

🌀 Advanced Django MCQs (1–50)

📁 Django Core & Architecture

1. **What is the purpose of Django's `settings.py` file?**
 - A) To configure project-wide settings including databases, middleware, and apps
 - B) To define database models
 - C) To handle HTTP requests
 - D) To manage templates**Answer: A**
 2. **What is the role of `manage.py` in a Django project?**
 - A) It's a command-line utility for administrative tasks like running server, migrations
 - B) It handles URL routing
 - C) It manages templates
 - D) It serves static files**Answer: A**
 3. **Which Django component handles URL routing?**
 - A) `urls.py` files in the project and apps
 - B) `views.py`
 - C) `models.py`
 - D) `middleware.py`**Answer: A**
-

📁 Models & ORM

4. **What is the default primary key field type in Django models?**
 - A) `AutoField` (an integer that auto-increments)
 - B) `CharField`
 - C) `UUIDField`
 - D) `ForeignKey`**Answer: A**
5. **How do you create a one-to-many relationship in Django models?**
 - A) Using `ForeignKey` in the “many” model pointing to the “one” model
 - B) Using `OneToOneField`
 - C) Using `ManyToManyField`
 - D) Using `CharField`**Answer: A**
6. **Which method must you override to customize the string representation of a Django model instance?**
 - A) `__str__()`
 - B) `__repr__()`
 - C) `toString()`
 - D) `__unicode__()`**Answer: A**

7. **What does `select_related()` do in Django ORM?**
A) Performs a SQL join to retrieve related objects in a single query
B) Filters querysets
C) Creates a new database table
D) Deletes related objects

Answer: A

8. **How do you ensure a model field is unique in the database?**
A) Set `unique=True` on the field
B) Use `primary_key=True` on the field
C) Override `clean()` method
D) Use `unique_together` only

Answer: A

Views

9. **What is the difference between function-based views (FBVs) and class-based views (CBVs)?**
A) FBVs are simple Python functions; CBVs are classes that provide reusable methods and mixins
B) FBVs cannot handle forms
C) CBVs don't support URL parameters
D) FBVs are deprecated

Answer: A

10. **Which Django mixin can be used with CBVs to restrict access to authenticated users only?**
A) `LoginRequiredMixin`
B) `PermissionRequiredMixin`
C) `UserPassesTestMixin`
D) `AnonymousUserMixin`

Answer: A

11. **What method in a CBV handles GET requests?**
A) `get()`
B) `post()`
C) `dispatch()`
D) `handle()`

Answer: A

Middleware

12. **What is middleware in Django?**
A) A framework of hooks to process requests/responses globally before reaching views
B) A template engine
C) Database connection layer

D) URL dispatcher

Answer: A

13. In which order are middleware classes applied?

A) In the order they are listed in `MIDDLEWARE` for requests, reverse order for responses

B) Reverse order for requests and responses

C) Random order

D) Alphabetical order

Answer: A

Authentication & Authorization

14. What Django app provides authentication support?

A) `django.contrib.auth`

B) `django.contrib.sessions`

C) `django.contrib.admin`

D) `django.contrib.contenttypes`

Answer: A

15. How can you enforce permission checks on views?

A) Using decorators like `@permission_required` or mixins like `PermissionRequiredMixin`

B) Middleware only

C) By overriding `dispatch()`

D) No way to enforce permissions

Answer: A

16. What is the purpose of the `user` model in Django?

A) Represents users and their authentication information

B) Stores blog posts

C) Handles HTTP requests

D) Manages database migrations

Answer: A

Forms

17. How does Django validate form input?

A) Using form fields' built-in validators and custom `clean()` methods

B) Only client-side validation

C) Using middleware

D) Forms are not validated

Answer: A

18. What method is called to save a form's data to the database?

A) `form.save()`

B) `form.clean()`

C) `form.validate()`

D) `form.submit()`

Answer: A

Django REST Framework (DRF)

19. What class is typically extended to create API views in DRF?

- A) `APIView` or `GenericAPIView`
- B) `View`
- C) `FormView`
- D) `ListView`

Answer: A

20. How do serializers differ from Django forms?

- A) Serializers convert complex data (models/querysets) to native Python datatypes and validate input for APIs
- B) Forms only validate data for HTML forms
- C) Serializers handle only database queries
- D) No difference

Answer: A

21. Which DRF class provides built-in list and create functionality?

- A) `ListCreateAPIView`
- B) `RetrieveAPIView`
- C) `CreateAPIView`
- D) `UpdateAPIView`

Answer: A

Signals

22. What are signals used for in Django?

- A) To decouple certain actions by letting different parts of the app listen for specific events
- B) For database queries
- C) For rendering templates
- D) For middleware processing

Answer: A

23. Which signal is sent after a model instance is saved?

- A) `post_save`
- B) `pre_save`
- C) `post_delete`
- D) `pre_delete`

Answer: A

24. How do you connect a signal handler to a signal?

- A) Using `signal.connect(receiver_function)`
- B) Using `receiver_function.connect()`
- C) Signals connect automatically
- D) Signals are deprecated

Answer: A

🔧 Database & Migrations

25. What command creates migration files based on model changes?

- A) `python manage.py makemigrations`
- B) `python manage.py migrate`
- C) `python manage.py runserver`
- D) `python manage.py flush`

Answer: A

26. What is the purpose of Django migrations?

- A) Track and apply incremental changes to the database schema
- B) Back up the database
- C) Import data
- D) Generate admin interface

Answer: A

27. Which migration operation removes a field from a model?

- A) `RemoveField`
- B) `DeleteField`
- C) `DropField`
- D) `RemoveColumn`

Answer: A

⚙️ Templates

28. Which Django template tag is used to extend a base template?

- A) `{% extends "base.html" %}`
- B) `{% include "base.html" %}`
- C) `{% load "base.html" %}`
- D) `{% block "base.html" %}`

Answer: A

29. How can you pass variables from views to templates?

- A) Through the context dictionary passed in `render()`
- B) Through middleware
- C) Through URLs only
- D) Via sessions only

Answer: A

🛡️ Security

30. What is the default CSRF protection mechanism in Django?

- A) Middleware that verifies CSRF tokens in POST requests
- B) No default protection
- C) Only client-side validation
- D) HTTPS only

Answer: A

31. Which decorator can you use to exempt a view from CSRF validation?

- A) `@csrf_exempt`
- B) `@login_required`
- C) `@csrf_protect`
- D) `@safe`

Answer: A

Static & Media Files

32. Where should you place user-uploaded files in Django?

- A) In the `MEDIA_ROOT` directory
- B) In the `STATIC_ROOT` directory
- C) In the `templates` directory
- D) In the root project folder

Answer: A

33. Which settings configure static files handling in production?

- A) `STATIC_ROOT` and `STATIC_URL`
- B) `MEDIA_ROOT` and `MEDIA_URL`
- C) `TEMPLATES_DIR`
- D) `STATICFILES_DIRS` only

Answer: A

Deployment

34. Which WSGI server is commonly used to deploy Django applications?

- A) Gunicorn
- B) Apache
- C) Nginx
- D) uWSGI

Answer: A (though uWSGI also common)

35. What is the purpose of `collectstatic`?

- A) To gather static files into `STATIC_ROOT` for production serving
- B) To migrate the database
- C) To run tests
- D) To restart the server

Answer: A

☐ Advanced Concepts

36. What is the role of Django's `ContentType` framework?

- A) To allow generic relations by tracking all models installed
- B) To serve media files
- C) To handle forms

D) To provide authentication

Answer: A

37. **What is `prefetch_related()` used for?**

A) To fetch many-to-many and reverse foreign key related objects efficiently in separate queries

B) To prefetch foreign keys in the same query

C) To sort querysets

D) To filter querysets

Answer: A

Caching

38. **Which cache backend is provided by Django by default?**

A) Local-memory cache

B) Redis

C) Memcached

D) Database cache only

Answer: A

39. **How can you cache a view's entire output in Django?**

A) Using the `@cache_page` decorator

B) Using middleware only

C) Using `cache.set()` only

D) Caching not supported

Answer: A

☐ Testing

40. **Which Django module provides tools for testing?**

A) `django.test`

B) `django.forms`

C) `django.views`

D) `django.urls`

Answer: A

41. **How do you simulate a client request in Django tests?**

A) Using `Client` class from `django.test`

B) Using `RequestFactory` only

C) Using Selenium only

D) By calling views directly only

Answer: A

Signals Advanced

42. **How to prevent a signal handler from running multiple times during one save operation?**

- A) Use a flag or disconnect signal temporarily
- B) Signals always run once automatically
- C) Use middleware instead
- D) Signals are not suitable for this

Answer: A

☐ **Miscellaneous**

43. **What does the `@receiver` decorator do?**

- A) Connects a signal to a receiver function in a cleaner way
- B) Starts a view
- C) Handles authentication
- D) Cleans form data

Answer: A

44. **How can you make a Django model abstract?**

- A) By adding `abstract = True` inside `Meta` class
- B) By inheriting from `AbstractModel`
- C) By setting `abstract_model = True`
- D) By making it a proxy model

Answer: A

45. **What is a proxy model in Django?**

- A) A subclass of a model that doesn't create a new table but changes Python behavior
- B) A model for proxies
- C) A model for database views
- D) An abstract model

Answer: A

46. **What is the `sites` framework used for?**

- A) Managing multiple websites with one Django project
- B) Caching pages
- C) Authentication
- D) Forms handling

Answer: A

47. **Which Django command shows SQL queries generated by a queryset?**

- A) `str(queryset.query)`
- B) `queryset.sql()`
- C) `python manage.py sql`
- D) `queryset.show()`

Answer: A

48. **What method do you override to customize querysets returned by a model's manager?**

- A) `get_queryset()`
- B) `get()`
- C) `filter()`
- D) `all()`

Answer: A

49. **How do you create a custom manager in Django?**

- A) Subclass `models.Manager` and add methods, then assign it to model's `objects` attribute
- B) Override `__init__`
- C) Use `@manager` decorator
- D) Modify the model's Meta class

Answer: A

50. **Which setting controls allowed hosts for security?**

- A) `ALLOWED_HOSTS`
- B) `SECURE_HOSTS`
- C) `HOSTS`
- D) `HOSTNAMES`

Answer: A

If you want, I can provide explanations for any question or create more targeted quizzes on any Django topic! Just let me know.

You said:

more

ChatGPT said:

Absolutely! Here are **50 more advanced Django MCQs** to extend your deep dive:

🌀 Advanced Django MCQs (51–100)

⚙️ Advanced Models & ORM

51. **What is the difference between `null=True` and `blank=True` in model fields?**

- A) `null=True` affects the database (allows NULL), `blank=True` affects form validation (allows empty input)
- B) Both mean the same
- C) `null=True` affects forms only, `blank=True` affects the database
- D) Neither affects forms or database

Answer: A

52. **What is a `through` model in a `ManyToManyField`?**

- A) A custom intermediary model that defines the relationship explicitly
- B) A built-in table created automatically
- C) A proxy model
- D) A field for one-to-one relationships

Answer: A

53. **How do you implement model inheritance in Django?**

- A) By subclassing `models.Model` or other models with abstract/base/proxy inheritance
- B) Using `extends` keyword
- C) Using signals
- D) Models cannot inherit

Answer: A

54. **Which method of a model manager returns a new queryset with filters applied?**

- A) `filter()`
- B) `get()`
- C) `create()`
- D) `update()`

Answer: A

Model Field Options

55. **Which option would you use to index a field in Django?**

- A) `db_index=True`
- B) `index=True`
- C) `unique=True`
- D) `primary_key=True`

Answer: A

56. **What does `choices` do in a model field?**

- A) Restricts the field's values to a predefined set
- B) Adds multiple values in one field
- C) Allows null values
- D) Enables foreign key relations

Answer: A

QuerySets and Performance

57. **What does `defer()` do in a Django queryset?**

- A) Defers loading of specified fields until accessed
- B) Loads all fields eagerly
- C) Filters the queryset
- D) Deletes the queryset objects

Answer: A

58. **What's the difference between `values()` and `values_list()` in querysets?**

- A) `values()` returns dictionaries, `values_list()` returns tuples or flat lists
- B) They are identical
- C) `values()` returns lists, `values_list()` returns dictionaries
- D) `values_list()` is deprecated

Answer: A

Class-Based Views (CBVs) Advanced

59. What method in CBVs handles POST requests?

- A) `post()`
- B) `get()`
- C) `dispatch()`
- D) `handle_post()`

Answer: A

60. How can you customize the context data passed to templates in CBVs?

- A) Override `get_context_data()`
- B) Override `dispatch()`
- C) Use `render_to_response()` only
- D) Use middleware

Answer: A

61. Which CBV provides built-in support for editing objects?

- A) `UpdateView`
- B) `DetailView`
- C) `ListView`
- D) `CreateView`

Answer: A

62. Which mixin would you use to restrict CBV access to users with specific permissions?

- A) `PermissionRequiredMixin`
- B) `LoginRequiredMixin`
- C) `AccessMixin`
- D) `UserPassesTestMixin`

Answer: A

Middleware Deep Dive

63. How can middleware alter response headers?

- A) By modifying the response object in `process_response()` or `__call__()`
- B) Middleware cannot alter responses
- C) Only the view can modify headers
- D) By modifying request objects only

Answer: A

64. What is the difference between old-style and new-style middleware in Django?

- A) New-style middleware is a callable class implementing `__call__` with `get_response`; old style uses separate `process_request` etc. methods
- B) Old style supports async, new style doesn't
- C) New style middleware must be functions
- D) No difference

Answer: A

Authentication & Security Advanced

65. What is Django's `UserManager` class used for?
- A) To create and manage `User` instances, including helper methods like `create_user()`
 - B) To handle sessions
 - C) To manage middleware
 - D) To send emails
- Answer: A**
66. How do you implement custom user models?
- A) Subclass `AbstractBaseUser` and define custom manager
 - B) Subclass `User` directly
 - C) Use `auth.User` as is
 - D) Modify the default `User` model fields directly
- Answer: A**
67. Which middleware protects against clickjacking attacks?
- A) `XFrameOptionsMiddleware`
 - B) `CsrfViewMiddleware`
 - C) `SecurityMiddleware`
 - D) `SessionMiddleware`
- Answer: A**
68. Which setting enforces HTTPS across the site?
- A) `SECURE_SSL_REDIRECT = True`
 - B) `USE_HTTPS`
 - C) `FORCE_SSL`
 - D) `HTTPS_ENABLED`
- Answer: A**
-

REST Framework Advanced

69. How can you implement throttling in DRF?
- A) By using throttling classes like `UserRateThrottle`
 - B) Only via middleware
 - C) Throttling is not supported
 - D) Only via third-party apps
- Answer: A**
70. What is the purpose of `ViewSet` in DRF?
- A) To combine multiple views (list, create, retrieve) into a single class
 - B) To serve static files
 - C) To generate migrations
 - D) To validate serializers
- Answer: A**
71. Which class provides automatic URL routing for ViewSets?
- A) `DefaultRouter`
 - B) `SimpleRouter`
 - C) `RouteSet`
 - D) `ViewRouter`
- Answer: A**

□ Signals and Asynchronous Tasks

72. What are the caveats of using Django signals for complex workflows?

- A) Can lead to tightly coupled code, harder to debug, and unexpected side effects
- B) Signals are slow
- C) Signals replace middleware
- D) Signals can't access database

Answer: A

73. Which Django package integrates Celery for async tasks?

- A) `django-celery`
- B) `django-async`
- C) `django-rq`
- D) `channels`

Answer: A

■ File Uploads & Storage

74. How do you handle file uploads in Django?

- A) Use `FileField` or `ImageField` in models, and handle files in forms/views
- B) Files are not supported by Django
- C) Store files in sessions
- D) Use `JSONField` for files

Answer: A

75. What does the `upload_to` attribute of a `FileField` specify?

- A) The subdirectory where files are saved
- B) The filename
- C) The file type
- D) The file size

Answer: A

□ Admin Interface

76. How do you customize the Django admin list view?

- A) Use `list_display` in `ModelAdmin` classes
- B) Override templates only
- C) Use middleware
- D) Admin interface cannot be customized

Answer: A

77. Which method is used to add custom actions in Django admin?

- A) Define methods and add them to `actions` list in `ModelAdmin`
- B) Use signals
- C) Use middleware

D) Override `save_model()`

Answer: A

□ Testing Advanced

78. How do you test views with authentication in Django tests?

- A) Use `client.login()` to simulate logged-in user
- B) Use raw HTTP requests
- C) Bypass authentication in tests
- D) Authentication can't be tested

Answer: A

79. Which method resets the database during tests?

- A) `flush` or the test runner's transaction management
- B) `makemigrations`
- C) `migrate`
- D) `collectstatic`

Answer: A

🔗 Miscellaneous Advanced

80. What is the use of Django's `transaction.atomic` decorator?

- A) To run code within a database transaction that can be rolled back on failure
- B) To speed up queries
- C) To run async code
- D) To lock tables

Answer: A

81. How do you create a management command?

- A) Add a Python file in the `management/commands` directory of an app
- B) Override `manage.py`
- C) Use middleware
- D) Use signals

Answer: A

82. What's the purpose of Context Processors?

- A) Inject variables automatically into templates' context globally
- B) Process HTTP requests
- C) Handle signals
- D) Run middleware

Answer: A

83. What is `django.middleware.security.SecurityMiddleware` responsible for?

- A) Setting security-related HTTP headers like HSTS, Content Security Policy
- B) Managing authentication
- C) Database connection pooling
- D) Rendering templates

Answer: A

Routing

84. How do you include app URLs in the main project's `urls.py`?

- A) Use `include('app.urls')` in `urlpatterns`
- B) Directly write all URLs in main `urls.py`
- C) Use middleware
- D) Use `import_urls()`

Answer: A

85. What does the `name` argument in `path()` or `url()` do?

- A) Provides a unique identifier for reverse URL resolution
- B) Adds a display label
- C) Specifies HTTP method
- D) None

Answer: A

Channels and Websockets

86. What is Django Channels used for?

- A) To handle WebSockets and asynchronous protocols alongside HTTP
- B) To manage REST APIs
- C) To optimize SQL queries
- D) To cache templates

Answer: A

87. Which backend is commonly used with Channels?

- A) Redis
- B) Memcached
- C) RabbitMQ
- D) Postgres

Answer: A

Async in Django

88. From which Django version is async view support available?

- A) 3.1+
- B) 2.0+
- C) 1.11+
- D) Not supported yet

Answer: A

89. What keyword marks an asynchronous view?

- A) `async def`
- B) `def async_view`
- C) `await def`

D) `async` function

Answer: A

Internationalization (i18n)

90. How do you mark strings for translation in Django?

A) Using `gettext` or `_()` function

B) Using `translate()` only

C) Using `@translate` decorator

D) Using `trans()`

Answer: A

91. What is the command to generate translation message files?

A) `django-admin makemessages`

B) `maketranslations`

C) `gettext`

D) `makelang`s

Answer: A

Database Indexing and Optimization

92. What is the purpose of the `db_tablespace` option in model Meta?

A) To specify tablespace for the table in supported databases

B) To define database connection

C) To set schema for the table

D) To add an index

Answer: A

93. What is the effect of setting `managed = False` in model Meta?

A) Django won't create or manage the table (used for existing DB tables)

B) Table will be deleted after migration

C) Table is hidden in admin

D) None

Answer: A

Signals Deep

94. Which signal is sent before deleting a model instance?

A) `pre_delete`

B) `post_delete`

C) `pre_save`

D) `post_save`

Answer: A

✂ □ Other Utilities

95. What is `django.contrib.contenttypes` used for?

- A) To enable generic relations between models of different types
- B) For static files
- C) For form validation
- D) For admin interface

Answer: A

96. How do you restrict file types uploaded through a form?

- A) Use custom validators on the `FileField` or `ImageField`
- B) Use middleware
- C) Use static files settings
- D) Not possible

Answer: A

□ Miscellaneous

97. What does the `get_absolute_url()` method in models do?

- A) Returns the canonical URL for a model instance
- B) Gets absolute file path
- C) Gets static URL
- D) None

Answer: A

98. What's the difference between `reverse()` and `redirect()`?

- A) `reverse()` returns URL string from view name, `redirect()` returns an `HttpResponseRedirect`
- B) They do the same
- C) `redirect()` reverses URLs
- D) `reverse()` is deprecated

Answer: A

99. How can you create a proxy model?

- A) Set `proxy = True` in Meta class of a subclass of the original model
- B) Subclass without Meta
- C) Use `proxy=True` in model field
- D) Not supported

Answer: A

100. What's the purpose of `django.middleware.common.CommonMiddleware`?

- A) Handles things like URL normalization, trailing slash appending, and ETags
- B) Authentication
- C) Template rendering
- D) Session management

Answer: A

If you'd like, I can also generate explanations or create PDF files for these MCQs. Just say the word!

You said:

more

ChatGPT said:

Got it! Here's **another batch of 50 advanced Django MCQs** (101–150) to keep your mastery growing:

🌀 Django Advanced MCQs (101–150)

🔪 Models & ORM Deep Dive

101. What's the effect of setting `unique_together` in a model's Meta class?
- A) Enforces uniqueness on the combination of specified fields at the database level
 - B) Makes individual fields unique separately
 - C) Creates indexes on those fields
 - D) None

Answer: A

102. What's the purpose of `select_related()` in querysets?
- A) Performs SQL join to fetch related foreign key objects in one query
 - B) Prefetches related objects in separate queries
 - C) Filters related objects
 - D) Deletes related objects

Answer: A

103. How does `prefetch_related()` differ from `select_related()`?
- A) `prefetch_related()` executes separate queries and joins in Python, ideal for ManyToMany or reverse relations
 - B) They are identical
 - C) `prefetch_related()` is deprecated
 - D) It doesn't exist

Answer: A

🔗 QuerySet Methods

104. What does `annotate()` do in Django querysets?
- A) Adds calculated fields to each object in the queryset
 - B) Filters querysets
 - C) Deletes entries
 - D) Changes order

Answer: A

105. Which function can be used to count related objects efficiently?
- A) `Count` from `django.db.models` annotations

- B) `len()`
 - C) `count()` on related manager only
 - D) `aggregate()` only
- Answer: A**
-

