

1. General Concepts

1. **What is Django REST Framework (DRF)?**

- A) A web server for Django
- B) A toolkit for building Web APIs in Django
- C) A database management system
- D) A templating engine for Django

Answer: B) A toolkit for building Web APIs in Django

2. **Which of the following is NOT a feature of DRF?**

- A) Serialization
- B) Authentication
- C) Middleware
- D) ViewSets

Answer: C) Middleware

3. **Which class is used to create a serializer in DRF?**

- A) serializers.ModelSerializer
- B) serializers.Serializer
- C) serializers.BaseSerializer
- D) serializers.ModelViewSet

Answer: B) serializers.Serializer

4. **What does the `ModelSerializer` class do in DRF?**

- A) Validates data
- B) Converts data to JSON
- C) Automatically generates fields based on a Django model
- D) Handles HTTP requests

Answer: C) Automatically generates fields based on a Django model

5. **Which method is used to handle GET requests in a DRF view?**

- A) `get()`
- B) `retrieve()`
- C) `list()`
- D) `get_queryset()`

Answer: A) `get()`

6. **What is the purpose of the `APIView` class in DRF?**

- A) To define a view that handles HTTP methods
- B) To serialize data
- C) To authenticate users
- D) To handle database queries

Answer: A) To define a view that handles HTTP methods

7. **Which of the following is used to handle authentication in DRF?**

- A) Authentication classes
- B) Permission classes
- C) ViewSets
- D) Serializers

Answer: A) Authentication classes

8. **What is the default authentication class in DRF?**

- A) SessionAuthentication
- B) BasicAuthentication
- C) TokenAuthentication
- D) JWTAuthentication

Answer: A) SessionAuthentication

9. **Which class is used to implement token-based authentication in DRF?**

- A) TokenAuthentication
- B) BasicAuthentication
- C) SessionAuthentication
- D) JWTAuthentication

Answer: A) TokenAuthentication

10. **What is the purpose of permission classes in DRF?**

- A) To authenticate users
- B) To define access control for views
- C) To serialize data
- D) To handle HTTP requests

Answer: B) To define access control for views

2. Serializers

11. **Which method is used to validate incoming data in a serializer?**

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

Answer: B) is_valid()

12. **How do you specify the model for a `ModelSerializer`?**

- A) `model = ModelName`
- B) `class Meta: model = ModelName`
- C) `model_name = ModelName`
- D) `meta_model = ModelName`

Answer: B) class Meta: model = ModelName

13. Which field type is used for a foreign key relationship in DRF serializers?

- A) ForeignKeyField
- B) RelatedField
- C) PrimaryKeyRelatedField
- D) ForeignKey

Answer: C) PrimaryKeyRelatedField

14. What does the `read_only` attribute do in a serializer field?

- A) Makes the field writable
- B) Makes the field read-only
- C) Makes the field required
- D) Makes the field optional

Answer: B) Makes the field read-only

15. Which method is used to create a new object from validated data in a serializer?

- A) create()
- B) save()
- C) build()
- D) new()

Answer: A) create()

16. How do you specify the fields to be included in a `ModelSerializer`?

- A) fields = 'all'
- B) fields = ['field1', 'field2']
- C) fields = True
- D) fields = None

Answer: B) fields = ['field1', 'field2']

17. Which method is used to update an existing object in a serializer?

- A) update()
- B) save()
- C) modify()
- D) change()

Answer: A) update()

18. What is the purpose of the `validate_<field_name>` method in a serializer?

- A) To validate the entire serializer
- B) To validate individual fields
- C) To serialize data
- D) To handle HTTP requests

Answer: B) To validate individual fields

19. Which of the following is a built-in field type in DRF serializers?

- A) CharField
- B) DateTimeField
- C) IntegerField
- D) All of the above

Answer: D) All of the above

20. How do you handle nested serializers in DRF?

- A) By using the `NestedSerializer` class
- B) By using the `Serializer` class
- C) By including a serializer as a field in another serializer
- D) By using the `NestedField` class

Answer: C) By including a serializer as a field in another serializer

3. Views and ViewSets

21. Which class is used to create a view that handles HTTP methods in DRF?

- A) `APIView`
- B) `ViewSet`
- C) `ModelViewSet`
- D) `GenericAPIView`

Answer: A) `APIView`

22. What is the purpose of the `ModelViewSet` class in DRF?

- A) To define a view that handles HTTP methods
- B) To define a view that handles CRUD operations for a model
- C) To serialize data
- D) To handle database queries

Answer: B) To define a view that handles CRUD operations for a model

23. Which method is used to retrieve a single object in a view?

- A) `get()`
- B) `retrieve()`
- C) `list()`
- D) `get_object()`

Answer: B) `retrieve()`

24. Which method is used to list multiple objects in a view?

- A) `get()`
- B) `retrieve()`
- C) `list()`
- D) `get_queryset()`

Answer: C) list()

25. **What is the purpose

Sources

You said:

?

ChatGPT said:

Certainly! Continuing from the previous set, here are additional multiple-choice questions (MCQs) related to Django REST Framework (DRF), focusing on views, authentication, permissions, pagination, and testing.

3. Views and ViewSets (Continued)

26. Which class in DRF provides default implementations for CRUD operations?

- A) APIView
- B) ViewSet
- C) ModelViewSet
- D) GenericAPIView

Answer: C) ModelViewSet

27. What is the purpose of the `get_queryset()` method in a DRF view?

- A) To retrieve the current request object
- B) To define the queryset for the view
- C) To serialize the data
- D) To handle HTTP methods

Answer: B) To define the queryset for the view

28. Which decorator is used to create function-based views in DRF?

- A) `@api_view`
- B) `@view_set`
- C) `@model_view`
- D) `@serializer_view`

Answer: A) `@api_view`

29. How do you handle different HTTP methods in a function-based API view?

- A) By defining separate functions for each method
- B) By using conditional statements based on `request.method`
- C) By using the `@method_decorator`
- D) By defining a `dispatch()` method

Answer: B) By using conditional statements based on `request.method`

30. Which of the following is NOT a valid HTTP method in DRF?

- A) GET
- B) POST
- C) PUT
- D) PATCH

Answer: D) PATCH

4. Authentication and Permissions

31. Which authentication class is used for token-based authentication in DRF?

- A) SessionAuthentication
- B) BasicAuthentication
- C) TokenAuthentication
- D) JWTAuthentication

Answer: C) TokenAuthentication

32. Which of the following is a built-in permission class in DRF?

- A) IsAuthenticated
- B) IsAdminUser
- C) IsAuthenticatedOrReadOnly
- D) All of the above

Answer: D) All of the above

33. How do you apply permissions to a view in DRF?

- A) By setting the `permission_classes` attribute
- B) By using the `@permission_classes` decorator
- C) By overriding the `has_permission()` method
- D) By configuring permissions in `settings.py`

Answer: A) By setting the `permission_classes` attribute

34. Which of the following is NOT a valid permission class in DRF?

- A) IsAuthenticated
- B) IsAdminUser
- C) IsAuthenticatedOrReadOnly
- D) IsSuperUser

Answer: D) IsSuperUser

35. What is the purpose of the `IsAuthenticated` permission class in DRF?

- A) To allow access to authenticated users only
- B) To allow access to all users

- C) To restrict access to unauthenticated users
- D) To allow access based on user roles

Answer: A) To allow access to authenticated users only

5. Pagination

36. Which pagination class is used for page number-based pagination in DRF?

- A) PageNumberPagination
- B) LimitOffsetPagination
- C) CursorPagination
- D) None of the above

Answer: A) PageNumberPagination

37. How do you set the default pagination class in DRF?

- A) By setting `DEFAULT_PAGINATION_CLASS` in `settings.py`
- B) By using the `pagination_class` attribute in views
- C) By configuring pagination in `urls.py`
- D) By using the `@pagination_class` decorator

Answer: A) By setting `DEFAULT_PAGINATION_CLASS` in `settings.py`

38. Which of the following is NOT a valid pagination class in DRF?

- A) PageNumberPagination
- B) LimitOffsetPagination
- C) CursorPagination
- D) RangePagination

Answer: D) RangePagination

39. How do you specify the page size in DRF pagination?

- A) By setting `PAGE_SIZE` in `settings.py`
- B) By using the `page_size` attribute in views
- C) By passing `page_size` as a query parameter
- D) By configuring pagination in `urls.py`

Answer: A) By setting `PAGE_SIZE` in `settings.py`

40. Which pagination class provides cursor-based pagination in DRF?

- A) PageNumberPagination
- B) LimitOffsetPagination
- C) CursorPagination
- D) None of the above

Answer: C) CursorPagination

6. Testing

41. Which class is used for testing DRF views in Django?

- A) APITestCase
- B) APIClient
- C) RequestsClient
- D) TestCase

Answer: A) APITestCase

42. How do you make a GET request in DRF tests?

- A) `client.get()`
- B) `client.post()`
- C) `client.put()`
- D) `client.delete()`

Answer: A) `client.get()`

43. Which method is used to assert the status code of a response in DRF tests?

- A) `assertEqual()`
- B) `assertStatusCode()`
- C) `assertResponseCode()`
- D) `assertEqualStatusCode()`

Answer: A) `assertEqual()`

44. How do you authenticate a user in DRF tests?

- A) By using the `force_authenticate()` method
- B) By setting the `Authorization` header
- C) By using the `authenticate()` method
- D) By logging in through the admin panel

Answer: A) By using the `force_authenticate()` method

45. Which class is used to make HTTP requests in DRF tests?

- A) APIClient
- B) RequestsClient
- C) TestClient
- D) HttpClient

Answer: A) APIClient

7. Advanced Topics

46. Which package is used to implement JWT authentication in DRF?

- A) djangorestframework-jwt
- B) djangorestframework-simplejwt
- C) django-jwt
- D) django-rest-jwt

Answer: B) djangorestframework-simplejwt

47. How do you include JWT authentication in DRF settings?

- A) By adding `'DEFAULT_AUTHENTICATION_CLASSES': ['rest_framework_simplejwt.authentication.JWTAuthentication']`

Sources

You said:

more from 47

ChatGPT said:

Absolutely! Continuing from question 47, here are more advanced MCQs on **JWT Authentication, Routers, Filtering, Throttling, and Versioning** in Django REST Framework.

7. Advanced Topics (continued)

47. How do you include JWT authentication in DRF settings?

- A) `'DEFAULT_AUTHENTICATION_CLASSES': ['rest_framework_simplejwt.authentication.JWTAuthentication']`
- B) `'AUTH_CLASSES': ['jwt.authentication']`
- C) `'DEFAULT_AUTHENTICATION_CLASSES': ['rest_framework.authentication.JWTAuth']`
- D) `'AUTHENTICATION_BACKENDS': ['JWTAuth']`

Answer: A) `'DEFAULT_AUTHENTICATION_CLASSES': ['rest_framework_simplejwt.authentication.JWTAuthentication']`

48. What is the default lifespan of an access token in Simple JWT?

- A) 5 minutes
- B) 30 minutes
- C) 1 hour
- D) 24 hours

Answer: A) 5 minutes

49. Which command is used to obtain a JWT token pair in Simple JWT?

- A) `/api/token/`
- B) `/auth/jwt/`
- C) `/api/login/`
- D) `/token/`

Answer: A) `/api/token/`

50. Which endpoint is used to refresh a JWT token?

- A) `/api/token/refresh/`
- B) `/token/refresh/`
- C) `/auth/token/refresh/`
- D) `/api/refresh/`

Answer: A) `/api/token/refresh/`

8. Routers and URLs

51. What is the primary purpose of routers in DRF?

- A) To define URL patterns for viewsets automatically
- B) To create database models
- C) To authenticate users
- D) To serialize data

Answer: A) To define URL patterns for viewsets automatically

52. Which router class is used by default in DRF?

- A) `SimpleRouter`
- B) `DefaultRouter`
- C) `AutoRouter`
- D) `ModelRouter`

Answer: B) `DefaultRouter`

53. What's the difference between `DefaultRouter` and `SimpleRouter`?

- A) `DefaultRouter` adds a default root view
- B) `SimpleRouter` supports nested routes
- C) `SimpleRouter` is more configurable
- D) No difference

Answer: A) `DefaultRouter` adds a default root view

54. How do you register a viewset with a router?

- A) `router.register(r'endpoint', ViewSetClass)`
- B) `router.map(r'endpoint', ViewSetClass)`
- C) `urlpatterns.add(ViewSetClass)`
- D) `router.bind(ViewSetClass)`

Answer: A) `router.register(r'endpoint', ViewSetClass)`

55. Which of these is required to use routers in `urls.py`?

- A) `include(router.urls)`
- B) `url(router.urls)`
- C) `path(router)`
- D) `add(router.views)`

Answer: A) `include(router.urls)`

9. Filtering and Search

56. Which package is commonly used for advanced filtering in DRF?

- A) `django-filter`
- B) `rest-filter`
- C) `drf-filters`
- D) `drf-query`

Answer: A) `django-filter`

57. Which setting is used to enable filtering globally in DRF?

- A) `'DEFAULT_FILTER_BACKENDS'`
- B) `'FILTER_CLASSES'`
- C) `'FILTER_SETTINGS'`
- D) `'DEFAULT_FILTER_CLASSES'`

Answer: A) `'DEFAULT_FILTER_BACKENDS'`

58. How do you define filterable fields in a viewset using `django-filter`?

- A) `filterset_fields = ['field1', 'field2']`
- B) `filter_fields = ['field1', 'field2']`
- C) `filters = ['field1', 'field2']`
- D) `filterable = ['field1', 'field2']`

Answer: A) `filterset_fields = ['field1', 'field2']`

59. Which class provides search functionality in DRF?

- A) SearchFilter
- B) QueryFilter
- C) TextFilter
- D) RegexFilter

Answer: A) SearchFilter

60. How do you define searchable fields in a DRF view?

- A) `search_fields = ['name', 'description']`
- B) `filterset_fields = ['name', 'description']`
- C) `query_fields = ['name', 'description']`
- D) `text_fields = ['name', 'description']`

Answer: A) `search_fields = ['name', 'description']`

10. Throttling

61. What is throttling used for in DRF?

- A) To control the rate of API requests
- B) To encrypt data
- C) To increase speed of response
- D) To validate requests

Answer: A) To control the rate of API requests

62. Which class provides anonymous user throttling?

- A) AnonRateThrottle
- B) BasicRateThrottle
- C) AnonymousThrottle
- D) RateLimitAnon

Answer: A) AnonRateThrottle

63. Where are throttling rates defined in DRF?

- A) In `settings.py` under `THROTTLE_RATES`
- B) In the views
- C) In the model
- D) In the serializer

Answer: A) In `settings.py` under `THROTTLE_RATES`

64. What is the default throttle rate format in settings?

- A) '100/day'
- B) 'per_day:100'
- C) 'daily:100'
- D) '100_daily'

Answer: A) '100/day'

65. Which throttle class is used for authenticated users?

- A) UserRateThrottle
- B) AuthThrottle
- C) LoginThrottle
- D) SessionThrottle

Answer: A) UserRateThrottle

11. Versioning

66. What is the purpose of API versioning in DRF?

- A) To maintain multiple versions of an API
- B) To limit requests
- C) To paginate results
- D) To authorize users

Answer: A) To maintain multiple versions of an API

67. Which setting enables versioning in DRF?

- A) DEFAULT_VERSIONING_CLASS
- B) API_VERSION
- C) VERSION_CLASSES
- D) ENABLE_VERSIONING

Answer: A) DEFAULT_VERSIONING_CLASS

68. Which of the following is NOT a versioning class in DRF?

- A) URLPathVersioning
- B) NamespaceVersioning
- C) HeaderVersioning
- D) RouteVersioning

Answer: D) RouteVersioning

69. How is the version specified in URLPathVersioning?

- A) Part of the URL path (e.g., /v1/endpoint/)
- B) Part of the query parameter
- C) As a header
- D) As a form field

Answer: A) Part of the URL path (e.g., /v1/endpoint/)

70. How is the version accessed in a view?

- A) `request.version`
- B) `request.api_version`
- C) `request.VER`
- D) `request['version']`

Answer: A) `request.version`

Would you like me to continue and generate the rest of the 100 MCQs (71–100) on topics like **Permissions, Customizing DRF, Best Practices, or Deployment?**

You said:

more

ChatGPT said:

Great! Let's continue with the final batch of **30 multiple-choice questions (71–100)** related to Django REST Framework (DRF), covering topics like **Custom Permissions, Custom Serializers, Testing, Best Practices, and Deployment.**

12. Custom Permissions

71. How do you create a custom permission in DRF?

- A) By subclassing `BasePermission` and overriding `has_permission()`
- B) By subclassing `APIView`
- C) By creating a new mixin
- D) By modifying `settings.py`

Answer: A) By subclassing `BasePermission` and overriding `has_permission()`

72. What does the `has_object_permission()` method in a permission class do?

- A) Checks permission at the object level
- B) Grants global access
- C) Validates API keys
- D) None of the above

Answer: A) Checks permission at the object level

73. Where should custom permission classes be applied in DRF?

- A) On the `permission_classes` attribute of views or viewsets
- B) In the model's Meta class
- C) Inside serializers
- D) In the database schema

Answer: A) On the `permission_classes` attribute of views or viewsets

13. Custom Serializers and Fields

74. How do you define a custom serializer field in DRF?

- A) Subclass `serializers.Field` and override `to_representation` and `to_internal_value`
- B) Override `validate()` method only
- C) Create a new model
- D) Subclass `FieldSerializer`

Answer: A) Subclass `serializers.Field` and override `to_representation` and `to_internal_value`

75. What is the purpose of the `to_representation()` method?

- A) Converts Python objects to JSON
- B) Validates data
- C) Encrypts data
- D) Saves data to the database

Answer: A) Converts Python objects to JSON

76. What is the purpose of `to_internal_value()` method?

- A) Parses input data into Python objects
- B) Serializes data
- C) Stores data in cache
- D) Applies throttling

Answer: A) Parses input data into Python objects

14. Testing APIs

77. Which of the following is used for making test requests in DRF?

- A) `APIClient`
- B) `HttpClient`
- C) `Client`
- D) `BrowserClient`

Answer: A) `APIClient`

78. What does the `force_authenticate()` method do in testing?

- A) Authenticates requests without login
- B) Forces token refresh
- C) Tests invalid credentials
- D) Logs out users

Answer: A) Authenticates requests without login

79. What method is used to check JSON response content in tests?

- A) `response.json()`
- B) `response.get_json()`
- C) `response.content()`
- D) `response.body`

Answer: A) `response.json()`

80. Which method is used to test DELETE requests in DRF?

- A) `client.delete()`
- B) `client.remove()`
- C) `client.clear()`
- D) `client.truncate()`

Answer: A) `client.delete()`

15. Best Practices

81. What is the best way to structure large DRF projects?

- A) Use Django apps for separation of concerns
- B) Keep all logic in a single file
- C) Avoid serializers
- D) Use hardcoded views

Answer: A) Use Django apps for separation of concerns

82. Which of these is a recommended security best practice for APIs?

- A) Use HTTPS
- B) Disable authentication
- C) Allow all CORS origins
- D) Log sensitive data

Answer: A) Use HTTPS

83. Why should you prefer `ModelViewSet` over separate generic views for simple CRUD APIs?

- A) Less boilerplate code
- B) More customizable
- C) Forces validation
- D) Improves pagination

Answer: A) Less boilerplate code

84. What should you avoid including in API responses?

- A) Sensitive user information
- B) Status codes
- C) Pagination metadata
- D) HTTP headers

Answer: A) Sensitive user information

85. Which HTTP status code means 'Created'?

- A) 201
- B) 200
- C) 204
- D) 404

Answer: A) 201

16. API Documentation

86. Which library is commonly used to auto-generate API docs in DRF?

- A) drf-yasg
- B) drf-docs
- C) django-autodoc
- D) rest-docgen

Answer: A) drf-yasg

87. Which endpoint is typically used to serve Swagger UI with drf-yasg?

- A) /swagger/
- B) /api/docs/
- C) /docs/swagger/
- D) /swagger-ui/

Answer: A) /swagger/

88. Which setting is needed to use OpenAPI documentation?

- A) Add `drf_yasg` to `INSTALLED_APPS`
- B) Enable `OPENAPI=True`
- C) Set `API_DOCS` in `urls.py`
- D) Use `api_docs=True`

Answer: A) Add `drf_yasg` to `INSTALLED_APPS`

17. Deployment

89. Which command collects static files before deployment?

- A) `python manage.py collectstatic`
- B) `python manage.py deploy`
- C) `python deploy.py`
- D) `python manage.py build`

Answer: A) `python manage.py collectstatic`

90. Which WSGI servers are commonly used to serve DRF apps in production?

- A) Gunicorn or uWSGI
- B) Apache only
- C) SQLite server
- D) Flask internal server

Answer: A) Gunicorn or uWSGI

91. Which environment variable is used to set the Django settings module in production?

- A) `DJANGO_SETTINGS_MODULE`
- B) `SETTINGS_DJANGO`
- C) `DJANGO_CONF`
- D) `DJANGO_MODULE`

Answer: A) `DJANGO_SETTINGS_MODULE`

92. Which tool is used to reverse proxy DRF apps in deployment?

- A) Nginx
- B) SQLite
- C) Django Admin
- D) Redis

Answer: A) Nginx

93. What database is typically NOT recommended for production?

- A) SQLite
- B) PostgreSQL
- C) MySQL
- D) Oracle

Answer: A) SQLite

18. Miscellaneous

94. Which format does DRF use by default for API responses?

- A) JSON
- B) XML
- C) YAML
- D) CSV

Answer: A) JSON

95. What is `BrowsableAPIRenderer` used for?

- A) To provide a web interface to test APIs
- B) To convert data to XML
- C) To allow JWT refresh
- D) To throttle requests

Answer: A) To provide a web interface to test APIs

96. Which renderer class is used for JSON responses?

- A) `JSONRenderer`
- B) `TextRenderer`
- C) `TemplateRenderer`
- D) `YamlRenderer`

Answer: A) `JSONRenderer`

97. How can you disable the Browsable API interface?

- A) Remove `BrowsableAPIRenderer` from `DEFAULT_RENDERER_CLASSES`
- B) Delete `admin.py`
- C) Use `APIView.disable_ui = True`
- D) Set `DEBUG=False`

Answer: A) Remove `BrowsableAPIRenderer` from `DEFAULT_RENDERER_CLASSES`

98. Which of these is an HTTP status code for unauthorized access?

- A) 401
- B) 403
- C) 404
- D) 200

Answer: A) 401

99. Which status code is returned when a resource is successfully deleted?

- A) 204 No Content
- B) 200 OK
- C) 202 Accepted
- D) 500 Internal Server Error

Answer: A) 204 No Content

100. Which HTTP status code is used for a bad request in DRF?

- A) 400
- B) 401
- C) 403
- D) 500

Answer: A) 400

✓ That's **100 multiple choice questions** on Django REST Framework (DRF), complete with topics ranging from beginner to advanced.

Would you like this in a downloadable format (like PDF or Excel), or want the answers in a separate list for quiz purposes?

You said:

more

ChatGPT said:

You've now received a full set of **100 MCQs** on Django REST Framework, which covers nearly every critical topic including:

- General DRF usage
- Serializers
- Views & ViewSets
- Authentication & Permissions
- Routers & URLs
- Pagination
- Filtering & Search
- Throttling
- Versioning
- Testing
- Best Practices
- API Documentation
- Deployment
- Miscellaneous edge cases

Since you're asking for **more**, here are **10 bonus advanced MCQs** (101–110) on **signals, performance optimization, async views, and advanced customization** in DRF.

Bonus Advanced DRF MCQs (101–110)

101. What is the primary use of Django signals in a DRF project?

- A) Trigger functions on database events like save or delete
- B) Replace serializers
- C) Handle HTTP requests
- D) Optimize API performance

Answer: A) Trigger functions on database events like save or delete

102. Which signal would you typically connect to when monitoring object creation?

- A) `post_save`
- B) `pre_delete`
- C) `object_created`
- D) `post_request`

Answer: A) `post_save`

103. How can you reduce database queries in DRF when accessing related models?

- A) Use `.select_related()` or `.prefetch_related()` in the queryset

- B) Use `raw()`
- C) Use `exclude()`
- D) Use `order_by()`

Answer: A) Use `.select_related()` or `.prefetch_related()` in the queryset

104. Which setting can help optimize DRF for large responses?

- A) `DEFAULT_PARSER_CLASSES`
- B) `DEFAULT_RENDERER_CLASSES`
- C) `DEFAULT_THROTTLE_CLASSES`
- D) `TIME_ZONE`

Answer: B) `DEFAULT_RENDERER_CLASSES`
(Using only `JSONRenderer` in production is more efficient.)

105. Which of the following is an advantage of async views in DRF?

- A) Better handling of I/O-bound operations
- B) Avoids use of serializers
- C) Faster SQL execution
- D) Avoids need for routers

Answer: A) Better handling of I/O-bound operations

106. How do you define an async APIView in DRF (3.12+)?

- A) Use `async def` with standard APIView
- B) Use `AsyncAPIView`
- C) Use `AsyncViewSet`
- D) Use `@async` decorator

Answer: A) Use `async def` with standard APIView

107. When customizing the API root endpoint in DRF, which method is overridden?

- A) `get()` in `DefaultRouter`
- B) `get_api_root_view()` in `DefaultRouter`
- C) `get_root()` in `APIView`
- D) `list()` in `ViewSet`

