Drupal 9/10 & Drupal 10+ Exam Preparation Guide

1. Drupal Event Subscribers

What is it?

Drupal Event Subscribers is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Drupal Event Subscribers in a global enterprise Drupal rollout.
2. Used Drupal Event Subscribers to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

2. Creating Custom Block Plugin

What is it?

Creating Custom Block Plugin is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom Block Plugin in a global enterprise Drupal rollout.
2. Used Creating Custom Block Plugin to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.

- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

3. Creating Custom REST Resource

What is it?

Creating Custom REST Resource is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom REST Resource in a global enterprise Drupal rollout.
2. Used Creating Custom REST Resource to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

4. Using Dependency Injection in Drupal

What is it?

Using Dependency Injection in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Dependency Injection in Drupal in a global enterprise Drupal rollout.
2. Used Using Dependency Injection in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

5. Drupal Service Container

What is it?

Drupal Service Container is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Drupal Service Container in a global enterprise Drupal rollout.
2. Used Drupal Service Container to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

6. Config Management & Config Split

What is it?

Config Management & Config Split is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Config Management & Config Split in a global enterprise Drupal rollout.
2. Used Config Management & Config Split to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
```

```
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

7. Working with Views Programmatically

What is it?

Working with Views Programmatically is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Working with Views Programmatically in a global enterprise Drupal rollout.
2. Used Working with Views Programmatically to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

8. Creating Custom Entity Type

What is it?

Creating Custom Entity Type is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom Entity Type in a global enterprise Drupal rollout.
2. Used Creating Custom Entity Type to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
```

```
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

9. Creating Custom Field Formatter Plugin

What is it?

Creating Custom Field Formatter Plugin is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom Field Formatter Plugin in a global enterprise Drupal rollout.
2. Used Creating Custom Field Formatter Plugin to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

10. Using Queue Workers for Batch Processing

What is it?

Using Queue Workers for Batch Processing is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Queue Workers for Batch Processing in a global enterprise Drupal rollout.
2. Used Using Queue Workers for Batch Processing to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

11. Creating Custom Form with Form API

What is it?

Creating Custom Form with Form API is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom Form with Form API in a global enterprise Drupal rollout.
2. Used Creating Custom Form with Form API to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

12. Form Alter Hooks and Event Subscribers

What is it?

Form Alter Hooks and Event Subscribers is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Form Alter Hooks and Event Subscribers in a global enterprise Drupal rollout.
2. Used Form Alter Hooks and Event Subscribers to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

13. Working with Paragraphs Module

What is it?

Working with Paragraphs Module is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Working with Paragraphs Module in a global enterprise Drupal rollout.
2. Used Working with Paragraphs Module to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

14. Using Features Module for Config Export

What is it?

Using Features Module for Config Export is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Features Module for Config Export in a global enterprise Drupal rollout.
2. Used Using Features Module for Config Export to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

15. Implementing Search API + Solr Integration

What is it?

Implementing Search API + Solr Integration is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Implementing Search API + Solr Integration in a global enterprise Drupal rollout.
2. Used Implementing Search API + Solr Integration to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.

- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

16. Using Media & Media Library Module

What is it?

Using Media & Media Library Module is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Media & Media Library Module in a global enterprise Drupal rollout.
2. Used Using Media & Media Library Module to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

17. Using Layout Builder for Custom Layouts

What is it?

Using Layout Builder for Custom Layouts is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Layout Builder for Custom Layouts in a global enterprise Drupal rollout.
2. Used Using Layout Builder for Custom Layouts to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

18. Defining Custom Route and Controller

What is it?

Defining Custom Route and Controller is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Defining Custom Route and Controller in a global enterprise Drupal rollout.
2. Used Defining Custom Route and Controller to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

19. Using Drupal Settings System

What is it?

Using Drupal Settings System is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Drupal Settings System in a global enterprise Drupal rollout.
2. Used Using Drupal Settings System to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
```

```
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

20. Implementing Drupal Access Checkers

What is it?

Implementing Drupal Access Checkers is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Implementing Drupal Access Checkers in a global enterprise Drupal rollout.
2. Used Implementing Drupal Access Checkers to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

21. Using Twig Templating for Theme Overrides

What is it?

Using Twig Templating for Theme Overrides is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Twig Templating for Theme Overrides in a global enterprise Drupal rollout.
2. Used Using Twig Templating for Theme Overrides to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

```

```
function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

22. Creating Sub-themes from Base Themes

What is it?

Creating Sub-themes from Base Themes is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Sub-themes from Base Themes in a global enterprise Drupal rollout.
2. Used Creating Sub-themes from Base Themes to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

23. Using Libraries to Attach CSS/JS

What is it?

Using Libraries to Attach CSS/JS is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Libraries to Attach CSS/JS in a global enterprise Drupal rollout.
2. Used Using Libraries to Attach CSS/JS to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

24. Drupal Cache API Best Practices

What is it?

Drupal Cache API Best Practices is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Drupal Cache API Best Practices in a global enterprise Drupal rollout.
2. Used Drupal Cache API Best Practices to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

25. Implementing Multilingual & Content Translation

What is it?

Implementing Multilingual & Content Translation is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Implementing Multilingual & Content Translation in a global enterprise Drupal rollout.
2. Used Implementing Multilingual & Content Translation to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

26. Handling Ajax in Custom Forms

What is it?

Handling Ajax in Custom Forms is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Handling Ajax in Custom Forms in a global enterprise Drupal rollout.
2. Used Handling Ajax in Custom Forms to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

27. Using Migrate API for Data Migration

What is it?

Using Migrate API for Data Migration is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Migrate API for Data Migration in a global enterprise Drupal rollout.
2. Used Using Migrate API for Data Migration to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

28. Creating Custom Migration Plugins

What is it?

Creating Custom Migration Plugins is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom Migration Plugins in a global enterprise Drupal rollout.
2. Used Creating Custom Migration Plugins to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.

- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

29. Using Rules Module for Workflow Automation

What is it?

Using Rules Module for Workflow Automation is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Rules Module for Workflow Automation in a global enterprise Drupal rollout.
2. Used Using Rules Module for Workflow Automation to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

30. Role & Permissions Management Programmatically

What is it?

Role & Permissions Management Programmatically is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Role & Permissions Management Programmatically in a global enterprise Drupal rollout.
2. Used Role & Permissions Management Programmatically to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
```

```
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

31. Using User & Role APIs for User Management

What is it?

Using User & Role APIs for User Management is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using User & Role APIs for User Management in a global enterprise Drupal rollout.
2. Used Using User & Role APIs for User Management to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

32. Overriding Core Services Properly

What is it?

Overriding Core Services Properly is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Overriding Core Services Properly in a global enterprise Drupal rollout.
2. Used Overriding Core Services Properly to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

```

```
function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

33. Creating Custom Views Field Plugin

What is it?

Creating Custom Views Field Plugin is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom Views Field Plugin in a global enterprise Drupal rollout.
2. Used Creating Custom Views Field Plugin to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

34. Writing PHPUnit Tests for Custom Code

What is it?

Writing PHPUnit Tests for Custom Code is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Writing PHPUnit Tests for Custom Code in a global enterprise Drupal rollout.
2. Used Writing PHPUnit Tests for Custom Code to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

35. Using Behat for Behavior Testing in Drupal

What is it?

Using Behat for Behavior Testing in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Behat for Behavior Testing in Drupal in a global enterprise Drupal rollout.
2. Used Using Behat for Behavior Testing in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

36. Integrating Guzzle HTTP Client for External APIs

What is it?

Integrating Guzzle HTTP Client for External APIs is a critical competency for Drupal 9/10 enterprise development

including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Integrating Guzzle HTTP Client for External APIs in a global enterprise Drupal rollout.
2. Used Integrating Guzzle HTTP Client for External APIs to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

37. Working with Composer and Drupal Projects

What is it?

Working with Composer and Drupal Projects is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Working with Composer and Drupal Projects in a global enterprise Drupal rollout.
2. Used Working with Composer and Drupal Projects to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.

- Validate inputs and prevent security issues such as SQL injection.

38. Writing Drush Custom Commands

What is it?

Writing Drush Custom Commands is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Writing Drush Custom Commands in a global enterprise Drupal rollout.
2. Used Writing Drush Custom Commands to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

39. Using BLT (Build and Launch Tool) for Acquia

What is it?

Using BLT (Build and Launch Tool) for Acquia is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using BLT (Build and Launch Tool) for Acquia in a global enterprise Drupal rollout.
2. Used Using BLT (Build and Launch Tool) for Acquia to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

40. Implementing Custom Menu Links Programmatically

What is it?

Implementing Custom Menu Links Programmatically is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Implementing Custom Menu Links Programmatically in a global enterprise Drupal rollout.
2. Used Implementing Custom Menu Links Programmatically to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

41. Using Scheduler Module for Scheduled Publishing

What is it?

Using Scheduler Module for Scheduled Publishing is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Scheduler Module for Scheduled Publishing in a global enterprise Drupal rollout.
2. Used Using Scheduler Module for Scheduled Publishing to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
```

```
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

42. Implementing Fieldable Block Types with Block Content

What is it?

Implementing Fieldable Block Types with Block Content is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Implementing Fieldable Block Types with Block Content in a global enterprise Drupal rollout.
2. Used Implementing Fieldable Block Types with Block Content to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

43. Creating Multistep Forms in Drupal

What is it?

Creating Multistep Forms in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Multistep Forms in Drupal in a global enterprise Drupal rollout.
2. Used Creating Multistep Forms in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

44. Understanding and Using Event Dispatcher

What is it?

Understanding and Using Event Dispatcher is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Understanding and Using Event Dispatcher in a global enterprise Drupal rollout.
2. Used Understanding and Using Event Dispatcher to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

45. Creating Custom CKEditor Plugins

What is it?

Creating Custom CKEditor Plugins is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom CKEditor Plugins in a global enterprise Drupal rollout.
2. Used Creating Custom CKEditor Plugins to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

46. Contributing Patches to Drupal.org Projects

What is it?

Contributing Patches to Drupal.org Projects is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Contributing Patches to Drupal.org Projects in a global enterprise Drupal rollout.
2. Used Contributing Patches to Drupal.org Projects to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

47. Implementing Configuration Schema YAML Files

What is it?

Implementing Configuration Schema YAML Files is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Implementing Configuration Schema YAML Files in a global enterprise Drupal rollout.
2. Used Implementing Configuration Schema YAML Files to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

48. Using the Workflow & Content Moderation Module

What is it?

Using the Workflow & Content Moderation Module is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using the Workflow & Content Moderation Module in a global enterprise Drupal rollout.
2. Used Using the Workflow & Content Moderation Module to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

49. Best Practices for Large Site Performance Tuning

What is it?

Best Practices for Large Site Performance Tuning is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Best Practices for Large Site Performance Tuning in a global enterprise Drupal rollout.
2. Used Best Practices for Large Site Performance Tuning to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

50. Drupal Security: XSS, CSRF, SQL Injection Prevention

What is it?

Drupal Security: XSS, CSRF, SQL Injection Prevention is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Drupal Security: XSS, CSRF, SQL Injection Prevention in a global enterprise Drupal rollout.
2. Used Drupal Security: XSS, CSRF, SQL Injection Prevention to solve business problems for large complex content-driven websites.