Advanced Authentication

106. **What method would you override to customize password validation?**
- A) Write custom password validators and add them to `AUTH_PASSWORD_VALIDATORS` setting
 - B) Override `User.set_password()`
 - C) Override `authenticate()` function
 - D) Use signals
- Answer: A**
107. **What is Django's `session_auth_hash` used for?**
- A) Invalidates user sessions if their password changes to prevent session hijacking
 - B) Hashes passwords
 - C) Stores CSRF tokens
 - D) None
- Answer: A**
-

Django REST Framework (DRF)

108. **Which DRF class serializes data for complex nested relationships?**
- A) `Serializer` with nested serializer fields
 - B) `ModelSerializer` only
 - C) `BaseSerializer` only
 - D) `ListSerializer` only
- Answer: A**
109. **How can you customize DRF permissions globally?**
- A) Set `DEFAULT_PERMISSION_CLASSES` in `REST_FRAMEWORK` settings
 - B) Use middleware
 - C) Permissions cannot be customized globally
 - D) Override views individually only
- Answer: A**
-

☐ Middleware & Request Lifecycle

110. **What happens if middleware raises an exception during `process_request()`?**
- A) The exception is propagated up and the request is halted, response middleware isn't run
 - B) Middleware chain continues

- C) Exception is ignored
- D) Response with status 200 is returned

Answer: A

111. **Where should heavy computations be handled to avoid slowing down request/response cycle?**

- A) Offload to asynchronous tasks (e.g., Celery) or background jobs
- B) In middleware
- C) Directly in views
- D) In model methods

Answer: A

□ Testing Django

112. **How do you mock database queries in tests?**

- A) Use libraries like `unittest.mock` or `pytest-mock` to patch `queryset` methods
- B) You cannot mock queries
- C) Use `FakeDB` only
- D) Use signals

Answer: A

113. **What's the difference between `TestCase` and `TransactionTestCase`?**

- A) `TestCase` wraps each test in a transaction and rolls back for speed; `TransactionTestCase` does not, allowing testing of transaction behavior
- B) No difference
- C) `TransactionTestCase` is deprecated
- D) `TestCase` is for integration tests only

Answer: A

🌐 Internationalization & Localization

114. **Which setting controls the active language in Django?**

- A) `LANGUAGE_CODE`
- B) `LOCALE_PATHS`
- C) `TIME_ZONE`
- D) `LANGUAGES`

Answer: A

115. **How do you mark a string as non-translatable?**

- A) Use `gettext_noop()` or `ugettext_noop()`
- B) Use `_()`
- C) Use `translate()`
- D) Use `nontranslate()`

Answer: A

🏠 □ Advanced Routing & URL Handling

116. **What's the purpose of `re_path()`?**
A) To define URLs using regex patterns for more complex matching
B) It's deprecated
C) Used for static files only
D) Alias for `path()`
Answer: A
117. **How do you reverse URLs in templates?**
A) Using the `{% url 'name' %}` template tag
B) Using `reverse()` in templates
C) Using `get_url()`
D) Reverse is not possible in templates
Answer: A
-

Views & Templates

118. **How can you cache an entire view response?**
A) Using `@cache_page` decorator
B) Using signals
C) Using middleware only
D) Views can't be cached
Answer: A
119. **What does the `safe` filter in templates do?**
A) Marks a string as safe so it won't be auto-escaped
B) Escapes HTML
C) Adds security headers
D) Nothing
Answer: A
-

Forms & Validation

120. **How do you create custom validators in Django forms?**
A) Define a function or class that raises `ValidationError` and add it to `validators` argument
B) Override `clean()` method only
C) Use signals
D) Cannot create custom validators
Answer: A
121. **What is the difference between `clean()` and `clean_fieldname()` in forms?**
A) `clean_fieldname()` validates individual fields, `clean()` validates form-wide inter-field constraints
B) Both are same
C) `clean()` is for model forms only
D) `clean_fieldname()` is deprecated
Answer: A

⚡ Signals & Asynchronous

122. **How do you disconnect a signal receiver?**

- A) Use `signal.disconnect(receiver)`
- B) Use `receiver.disconnect()`
- C) Signals cannot be disconnected
- D) Use `@receiver(disconnect=True)`

Answer: A

123. **Which Django package adds async ORM support?**

- A) `asgiref` provides async primitives, but full async ORM is experimental or planned for future versions
- B) Django ORM is fully async now
- C) `django-async-orm` official package
- D) None

Answer: A

📦 Admin Interface Customization

124. **How do you add inline models in Django admin?**

- A) Use `InlineModelAdmin` subclasses and register in the `ModelAdmin` via `inlines` attribute
- B) Use middleware
- C) Add fields manually in templates
- D) Not possible

Answer: A

125. **How do you customize admin form widgets?**

- A) Override `formfield_for_dbfield()` or specify form classes in `ModelAdmin`
- B) Use middleware
- C) Override `save_model()` only
- D) Customize admin templates only

Answer: A

🔒 Database Transactions

126. **What does the `select_for_update()` queryset method do?**

- A) Locks selected rows in the database until the transaction is committed
- B) Prefetches related objects
- C) Rolls back transactions
- D) Deletes rows

Answer: A

127. **Which decorator ensures atomic database transactions on a function?**

- A) `@transaction.atomic`
- B) `@atomic_transaction`

- C) `@db_transaction`
- D) `@transaction.commit`

Answer: A

☐ Miscellaneous

128. **How do you add custom management commands?**

- A) Create a file in `management/commands/` inside an app with a class inheriting from `BaseCommand`
- B) Override `manage.py`
- C) Use signals
- D) Modify settings

Answer: A

129. **What is Django's `site` framework?**

- A) Framework to manage multiple sites with one project
- B) Middleware for site caching
- C) Template rendering system
- D) None

Answer: A

130. **What's the purpose of `django.contrib.sessions`?**

- A) Store and retrieve per-user session data
- B) Handle database connections
- C) Manage static files
- D) Provide authentication

Answer: A

✳ Advanced Topics

131. **Which command would you use to inspect migrations?**

- A) `python manage.py showmigrations`
- B) `migrate --list`
- C) `python manage.py migrations`
- D) `inspectmigrations`

Answer: A

132. **What is the purpose of `migrate --fake`?**

- A) Marks migrations as applied without actually running them, useful when syncing existing databases
- B) Rolls back migrations
- C) Deletes migrations
- D) Runs migrations twice

Answer: A

133. **What are `Meta` options `ordering` used for?**

- A) Define default ordering for querysets of that model
- B) Order migration files
- C) Order admin forms

D) Order URLs

Answer: A

☐ Testing

134. **How do you create a test client in Django?**

A) Use `django.test.Client()` in test cases

B) Use raw HTTP requests

C) Use Selenium only

D) Use `TestClient` in views

Answer: A

Static & Media Files

135. **Which setting defines the directory for uploaded media files?**

A) `MEDIA_ROOT`

B) `MEDIA_URL`

C) `STATIC_ROOT`

D) `STATIC_URL`

Answer: A

136. **How do you serve static files in development?**

A) Use `django.contrib.staticfiles` app and `runserver` serves them automatically

