and authorized users.
   * Sensitive operations are protected by CSRF (Cross-Site Request Forgery) tokens and other security measures.

**Activating Admin Interfaces**

When you create a Django model, you can easily add it to the admin interface by registering it with Django's admin site. Here is an example:

**Step 1: Define a Model:**

**from django.db import models**

**class Book(models.Model):**

**title = models.CharField(max\_length=100)**

**authors = models.ManyToManyField('Author')**

**publisher = models.ForeignKey('Publisher', on\_delete=models.CASCADE)**

**publication\_date = models.DateField()**

**num\_pages = models.IntegerField(blank=True, null=True)**

**Step 2:** **Register the Model with the Admin Site**:

**from django.contrib import admin**

**from .models import Book**

**@admin.register(Book)**

**class BookAdmin(admin.ModelAdmin):**

**list\_display = ('title', 'publisher', 'publication\_date')**

**search\_fields = ('title', 'authors\_\_name')**

**Step 3: Access the Admin Interface:**

* Add django.contrib.admin to INSTALLED\_APPS.

**# settings.py**

**INSTALLED\_APPS = [**

**'django.contrib.admin',]**

* Include the admin URL patterns in your urls.py.

**from django.conf.urls import include, url**

**from django.contrib import admin**

**urlpatterns = [**

**# other URL patterns**

**url(r'^admin/', include(admin.site.urls))]**

**After the Run Development Server python manage.py runserver**

**Using the Admin Interface**

The Django admin interface provides a comprehensive and user-friendly way to manage your application's data. Here's a detailed look at what you can do with it, including some specific examples:

**1. Login Screen**

To access the admin interface, you need to log in with an admin or staff user account. The login screen ensures that only authorized users can access and manage the data.

* **URL**: /admin/login/
* **Fields**: Username and Password
* **Actions**: Login button to authenticate the user.

**2. Admin Index Page**

After logging in, you're redirected to the admin index page. This page lists all the models that are registered with the admin site.

* **URL**: /admin/
* **Features**:
  + A list of all registered models grouped by the application.
  + Links to the change list views of each model.

**3. Change List View**

The change list view displays all the records for a specific model. It provides a way to list, search, filter, and perform bulk actions on records.

* **URL**: /admin/app\_name/model\_name/
* **Features**:
  + **List Display**: Shows selected fields of each record in a tabular format.
  + **Search**: A search bar to find records by specific fields.
  + **Filters**: Sidebar filters to narrow down the list based on certain criteria (e.g., by date, by related model).
  + **Actions**: Bulk actions like delete, export, or custom actions defined by the developer.
  + **Add Button**: A button to add a new record for the model.

**4. Change Form View**

The change form view is used for creating new records or editing existing ones. It provides a form based on the model's fields.

* **URL**: /admin/app\_name/model\_name/add/ (for adding new records) or /admin/app\_name/model\_name/{id}/change/ (for editing existing records).
* **Features**:
  + **Form Fields**: Inputs for all the fields defined in the model.
  + **Inline Editing**: Allows editing related objects directly within the form.
  + **Save Buttons**: Save, Save and continue editing, and Save and add another.

**5. Deleting Records**

You can delete records from the change list view or the change form view.

* **URL**: Typically accessed via the change form view or bulk action.
* **Features**:
  + A confirmation page to confirm the deletion of the record(s).

**Customizing the Admin Interface**

Customizing the Django admin interface allows you to tailor the admin site to better fit your needs and improve the user experience. Here’s a detailed guide on how to customize the admin interface: