PYTHON - DB AND PYTHON FRAMEWORK - INDUSTRY PROGRAM

Vadhiya_Niraj

1. Why Django should be used for web-development? Explain how you can create a project in Django?

Ans: - Django is a powerful and widely-used web framework that makes building web applications faster and easier. Here are some reasons why you should use Django for web development:

1. Rapid Development: Django follows the "Don't Repeat Yourself" (DRY) principle, which helps developers build complex applications with minimal code.
2. Security: Django has built-in protection against many common web security issues, including SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
3. Scalability: Django is designed to handle high-traffic sites, making it ideal for large applications.
4. Versatility: You can use Django for all kinds of web applications, from simple websites to complex enterprise solutions.
5. Built-in Admin Interface: Django automatically generates an admin panel, which makes managing data and models very easy without writing additional code.

6. ORM (Object-Relational Mapping): Django's ORM allows developers to interact with databases using Python code instead of SQL, streamlining database queries and management.

How to Create a Project in Django:

To create a Django project, follow these steps:

1. Install Django:
   pip install Django

2. **Create a New Django Project: Use the django-admin tool to create a new project.**

   django-admin startproject myproject

   cd myproject

3. **Create a Django App: Django projects consist of one or more apps. You can create an app within the project**

   python manage.py startapp myapp

4. **Run the Development Server**: **To test if everything is set up correctly, start the development server:**

   python manage.py runserver

5. **Visit the Project**: **Open your browser and go to http://127.0.0.1:8000/ to see the Django welcome page.**

2. How to check installed version of django?

Ans: - To check the version of Django installed on your system, run this command

django-admin –version & python -m django --version

3. Explain what does django-admin.py make messages command is used for?

Ans: - The django-admin.py makemessages command is used to create or update the .po files that Django uses for localization (translations). It extracts all text marked for translation in your project and prepares message files for various languages. You can use it like this:

django-admin makemessages -l <language_code>

For example:

django-admin makemessages -l de # Create translation files for German

4. What is Django URLs?make program to create django urls

Ans: - Django URLs are used to map web requests to views. A URL pattern is a string that is matched to a request, and the corresponding view is called to process the request.

Example Program to Create Django URLs:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

In views.py, define the view:

```
from django.shortcuts import render


def index(request):
    return render(request, 'index.html')
```

5. What is a QuerySet?Write program to create a new Post object in database:

Ans: - A QuerySet is a collection of database queries that Django's ORM returns. It represents a list of records from the database that you can iterate over or further filter.

Program to Create a New Post Object in Database:

Assuming you have a Post model:

1. Define the model in models.py:

```
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=100)
    content = models.TextField()

    def __str__(self):
```

```
        return self.title
```

2. To create a new Post in the database:

```
post = Post.objects.create(title="My First Post",
content="This is the content of my first post.")
```

3. Querying the Post:

```
posts = Post.objects.all()  # Get all posts
```

6. Mention what command line can be used to load data into Django?

Ans: -You can use the loaddata command to load data from a JSON, XML, or YAML file into your Django app.

Example:

```
python manage.py loaddata <data_file>.json
```

7. Explain what does django-admin.py make messages command is used for?

Ans: - The django-admin.py makemessages command is used for localization and translation in Django projects. It is part of Django's internationalization (i18n) framework, which allows you to create applications that can support multiple languages.

8. Make Django application to demonstrate following things o There will be 2 modules(Admin,Product manager) o Admin can add product name (ex.Product id and product name) ex. (1, Samsung), (2, Apple)...etc. Data should store in • Product_mst table with product id as primary key o Admin can add product subcategory details Like (Product price, product image, Product model, product Ram) data should store in Product_sub_cat table o Admin can get product name as foreign key from product_mst table in product_sub_category_details page Admin can view, update and delete all registered details

of product manager can search product on search bar and get all details about product.

Ans: - To create a Django application that demonstrates the Product Management system, follow these steps:

1. **Define Models:** In models.py, define ProductMst and ProductSubCat models**:**

```
from django.db import models

class ProductMst(models.Model):
    product_id = models.AutoField(primary_key=True)
    product_name = models.CharField(max_length=100)

    def __str__(self):
        return self.product_name

class ProductSubCat(models.Model):
```

```
    product = models.ForeignKey(ProductMst,
on_delete=models.CASCADE)

    product_price = models.DecimalField(max_digits=10,
decimal_places=2)

    product_image =
models.ImageField(upload_to='products/')

    product_model = models.CharField(max_length=100)

    product_ram = models.CharField(max_length=50)


    def __str__(self):

        return f"{self.product.product_name} -
{self.product_model}"
```

2. **Register Models in Admin**: In admin.py, register the models so the admin interface can manage them:

```
from django.contrib import admin

from .models import ProductMst, ProductSubCat


admin.site.register(ProductMst)

admin.site.register(ProductSubCat)
```

3. **Add Views and URLs**: Define views in views.py to manage products and product subcategories.

```
from django.shortcuts import render
from .models import ProductMst, ProductSubCat

def add_product(request):
    # Handle product addition
    pass

def product_list(request):
    products = ProductMst.objects.all()
    return render(request, 'product_list.html', {'products': products})
```

In urls.py:

```
from django.urls import path
from . import views

urlpatterns = [
```

```
    path('add-product/', views.add_product,
name='add_product'),

    path('products/', views.product_list,
name='product_list'),

]
```

4. **Admin Functions**: Once the app is registered in Django's admin interface, admins can add, update, delete products, and subcategory details through the admin interface.