B) Use middleware

C) Use nginx only

D) Django can't serve static files

Answer: A

☐ More Miscellaneous

137. **What's the purpose of `get_user_model()`?**

A) Returns the currently active User model, useful with custom users

B) Creates a user instance

C) Authenticates a user

D) None

Answer: A

138. **How do you mark a field as primary key?**

A) Use `primary_key=True` on the field

B) Use `unique=True`

C) Use `id=True`

D) Cannot set primary keys manually

Answer: A

🔍 Debugging & Logging

139. Which Django setting enables debug mode?

- A) `DEBUG = True`
- B) `ENABLE_DEBUG`
- C) `DEBUG_MODE`
- D) `LOG_LEVEL`

Answer: A

140. Where are Django logs typically configured?

- A) In `LOGGING` setting as a dictionary
- B) Middleware
- C) Settings unrelated to logging
- D) In `admin.py`

Answer: A

🏠 Channels & Async

141. Which layer handles WebSocket communication in Django Channels?

- A) Consumers
- B) Views
- C) Middleware
- D) Templates

Answer: A

142. What is an ASGI application?

- A) Asynchronous Server Gateway Interface, supports async protocols like WebSocket
- B) Another name for WSGI
- C) Middleware only
- D) Template system

Answer: A

🔧 Deployment

143. What is the role of `collectstatic`?

- A) Collects all static files into a single directory for production serving
- B) Runs migrations
- C) Creates admin users
- D) Starts the server

Answer: A

144. Which WSGI server is commonly used in production?

- A) Gunicorn
- B) Django dev server
- C) Nginx

D) Apache only

Answer: A

⚙️ Signals

145. Which signal is sent after a model instance is saved?

- A) `post_save`
- B) `pre_save`
- C) `post_delete`
- D) `pre_delete`

Answer: A

146. Can signals be used to enforce business logic?

- A) Yes, but overuse can make debugging difficult
- B) No
- C) Only for authentication
- D) Only for database migrations

Answer: A

📄 Forms

147. Which method is called when a form's data is validated?

- A) `is_valid()`
- B) `validate()`
- C) `cleaned_data()`
- D) `check()`

Answer: A

148. How do you render a Django form manually in a template?

- A) Use form fields explicitly like `{{ form.field_name }}`
- B) Use `{{ form.render }}`
- C) Use middleware
- D) Use static files

Answer: A

📄 Templates

149. Which template tag is used to extend another template?

- A) `{% extends "base.html" %}`
- B) `{% include "base.html" %}`
- C) `{% import "base.html" %}`
- D) `{% load "base.html" %}`

Answer: A

150. How do you load custom template tags?

- A) `{% load tag_library_name %}`

- B) `{% import tag_library %}`
- C) `{% include tag_library %}`
- D) Automatically loaded

Answer: A

If you want explanations, topics split into PDFs, or even quiz games based on these questions, just let me know! Want me to keep going?

You said:

more

ChatGPT said:

Here's the next batch of **Django advanced MCQs (151–200)** for you — let's keep that learning momentum going!

🔗 Django Advanced MCQs (151–200)

🔗 Advanced QuerySets & ORM

151. **What does the `values()` method return in a queryset?**
- A) A list of dictionaries with specified fields and values
 - B) A list of model instances
 - C) JSON response
 - D) None
- Answer: A**
152. **How does `defer()` optimize queries?**
- A) It excludes specified fields from the initial query, fetching them later on demand
 - B) It caches fields
 - C) It deletes fields from database
 - D) It encrypts fields
- Answer: A**
153. **Which method can you use to force a queryset to evaluate immediately?**
- A) Call `list(queryset)` or `queryset.count()`
 - B) Call `queryset.defer()`
 - C) Call `queryset.lazy()`
 - D) Call `queryset.annotate()`
- Answer: A**
-

🔗 Custom Model Managers

154. **Why create a custom manager in Django?**
A) To add extra queryset methods or modify queryset behavior for a model
B) To create custom views
C) To define templates
D) To handle migrations
Answer: A
155. **How do you set a custom manager as the default for a model?**
A) Assign it to the model's `objects` attribute
B) Use `Meta` class
C) Override `save()` method
D) Use signals
Answer: A
-

❑ Complex Relationships

156. **How do you define a self-referential foreign key?**
A) Use `models.ForeignKey('self', on_delete=...)`
B) Use `models.ManyToManyField('self')` only
C) Use `self()` method
D) It's not allowed
Answer: A
157. **What is the purpose of the `through` argument in `ManyToManyField`?**
A) Specify an explicit intermediate model for the relation
B) Defines cascade rules
C) Specifies uniqueness constraints
D) None
Answer: A
-

⚙️❑ Middleware & Request/Response

158. **In which order is middleware applied?**
A) `process_request` from top to bottom, `process_response` from bottom to top
B) Both in top-to-bottom order
C) Random order
D) Only `process_request` runs
Answer: A
159. **Which middleware is required for CSRF protection?**
A) `django.middleware.csrf.CsrfViewMiddleware`
B) `SessionMiddleware`
C) `CommonMiddleware`
D) `AuthenticationMiddleware`
Answer: A
-

Security

160. **How does Django protect against SQL injection?**

- A) By using parameterized queries and ORM abstractions
- B) It doesn't protect automatically
- C) Using raw SQL only
- D) Using middleware

Answer: A

161. **What setting controls the HTTP-only flag on cookies?**

- A) `SESSION_COOKIE_HTTPONLY`
- B) `CSRF_COOKIE_HTTPONLY`
- C) `SECURE_SSL_REDIRECT`
- D) `SESSION_EXPIRE_AT_BROWSER_CLOSE`

Answer: A

Django REST Framework (DRF) Advanced

162. **How do you customize the default pagination style in DRF?**

- A) Override `DEFAULT_PAGINATION_CLASS` in `REST_FRAMEWORK` settings
- B) Use middleware
- C) Pagination can't be customized
- D) Override `APIView` only

Answer: A

163. **What is the use of `throttle_classes` in DRF views?**

- A) Limit the rate of incoming requests per user or IP
- B) Cache responses
- C) Authenticate users
- D) Serialize data

Answer: A

☐ Signals

164. **Which signal is sent before a model instance is deleted?**

- A) `pre_delete`
- B) `post_delete`
- C) `pre_save`
- D) `post_save`

Answer: A

165. **What is a potential downside of using signals for business logic?**

- A) They can make code harder to follow and debug
- B) Signals always improve performance
- C) They are deprecated
- D) No downside

Answer: A

□ Testing

166. **How can you simulate logged-in users in Django tests?**

- A) Use `self.client.login()` method
- B) Manually create session cookies
- C) It's not possible
- D) Use middleware

Answer: A

167. **Which decorator can be used to run tests with an isolated database?**

- A) `@override_settings` with `DATABASES` or use `TransactionTestCase`
- B) `@mock_database`
- C) `@isolated_db`
- D) None

Answer: A

■ Templates

168. **What does the `{% block %}` tag do?**

- A) Defines a block of content that child templates can override
- B) Loads static files
- C) Adds comments
- D) Renders variables

Answer: A

169. **How do you include another template inside the current template?**

- A) `{% include "template_name.html" %}`
- B) `{% load "template_name.html" %}`
- C) `{% extends "template_name.html" %}`
- D) `{% import "template_name.html" %}`

Answer: A

🌐 Internationalization (i18n)

170. **Which function marks a string for translation in Python code?**

- A) `gettext()` or alias `_()`
- B) `trans()`
- C) `translate()`
- D) `localize()`

Answer: A

171. **How do you compile message files for translations?**

- A) Use `django-admin compilemessages`
- B) Use `makemessages` only
- C) Use `gettext` only

D) Done automatically

Answer: A

🏠 □ Routing & URLs

172. **What is the purpose of URL namespaces?**

- A) Organize URLs and avoid name collisions between apps
- B) Create dynamic URLs only
- C) Handle middleware order
- D) Manage static files