Answer: B) `get_api_root_view()` in `DefaultRouter`

108. How can you globally override exception handling in DRF?

- A) Override `DEFAULT_EXCEPTION_HANDLER` in `settings.py`
- B) Set `raise_exceptions=True`
- C) Use custom 500.html page
- D) Replace `status.py`

Answer: A) Override `DEFAULT_EXCEPTION_HANDLER` in `settings.py`

109. What is the purpose of a `Renderer` class in DRF?

- A) Convert serialized data into a specific format like JSON
- B) Parse HTTP requests
- C) Filter querysets
- D) Set permissions

Answer: A) Convert serialized data into a specific format like JSON

110. Which DRF feature allows you to return different representations of the same object?

- A) Multiple renderers
- B) Versioning
- C) Nested serializers
- D) `GenericAPIView`

Answer: A) Multiple renderers

Would you like:

- All 110 questions in a downloadable file (PDF, Word, Excel)?
- A quiz version with answers hidden?
- Flashcards or export to quiz platforms like Kahoot or Quizlet?

Let me know how you'd like to use this set!

You said:

more

ChatGPT said:

You're clearly diving deep into Django REST Framework—awesome! Since we've already covered **110 detailed MCQs**, we'll now move into even more **advanced and specialized areas**.

Here are **10 more ultra-advanced MCQs (111–120)**, covering topics like:

- **Custom Routers**
- **Custom Renderers**
- **API Version Negotiation**
- **Third-party Integrations**
- **Content Negotiation**
- **Internationalization (i18n) in DRF**
- **View Inheritance Patterns**
- **Schema Generation**
- **Exception Handling**
- **Rate Limiting Strategies**

□ Expert-Level DRF MCQs (111–120)

111. How do you create a custom router in DRF?

- A) Subclass `BaseRouter` and override `get_urls()`
- B) Override `ModelViewSet.get_urls()`
- C) Modify `urls.py` manually
- D) Subclass `DefaultRouter` and override `dispatch()`

Answer: A) Subclass `BaseRouter` and override `get_urls()`

112. Which method in a custom `Renderer` is responsible for output formatting?

- A) `render(self, data, accepted_media_type, renderer_context)`
- B) `encode(data)`
- C) `to_representation(data)`
- D) `output()`

Answer: A) `render(self, data, accepted_media_type, renderer_context)`

113. Which class is responsible for handling version negotiation in DRF?

- A) BaseVersioning
- B) VersionNegotiation
- C) AcceptHeaderVersioning
- D) VersionParser

Answer: A) BaseVersioning

114. When integrating DRF with third-party APIs, which serializer field is useful for passing complex nested input?

- A) JSONField
- B) DictField
- C) NestedSerializer
- D) ComplexField

Answer: A) JSONField (*DRF 3.11+ supports this out of the box*)

115. Which DRF component determines how the API selects the renderer or parser?

- A) ContentNegotiation
- B) MediaSelector
- C) MIMEHandler
- D) RequestDispatcher

Answer: A) ContentNegotiation

116. What is required for enabling DRF output in multiple languages?

- A) Mark strings with `gettext_lazy` in serializers and views
- B) Use language-specific serializers
- C) Enable `LANGUAGE_SWITCH` in settings
- D) Use `i18n=True` in URLs

Answer: A) Mark strings with `gettext_lazy` in serializers and views

117. Which DRF base class enables complex inheritance chains for reusable API logic?

- A) GenericAPIView

- B) `APIView`
- C) `BaseView`
- D) `MixinView`

Answer: A) `GenericAPIView`

118. Which tool generates OpenAPI schemas automatically in DRF 3.10+?

- A) `SchemaGenerator`
- B) `SwaggerAutoSchema`
- C) `AutoSchema`
- D) `drf-openapi`

Answer: C) `AutoSchema`

119. How can you customize how DRF handles exceptions globally?

- A) Override `EXCEPTION_HANDLER` in settings and define a custom handler
- B) Use `raise_exception=False` everywhere
- C) Set `HANDLE_ERRORS=True`
- D) Replace the `BaseException` class

Answer: A) Override `EXCEPTION_HANDLER` in settings and define a custom handler

120. What is the best strategy for differentiating rate limits across user roles (e.g., staff vs. regular)?

- A) Subclass `UserRateThrottle` and override `get_cache_key()`
- B) Use `AnonRateThrottle` for all
- C) Create multiple settings.py files
- D) Use `JWTRateThrottle`

Answer: A) Subclass `UserRateThrottle` and override `get_cache_key()`