Django Serializer Interview Questions & Answers

1. What is a model in Django?

A model in Django is a Python class that represents a table in the database. Each attribute of the model class is mapped to a database column.

2. How do you define a model in Django?

By creating a subclass of `django.db.models.Model` and defining fields as class attributes.

3. What field types are commonly used in Django models?

CharField, TextField, IntegerField, FloatField, BooleanField, DateTimeField, ForeignKey, ManyToManyField, etc.

4. What is the use of Meta class in Django models?

The Meta class provides metadata to the model such as ordering, db_table name, verbose_name, etc.

5. What is the primary key in Django models?

By default, Django adds an AutoField named 'id' as the primary key. You can also define your own using primary_key=True.

6. How do you create a one-to-many relationship in Django?

Using `models.ForeignKey` field.

7. How do you create a many-to-many relationship in Django?

Using `models.ManyToManyField`.

Django Serializer Interview Questions & Answers

8. How do you create a one-to-one relationship in Django?

Using `models.OneToOneField`.

9. What is the use of `blank` and `null` in model fields?

`blank=True` allows the field to be empty in forms, while `null=True` allows NULL in the database.

10. What is the difference between `auto_now` and `auto_now_add`?

`auto_now` updates timestamp on every save, `auto_now_add` sets timestamp only when the object is created.

11. How do you override the default table name for a model?

In the Meta class, set `db_table = 'your_table_name'`.

12. What are model managers in Django?

Managers are interfaces through which database query operations are provided to Django models.

13. How can you create a custom manager in Django?

By extending `models.Manager` and adding custom methods.

14. What is the difference between `objects.all()` and `objects.filter()`?

`all()` retrieves all records, `filter()` returns a queryset matching specific conditions.

15. What is the use of `get()` method in Django ORM?

`get()` returns a single object matching the query. Raises `DoesNotExist` if no match,

`MultipleObjectsReturned` if multiple.

16. How do you define choices in Django model fields?

Using a `choices` argument with a list of tuples:

`STATUS_CHOICES = [('D', 'Draft'), ('P', 'Published')]`

17. What is a proxy model in Django?

A proxy model uses the same database table as the original model but can have different Python behavior.

18. What is an abstract base class in Django models?

An abstract model is a base class that other models inherit from, but it's not created as a database table itself.

19. What is a `Meta.abstract = True` used for?

It tells Django not to create a table for the abstract base model.

20. How can you perform raw SQL queries in Django?

Using `Model.objects.raw('SQL QUERY')` or `connection.cursor()` for custom queries.

21. How do you perform migrations in Django?

Run `python manage.py makemigrations` and `python manage.py migrate`.

22. How do you delete a model instance in Django?

Call `.delete()` method on a model instance.

Django Serializer Interview Questions & Answers

23. How do you update a model instance in Django?

Change its fields and call `.save()` method.

24. What is the `save()` method in Django models?

Saves the current instance to the database. Can be overridden for custom behavior.

25. What is the `__str__` method in Django models used for?

It returns a human-readable string representation of the model instance.