Answer: A

173. **How can you reverse a namespaced URL?**

- A) Use `reverse('namespace:url_name')`
- B) Use `reverse('url_name')` only
- C) Namespaces cannot be reversed
- D) Use `get_url()`

Answer: A

✂ □ Forms

174. **What is the purpose of `ModelForm`?**

- A) Automatically create a form from a model's fields
- B) Serialize data
- C) Manage sessions
- D) Create database tables

Answer: A

175. **Which method do you override to add cross-field validation in a form?**

- A) `clean()`
- B) `clean_fieldname()`
- C) `is_valid()`
- D) `validate()`

Answer: A

⚡ Asynchronous Support

176. **Which Django version introduced async views support?**

- A) Django 3.1
- B) Django 2.0
- C) Django 4.0
- D) Django 1.11

Answer: A

177. **Can Django ORM be used asynchronously natively?**

- A) Not fully yet; async ORM support is experimental or partial as of Django 4.x

- B) Fully supported
- C) No plans to support
- D) ORM is synchronous only

Answer: A

Static and Media Files

178. Which management command collects static files?

- A) `collectstatic`
- B) `migratestatic`
- C) `staticcollect`
- D) `collectfiles`

Answer: A

179. How do you serve media files during development?

- A) Use `django.views.static.serve` with proper URL patterns and `MEDIA_URL/MEDIA_ROOT` settings
- B) Use middleware
- C) Nginx only
- D) Django cannot serve media files

Answer: A

Debugging & Profiling

180. Which Django setting should be False in production to avoid exposing sensitive data?

- A) `DEBUG`
- B) `ALLOWED_HOSTS`
- C) `DATABASES`
- D) `STATIC_URL`

Answer: A

181. Which package helps with profiling and debugging Django apps?

- A) `django-debug-toolbar`
- B) `django-profiling`
- C) `django-logger`
- D) `django-monitor`

Answer: A

Database & Transactions

182. How do you enforce referential integrity at the Django model level?

- A) Use `ForeignKey` with appropriate `on_delete` actions
- B) Use signals
- C) Use raw SQL only

D) Not enforceable

Answer: A

183. **What does the `on_delete=models.PROTECT` option do?**

A) Prevent deletion of referenced objects if related objects exist

B) Cascade deletion

C) Set null on delete

D) No action

Answer: A

☐ Miscellaneous

184. **How do you specify a custom User model?**

A) Set `AUTH_USER_MODEL` in settings to your custom model path

B) Override default User class only

C) Use signals

D) Use middleware

Answer: A

185. **Which command creates initial migrations for models?**

A) `python manage.py makemigrations`

B) `python manage.py migrate`

C) `python manage.py create_migrations`

D) `python manage.py initdb`

Answer: A

If you want, I can keep going with more, or break these into downloadable PDFs or quizzes!
What do you think?

You said:

more

ChatGPT said:

Absolutely! Here's the next batch of **Django advanced MCQs (201–250)** to keep the deep dive going strong:

🌀 Django Advanced MCQs (201–250)

🔍 Advanced ORM & Query Optimization

201. **What does the `select_related()` method do?**
A) Performs a SQL join and includes related objects in the same query (for foreign keys)
B) Defers loading of related objects
C) Executes raw SQL queries
D) Aggregates results
Answer: A
202. **When should you use `prefetch_related()` instead of `select_related()`?**
A) When querying ManyToMany or reverse ForeignKey relationships
B) For single ForeignKey only
C) When you want to skip related fields
D) For raw SQL queries
Answer: A
203. **How can you prevent N+1 query problems in Django ORM?**
A) Use `select_related()` or `prefetch_related()`
B) Use `filter()`
C) Use raw SQL only
D) Avoid joins
Answer: A
-

Custom Fields & Validators

204. **How can you create a custom model field?**
A) By subclassing `models.Field` and implementing required methods
B) By writing a custom form only
C) By subclassing `models.Model`
D) By overriding `save()` method
Answer: A
205. **Where do you define custom validation logic for model fields?**
A) Override the model's `clean()` method or use `validators` attribute on fields
B) In the view only
C) In middleware
D) In templates
Answer: A
-

Class-based Views (CBV) Deep Dive

206. **Which method in a CBV handles GET requests by default?**
A) `get()`
B) `post()`
C) `dispatch()`
D) `handle()`
Answer: A
207. **What is the role of `dispatch()` method in Django CBVs?**
A) It routes the HTTP request to the appropriate handler method (`get`, `post`, etc.)

- B) It handles database transactions
- C) It renders templates only
- D) It manages URL routing

Answer: A

208. **How do you add context data in a `DetailView`?**

- A) Override the `get_context_data()` method
- B) Modify the template
- C) Use `extra_context` attribute only
- D) Use signals

Answer: A

🔒 Authentication & Authorization

209. **What decorator restricts access to logged-in users only?**

- A) `@login_required`
- B) `@permission_required`
- C) `@authenticated_only`
- D) `@user_passes_test`

Answer: A

210. **How do you require a user to have a specific permission for a view?**

- A) Use `@permission_required('app.permission_code')` decorator
- B) Use `@login_required` only
- C) Use middleware only
- D) Use raw SQL permissions check

Answer: A

211. **Which model does Django use by default to represent users?**

- A) `auth.User`
- B) `auth.Profile`
- C) `custom.User`
- D) `models.User`

Answer: A

🔗 Signals and Lifecycle Hooks

212. **What signal would you use to perform an action after a model instance is saved?**

- A) `post_save`
- B) `pre_save`
- C) `pre_delete`
- D) `post_delete`

Answer: A

213. **How can you connect a signal handler to a signal?**

- A) Using `@receiver(signal, sender=Model)` decorator or `signal.connect()` method
- B) Using middleware only

- C) Automatically done by Django
 - D) Using templates
- Answer: A**
-

Static Files & Asset Management

214. **What is the purpose of the `STATIC_ROOT` setting?**
- A) Directory where `collectstatic` collects all static files for production
 - B) Location of media files
 - C) Path to static files in app directories
 - D) Where user uploads are stored
- Answer: A**
215. **What is the difference between `STATICFILES_DIRS` and `STATIC_ROOT`?**
- A) `STATICFILES_DIRS` lists locations of static files in development, `STATIC_ROOT` is the single directory where files are collected for deployment
 - B) Both are the same
 - C) `STATICFILES_DIRS` is for media files
 - D) `STATIC_ROOT` is for templates
- Answer: A**
-

URL Routing and Resolvers

216. **What is the purpose of the `path()` function in Django URLs?**
- A) Defines URL patterns with simple, human-readable syntax
 - B) Defines middleware
 - C) Defines templates
 - D) Manages static files
- Answer: A**
217. **How do you capture a string parameter in a URL pattern?**
- A) Use `<str:param_name>` inside the path string
 - B) Use `{param_name}`
 - C) Use `:param_name`
 - D) Use `?param_name=`
- Answer: A**
-

Management Commands

218. **How do you create a custom Django management command?**
- A) Create a `management/commands` directory inside an app and add a Python file with a `Command` class
 - B) Modify `settings.py`
 - C) Create a custom middleware

D) Create a template tag

Answer: A

219. Which method do you override to define the logic of a custom management command?

A) `handle()`

B) `execute()`

C) `run()`

D) `start()`

Answer: A

□ Testing

220. How can you test views in Django?

A) Using Django's `Client` class to simulate requests in tests

B) Using raw HTTP requests only

C) Using Selenium only

D) Views cannot be tested

Answer: A

221. What is the difference between `TestCase` and `TransactionTestCase`?

A) `TestCase` wraps each test in a transaction and rolls back, `TransactionTestCase` does not and is slower but supports testing transactions explicitly