End-to-End Solution

```
use Drupal\node\Entity\Node;

function load_published_nodes() {
  $nids = \Drupal::entityQuery('node')
    ->condition('status', 1)
```

```
    ->range(0, 5)
    ->execute();
  $nodes = Node::loadMultiple($nids);
  return $nodes;
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

51. Drupal Theme Layer Overview

What is it?

Drupal Theme Layer Overview is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Drupal Theme Layer Overview in a global enterprise Drupal rollout.
2. Used Drupal Theme Layer Overview to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

52. Twig Template Inheritance

What is it?

Twig Template Inheritance is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Twig Template Inheritance in a global enterprise Drupal rollout.
2. Used Twig Template Inheritance to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

53. Preprocess Functions in Themes

What is it?

Preprocess Functions in Themes is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Preprocess Functions in Themes in a global enterprise Drupal rollout.
2. Used Preprocess Functions in Themes to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

54. Creating Custom Theme Suggestions

What is it?

Creating Custom Theme Suggestions is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom Theme Suggestions in a global enterprise Drupal rollout.
2. Used Creating Custom Theme Suggestions to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

55. Twig Debugging Techniques

What is it?

Twig Debugging Techniques is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Twig Debugging Techniques in a global enterprise Drupal rollout.
2. Used Twig Debugging Techniques to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

56. Custom Libraries and Attaching Assets

What is it?

Custom Libraries and Attaching Assets is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Custom Libraries and Attaching Assets in a global enterprise Drupal rollout.
2. Used Custom Libraries and Attaching Assets to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

57. Creating a Sub-theme in Drupal

What is it?

Creating a Sub-theme in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating a Sub-theme in Drupal in a global enterprise Drupal rollout.
2. Used Creating a Sub-theme in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

58. Overriding Core Templates

What is it?

Overriding Core Templates is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Overriding Core Templates in a global enterprise Drupal rollout.
2. Used Overriding Core Templates to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

59. Creating Theme Settings Form

What is it?

Creating Theme Settings Form is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Theme Settings Form in a global enterprise Drupal rollout.
2. Used Creating Theme Settings Form to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

60. Using Attributes in Twig Templates

What is it?

Using Attributes in Twig Templates is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Attributes in Twig Templates in a global enterprise Drupal rollout.
2. Used Using Attributes in Twig Templates to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

61. Extending a Base Theme

What is it?

Extending a Base Theme is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Extending a Base Theme in a global enterprise Drupal rollout.
2. Used Extending a Base Theme to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

62. Custom JavaScript Behaviors in Drupal

What is it?

Custom JavaScript Behaviors in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Custom JavaScript Behaviors in Drupal in a global enterprise Drupal rollout.
2. Used Custom JavaScript Behaviors in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

63. Using Drupal Settings in JavaScript

What is it?

Using Drupal Settings in JavaScript is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Drupal Settings in JavaScript in a global enterprise Drupal rollout.
2. Used Using Drupal Settings in JavaScript to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

64. Responsive Images and Image Styles

What is it?

Responsive Images and Image Styles is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Responsive Images and Image Styles in a global enterprise Drupal rollout.
2. Used Responsive Images and Image Styles to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

65. Theme Accessibility Best Practices

What is it?

Theme Accessibility Best Practices is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theme Accessibility Best Practices in a global enterprise Drupal rollout.
2. Used Theme Accessibility Best Practices to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

66. Theming for Multilingual Sites

What is it?

Theming for Multilingual Sites is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming for Multilingual Sites in a global enterprise Drupal rollout.
2. Used Theming for Multilingual Sites to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

67. Advanced Responsive Design Techniques

What is it?

Advanced Responsive Design Techniques is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Advanced Responsive Design Techniques in a global enterprise Drupal rollout.
2. Used Advanced Responsive Design Techniques to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

68. Integrating Web Components in Drupal

What is it?

Integrating Web Components in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Integrating Web Components in Drupal in a global enterprise Drupal rollout.
2. Used Integrating Web Components in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

69. SVG Handling in Drupal Themes

What is it?

SVG Handling in Drupal Themes is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied SVG Handling in Drupal Themes in a global enterprise Drupal rollout.
2. Used SVG Handling in Drupal Themes to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

70. Implementing Design Tokens in Drupal

What is it?

Implementing Design Tokens in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Implementing Design Tokens in Drupal in a global enterprise Drupal rollout.
2. Used Implementing Design Tokens in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
```

```
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

71. Creating Layout Builder Custom Templates

What is it?

Creating Layout Builder Custom Templates is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Layout Builder Custom Templates in a global enterprise Drupal rollout.
2. Used Creating Layout Builder Custom Templates to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

72. Advanced Layout Builder Usage

What is it?

Advanced Layout Builder Usage is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Advanced Layout Builder Usage in a global enterprise Drupal rollout.
2. Used Advanced Layout Builder Usage to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.

- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

73. Paragraphs and Theming Paragraph Types

What is it?

Paragraphs and Theming Paragraph Types is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Paragraphs and Theming Paragraph Types in a global enterprise Drupal rollout.
2. Used Paragraphs and Theming Paragraph Types to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

74. Theming Views Output

What is it?

Theming Views Output is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming Views Output in a global enterprise Drupal rollout.
2. Used Theming Views Output to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

75. Field Formatters and Field Templates

What is it?

Field Formatters and Field Templates is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Field Formatters and Field Templates in a global enterprise Drupal rollout.
2. Used Field Formatters and Field Templates to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

76. Overriding Field Templates Globally