B) No difference

C) `TransactionTestCase` is deprecated

D) `TestCase` is only for API tests

Answer: A

■ □ Class-based View Mixins

222. Which mixin would you use to require a user to be logged in to access a CBV?

A) `LoginRequiredMixin`

B) `UserPassesTestMixin`

C) `PermissionRequiredMixin`

D) `AccessMixin`

Answer: A

223. How do you add permission checks in a CBV?

A) Use `PermissionRequiredMixin` and set `permission_required` attribute

B) Use `@permission_required` decorator only

C) Use middleware only

D) Override `get()` method only

Answer: A

■ Forms and Validation

224. Which method on a form instance is used to check if the data is valid?

- A) `is_valid()`
- B) `clean()`
- C) `validate()`
- D) `check()`

Answer: A

225. How do you customize error messages for a form field?

- A) Pass `error_messages` dictionary when defining the form field
- B) Override `__init__` only
- C) Change the template only
- D) Use middleware

Answer: A

Migrations

226. What command applies migrations to the database?

- A) `python manage.py migrate`
- B) `python manage.py makemigrations`
- C) `python manage.py syncdb`
- D) `python manage.py flush`

Answer: A

227. How can you create empty migrations for manual operations?

- A) `python manage.py makemigrations --empty appname`
- B) `makemigrations --manual`
- C) Create migrations manually in filesystem only
- D) Not supported

Answer: A

Internationalization and Localization

228. How do you mark a template string for translation?

- A) Use `{% trans "string" %}` tag
- B) Use `{{ _("string") }}` only
- C) Use `gettext()` only
- D) Use `{{ trans("string") }}`

Answer: A

229. Which setting controls the current language in Django?

- A) `LANGUAGE_CODE`
- B) `LOCALE`
- C) `LANGUAGE`
- D) `TIME_ZONE`

Answer: A

⚡ Async and Channels

230. Which Django component adds support for WebSockets and async protocols?

- A) Django Channels
- B) Django REST Framework
- C) Django Middleware
- D) Django ORM

Answer: A

231. Which protocol does Channels use under the hood for async communication?

- A) ASGI (Asynchronous Server Gateway Interface)
- B) WSGI
- C) HTTP/1.1 only
- D) FTP

Answer: A

🔒 Security

232. Which middleware helps protect against clickjacking?

- A) XFrameOptionsMiddleware
- B) CsrfViewMiddleware
- C) SecurityMiddleware
- D) SessionMiddleware

Answer: A

233. How do you enforce HTTPS in Django?

- A) Set `SECURE_SSL_REDIRECT = True` and configure web server accordingly
- B) Use middleware only
- C) Use raw SQL
- D) Not possible

Answer: A

Would you like me to continue? I can keep going till 300 or prepare PDFs for these batches too!

You said:

yes

ChatGPT said:

Great! Here's the next batch of **Django advanced MCQs (251–300)** to keep the streak alive:

🌀 Django Advanced MCQs (251–300)

🔧 Advanced Model Features

251. **What does the `unique_together` Meta option enforce?**
A) Composite uniqueness constraint across multiple fields
B) Uniqueness of a single field
C) Foreign key constraints
D) Index creation only
Answer: A
252. **How do you specify a database index on a model field?**
A) Use `db_index=True` on the field
B) Use `index=True`
C) Use `unique=True`
D) Use `primary_key=True`
Answer: A
253. **What does the `get_or_create()` method do?**
A) Returns an object matching the lookup parameters or creates it if it doesn't exist
B) Only gets the object, throws exception if not found
C) Only creates a new object
D) Deletes then creates object
Answer: A
-

💰 Transactions and Atomic Operations

254. **How do you run multiple database operations atomically in Django?**
A) Use `transaction.atomic()` context manager or decorator
B) Use raw SQL only
C) Use signals
D) Django does not support atomic transactions
Answer: A
255. **What happens if an exception is raised inside a `transaction.atomic()` block?**
A) The transaction is rolled back
B) The transaction commits anyway
C) Nothing
D) Database crashes
Answer: A
-

🔗 Custom Template Tags and Filters

256. **Where do you register custom template tags and filters?**

- A) Inside a `templatetags` package within a Django app
- B) In `views.py`
- C) In `urls.py`
- D) In `settings.py`

Answer: A

257. **How do you use a custom template filter in a template?**

- A) Load the tag library with `{% load your_tag_library %}` and then use `{{ value|filter_name }}`
- B) Use `{% filter %}` only
- C) Use `{% include %}`
- D) Use raw Python code in template

Answer: A

☐ Signals (More Details)

258. **Which signal fires immediately before a model's `save()` method?**

- A) `pre_save`
- B) `post_save`
- C) `pre_delete`
- D) `post_delete`

Answer: A

259. **Can you disconnect a signal receiver?**

- A) Yes, using `signal.disconnect(receiver)` method
- B) No, once connected it cannot be disconnected
- C) Only via settings
- D) Only in Django admin

Answer: A

☒ Advanced Authentication

260. **How do you implement custom authentication backends?**

- A) Subclass `BaseBackend` and implement `authenticate()` and `get_user()` methods
- B) Modify `settings.py` only
- C) Use middleware only
- D) Override User model only

Answer: A

261. **Where do you configure your authentication backends?**

- A) In `AUTHENTICATION_BACKENDS` setting
- B) In middleware settings
- C) In `INSTALLED_APPS`
- D) In `urls.py`

Answer: A

□ URL Dispatching and Reverse Lookups

262. **What does the `reverse()` function do?**

- A) Returns the URL string for a given view name and parameters
- B) Redirects to a URL
- C) Resolves URLs from the browser only
- D) Generates static files path

Answer: A

263. **What is the difference between `reverse()` and `redirect()`?**

- A) `reverse()` returns a URL string, `redirect()` returns an HTTP redirect response
- B) They are the same
- C) `redirect()` only for external URLs
- D) `reverse()` does not accept parameters

Answer: A

□ Testing (More Advanced)

264. **How do you test Django signals?**

- A) Connect a test receiver during tests and assert it was called
- B) Signals cannot be tested
- C) Test via middleware only
- D) By mocking database only

Answer: A

265. **How do you mock external APIs in Django tests?**

- A) Use libraries like `unittest.mock` or `responses` to patch HTTP calls
- B) Use raw HTTP requests
- C) Django does not support mocking
- D) Use middleware only

Answer: A

□ Advanced Class-based Views

266. **Which method handles form submission in `FormView`?**

- A) `form_valid()`
- B) `post()`
- C) `get()`
- D) `save()`

Answer: A

267. **How do you redirect after successful form submission in CBVs?**

- A) Define `get_success_url()` method or set `success_url` attribute
- B) Override `post()` method only
- C) Use `redirect()` in template

D) Use middleware

Answer: A

🔍 Debugging and Logging

268. **How can you configure Django logging?**

- A) Use the `LOGGING` dictionary in settings to define handlers, formatters, and loggers
- B) Use middleware only
- C) Use environment variables only
- D) Django does not support logging

Answer: A

269. **Which logging level indicates detailed debug information?**

- A) `DEBUG`
- B) `INFO`
- C) `WARNING`
- D) `ERROR`

Answer: A

📦 Django REST Framework (DRF) Advanced

270. **What is a `SerializerMethodField` used for in DRF?**

- A) To include a read-only field whose value is computed by a method on the serializer
- B) To define writable fields only
- C) To link related serializers only
- D) To paginate results