What is it?

Overriding Field Templates Globally is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Overriding Field Templates Globally in a global enterprise Drupal rollout.
2. Used Overriding Field Templates Globally to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

77. Theming User Profiles and User Pages

What is it?

Theming User Profiles and User Pages is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming User Profiles and User Pages in a global enterprise Drupal rollout.
2. Used Theming User Profiles and User Pages to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

78. Creating Custom Block Templates

What is it?

Creating Custom Block Templates is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom Block Templates in a global enterprise Drupal rollout.
2. Used Creating Custom Block Templates to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

79. Advanced Preprocess Logic for Themes

What is it?

Advanced Preprocess Logic for Themes is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Advanced Preprocess Logic for Themes in a global enterprise Drupal rollout.
2. Used Advanced Preprocess Logic for Themes to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

80. Theming Forms and Field Widgets

What is it?

Theming Forms and Field Widgets is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming Forms and Field Widgets in a global enterprise Drupal rollout.
2. Used Theming Forms and Field Widgets to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

81. Working with Theme Hooks

What is it?

Working with Theme Hooks is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Working with Theme Hooks in a global enterprise Drupal rollout.
2. Used Working with Theme Hooks to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

82. Theming Drupal Commerce

What is it?

Theming Drupal Commerce is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming Drupal Commerce in a global enterprise Drupal rollout.
2. Used Theming Drupal Commerce to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

83. Theming with Component Libraries

What is it?

Theming with Component Libraries is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming with Component Libraries in a global enterprise Drupal rollout.
2. Used Theming with Component Libraries to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

Drupal 9/10 & Drupal 10+ Exam Preparation Guide

84. Drupal Pattern Lab Integration

What is it?

Drupal Pattern Lab Integration is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Drupal Pattern Lab Integration in a global enterprise Drupal rollout.
2. Used Drupal Pattern Lab Integration to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

85. Creating Custom Theme Regions

What is it?

Creating Custom Theme Regions is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom Theme Regions in a global enterprise Drupal rollout.
2. Used Creating Custom Theme Regions to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

86. Working with Theme Suggestion Alter Hooks

What is it?

Working with Theme Suggestion Alter Hooks is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Working with Theme Suggestion Alter Hooks in a global enterprise Drupal rollout.
2. Used Working with Theme Suggestion Alter Hooks to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

87. Creating Custom Field Formatters (Theming)

What is it?

Creating Custom Field Formatters (Theming) is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom Field Formatters (Theming) in a global enterprise Drupal rollout.
2. Used Creating Custom Field Formatters (Theming) to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

88. Theming Error Pages (403, 404)

What is it?

Theming Error Pages (403, 404) is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming Error Pages (403, 404) in a global enterprise Drupal rollout.
2. Used Theming Error Pages (403, 404) to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

89. Creating Custom Page Templates

What is it?

Creating Custom Page Templates is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Custom Page Templates in a global enterprise Drupal rollout.
2. Used Creating Custom Page Templates to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

90. Using Theme Hook Suggestions Effectively

What is it?

Using Theme Hook Suggestions Effectively is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Theme Hook Suggestions Effectively in a global enterprise Drupal rollout.
2. Used Using Theme Hook Suggestions Effectively to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
```

```
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

91. Theming with View Modes

What is it?

Theming with View Modes is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming with View Modes in a global enterprise Drupal rollout.
2. Used Theming with View Modes to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

92. Using Color Module for Theme Color Settings

What is it?

Using Color Module for Theme Color Settings is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Color Module for Theme Color Settings in a global enterprise Drupal rollout.
2. Used Using Color Module for Theme Color Settings to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

93. Theming Blocks with Contextual Filters

What is it?

Theming Blocks with Contextual Filters is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming Blocks with Contextual Filters in a global enterprise Drupal rollout.
2. Used Theming Blocks with Contextual Filters to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

94. Theming Admin UI with Admin Themes

What is it?

Theming Admin UI with Admin Themes is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming Admin UI with Admin Themes in a global enterprise Drupal rollout.
2. Used Theming Admin UI with Admin Themes to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

Drupal 9/10 & Drupal 10+ Exam Preparation Guide

95. Performance Best Practices for Theming

What is it?

Performance Best Practices for Theming is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Performance Best Practices for Theming in a global enterprise Drupal rollout.
2. Used Performance Best Practices for Theming to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

96. Theming Media Entities

What is it?

Theming Media Entities is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming Media Entities in a global enterprise Drupal rollout.
2. Used Theming Media Entities to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

97. Advanced Drupal Frontend Debugging

What is it?

Advanced Drupal Frontend Debugging is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Advanced Drupal Frontend Debugging in a global enterprise Drupal rollout.
2. Used Advanced Drupal Frontend Debugging to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

98. Twig Filters and Custom Twig Extensions

What is it?

Twig Filters and Custom Twig Extensions is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Twig Filters and Custom Twig Extensions in a global enterprise Drupal rollout.
2. Used Twig Filters and Custom Twig Extensions to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

99. Theming Reusable Components with Single Directory Components

What is it?

Theming Reusable Components with Single Directory Components is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Theming Reusable Components with Single Directory Components in a global enterprise Drupal rollout.
2. Used Theming Reusable Components with Single Directory Components to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

100. Introduction to Drupal Database API

What is it?

Introduction to Drupal Database API is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Introduction to Drupal Database API in a global enterprise Drupal rollout.
2. Used Introduction to Drupal Database API to solve business problems for large complex content-driven websites.

End-to-End Solution