Answer: A

271. **How do you implement token authentication in DRF?**

- A) Add `TokenAuthentication` to `DEFAULT_AUTHENTICATION_CLASSES` and use `rest_framework.authtoken` app
- B) Use session only
- C) Use middleware only
- D) DRF does not support token authentication

Answer: A

⚡ Asynchronous Programming

272. **How do you define an async view in Django?**

- A) Define the view function with `async def` syntax
- B) Use `@async_view` decorator only
- C) Use `sync_to_async` everywhere
- D) Async views are not supported

Answer: A

273. **What is `sync_to_async` used for?**
- A) To run synchronous code asynchronously in async contexts
 - B) To sync databases
 - C) To convert async views to sync
 - D) To connect signals asynchronously
- Answer: A**
-

Middleware

274. **What must middleware classes implement in Django 3.x+?**
- A) Be callable with signature `(self, request, get_response)`
 - B) Implement `process_request` only
 - C) Implement `process_response` only
 - D) Implement `__call__` with no parameters
- Answer: A**
275. **How do you stop processing a request in middleware?**
- A) Return an `HttpResponse` object early from the middleware callable
 - B) Raise exception
 - C) Call `next()`
 - D) Use signals
- Answer: A**
-

Would you like me to continue with even more MCQs or prepare these in a downloadable PDF? Just let me know!

You said:

more

ChatGPT said:

Failed to generate

To continue creating and organizing more MCQs using **ChatGPT canvas**, you'll need to [log in to ChatGPT](#). Once logged in, you can use the canvas feature to collaborate and structure content like quizzes or documents efficiently.

Would you like me to continue listing more Django MCQs here directly in this chat for now?

You said:

more

ChatGPT said:

Excellent! Here's the next batch of **Django Advanced MCQs (301–350)** — perfect for deep-level interviews, certifications, or mastery:

🌀 Django Advanced MCQs (301–350)

❑ Model Inheritance and Polymorphism

301. Which inheritance style allows you to create multiple models from a common abstract base class without creating a database table for it?

- A) Abstract base classes (`abstract = True`)
- B) Multi-table inheritance
- C) Proxy models
- D) Joined inheritance

Answer: A

302. What is the key feature of multi-table inheritance in Django?

- A) Each subclass gets its own table and links to the parent with `OneToOneField`
- B) No database table is created
- C) No relationship is maintained
- D) The parent is ignored

Answer: A

303. Which of the following creates a model that uses the same database table as its parent?

- A) Proxy models
- B) Abstract base class
- C) Multi-table inheritance
- D) Raw SQL

Answer: A

❑ Forms (Advanced)

304. What is `form.instance` used for in Django forms?

- A) To access the model instance bound to a `ModelForm`
- B) To set the form's template
- C) To track form submissions
- D) To bind user info

Answer: A

305. How do you dynamically change form fields in Django?

- A) Modify `self.fields` in the form's `__init__()` method
- B) Use `cleaned_data`
- C) Use middleware
- D) Forms can't be changed dynamically

Answer: A

🔗 Advanced Views and Mixins

306. Which mixin is typically used to enforce permissions in Django CBVs?

- A) `PermissionRequiredMixin`
- B) `LoginRequiredMixin`
- C) `SuperUserMixin`
- D) `FormMixin`

Answer: A

307. In a `ListView`, which context variable contains the list of objects?

- A) `object_list`
- B) `results`
- C) `queryset`
- D) `items`

Answer: A

308. How do you change the default context object name in a `DetailView`?

- A) Set `context_object_name` attribute
- B) Set `template_name`
- C) Override `get()`
- D) Override `get_queryset()`

Answer: A

🔗 Permissions and Groups

309. How do you assign a permission to a group in Django?

- A) Add a `Permission` object to `group.permissions`
- B) Use signals
- C) Override the `save()` method
- D) It happens automatically

Answer: A

310. Which method checks if a user has a specific permission?

- A) `user.has_perm('app_label.permission_codename')`
- B) `user.check_perm()`
- C) `user.get_perm()`
- D) `user.auth()`

Answer: A

🔗 Optimization & Query Debugging

311. How can you view SQL queries executed by a Django view?

- A) Use `django.db.connection.queries` in debug mode
- B) Use `SELECT *` in raw SQL
- C) Only with external profiler

D) Not possible

Answer: A

312. **What does `only()` do in a queryset?**

A) Fetches only the listed fields, deferring the others

B) Restricts filtering

C) Deletes unrelated fields

D) Sorts the result

Answer: A

☐ Advanced Testing (Continued)

313. **What does `self.assertContains(response, text)` do in tests?**

A) Checks if `text` appears in `response.content`

B) Checks DB state

C) Tests if model saved

D) Skips test

Answer: A

314. **Which test class should you use to test raw SQL or transactions?**

A) `TransactionTestCase`

B) `TestCase`

C) `SimpleTestCase`

D) `ClientTestCase`

Answer: A

☐ Admin Customization

315. **How can you customize the admin list page display?**

A) Use `list_display` in your `ModelAdmin` class

B) Override `save()`

C) Use middleware

D) Edit `settings.py`

Answer: A

316. **How do you add filters in Django admin sidebar?**

A) Use `list_filter` attribute

B) Use `filter_fields`

C) Use `filter_sidebar`

D) Override `get()`

Answer: A

317. **How do you make a field editable in the list view?**

A) Add it to `list_editable` in the `ModelAdmin`

B) Make it readonly

C) Use `form_class`

D) Not possible

Answer: A

□ DRF: Permissions and Throttling

318. **What does `IsAuthenticatedOrReadOnly` permission class do?**

- A) Allows full access to authenticated users, read-only for others
- B) Requires superuser
- C) Denies all access
- D) Allows anonymous write

Answer: A

319. **Which throttle class limits requests per user/IP per minute?**

- A) `UserRateThrottle` or `AnonRateThrottle`
- B) `DailyThrottle`
- C) `MaxThrottle`
- D) `TimedThrottle`

Answer: A

⚡ Async Support in Views and Tasks

320. **Which library is typically used to schedule background tasks in Django?**

- A) Celery
- B) DRF
- C) Admin
- D) Django ORM

Answer: A

321. **What is the role of a message broker in Django-Celery?**

- A) Queues and dispatches tasks (e.g., via Redis or RabbitMQ)
- B) Connects to templates
- C) Hosts the frontend
- D) Manages settings

Answer: A

□ Model Meta Options (Advanced)

322. **What does `ordering = ['-created_at']` do in a model's Meta class?**

- A) Orders querysets by `created_at` descending by default
- B) Makes fields editable
- C) Changes database schema
- D) Does nothing

Answer: A

323. **How do you make a model's plural name correct in the admin?**

- A) Set `verbose_name_plural` in Meta class
- B) Use `app_label`
- C) Edit `settings.py`

D) Use `model_plural_name`

Answer: A

☐ Miscellaneous

324. Which file is responsible for defining application-specific config?

- A) `apps.py`
- B) `models.py`
- C) `views.py`
- D) `admin.py`

Answer: A

325. What does `AppConfig.ready()` allow you to do?

- A) Perform startup logic such as connecting signals
- B) Register URL patterns
- C) Render templates
- D) Modify database schema

Answer: A

☐ Custom User Models

326. When should you define a custom user model in Django?

- A) Before the first migration if custom fields or behaviors are needed
- B) Anytime
- C) Only in admin
- D) Only for APIs

Answer: A

327. Which method must a custom user model implement?

- A) `USERNAME_FIELD`, `is_active`, `create_user()`
- B) `save()`
- C) `update()`
- D) `delete()`

Answer: A