```
{# Example Twig template override #}
<div class="my-custom-class">
  <h2>{{ label }}</h2>
  <div>{{ content }}</div>
</div>
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

101. Connecting to External Databases in Drupal

What is it?

Connecting to External Databases in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Connecting to External Databases in Drupal in a global enterprise Drupal rollout.
2. Used Connecting to External Databases in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
```

```
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

102. Using db_query() for Raw Queries

What is it?

Using db_query() for Raw Queries is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using db_query() for Raw Queries in a global enterprise Drupal rollout.
2. Used Using db_query() for Raw Queries to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

103. Writing Secure db_select() Queries

What is it?

Writing Secure db_select() Queries is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Writing Secure db_select() Queries in a global enterprise Drupal rollout.
2. Used Writing Secure db_select() Queries to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
```

```
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

104. Using db_insert() for Programmatic Data Insertion

What is it?

Using db_insert() for Programmatic Data Insertion is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using db_insert() for Programmatic Data Insertion in a global enterprise Drupal rollout.
2. Used Using db_insert() for Programmatic Data Insertion to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

105. Updating Records Programmatically with db_update()

What is it?

Updating Records Programmatically with db_update() is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Updating Records Programmatically with db_update() in a global enterprise Drupal rollout.
2. Used Updating Records Programmatically with db_update() to solve business problems for large complex

content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

106. Deleting Records with db_delete() Safely

What is it?

Deleting Records with db_delete() Safely is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Deleting Records with db_delete() Safely in a global enterprise Drupal rollout.
2. Used Deleting Records with db_delete() Safely to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

107. Joining Tables with Drupal Database API

What is it?

Joining Tables with Drupal Database API is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Joining Tables with Drupal Database API in a global enterprise Drupal rollout.
2. Used Joining Tables with Drupal Database API to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

108. Using db_transaction() for Atomic Operations

What is it?

Using db_transaction() for Atomic Operations is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using db_transaction() for Atomic Operations in a global enterprise Drupal rollout.
2. Used Using db_transaction() for Atomic Operations to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

109. Fetching Results using fetchAll(), fetchAssoc()

What is it?

Fetching Results using fetchAll(), fetchAssoc() is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Fetching Results using fetchAll(), fetchAssoc() in a global enterprise Drupal rollout.
2. Used Fetching Results using fetchAll(), fetchAssoc() to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

110. Creating Dynamic Queries with Condition Groups

What is it?

Creating Dynamic Queries with Condition Groups is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Dynamic Queries with Condition Groups in a global enterprise Drupal rollout.
2. Used Creating Dynamic Queries with Condition Groups to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

111. Extending Entity Query for Complex Conditions

What is it?

Extending Entity Query for Complex Conditions is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Extending Entity Query for Complex Conditions in a global enterprise Drupal rollout.
2. Used Extending Entity Query for Complex Conditions to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

112. Understanding Drupal Schema API for Table Definitions

What is it?

Understanding Drupal Schema API for Table Definitions is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Understanding Drupal Schema API for Table Definitions in a global enterprise Drupal rollout.
2. Used Understanding Drupal Schema API for Table Definitions to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

113. Writing hook_schema() for Custom Tables

What is it?

Writing hook_schema() for Custom Tables is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Writing hook_schema() for Custom Tables in a global enterprise Drupal rollout.
2. Used Writing hook_schema() for Custom Tables to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

114. Creating Schema with Install Files in Modules

What is it?

Creating Schema with Install Files in Modules is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Schema with Install Files in Modules in a global enterprise Drupal rollout.
2. Used Creating Schema with Install Files in Modules to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
```

```
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

115. Working with Custom Database Tables in Drupal 9/10

What is it?

Working with Custom Database Tables in Drupal 9/10 is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Working with Custom Database Tables in Drupal 9/10 in a global enterprise Drupal rollout.
2. Used Working with Custom Database Tables in Drupal 9/10 to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

116. Using Drupal Connection Service for Multiple DBs

What is it?

Using Drupal Connection Service for Multiple DBs is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Drupal Connection Service for Multiple DBs in a global enterprise Drupal rollout.
2. Used Using Drupal Connection Service for Multiple DBs to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
```

```
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

117. Configuring Read and Write DB Split in Settings.php

What is it?

Configuring Read and Write DB Split in Settings.php is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Configuring Read and Write DB Split in Settings.php in a global enterprise Drupal rollout.
2. Used Configuring Read and Write DB Split in Settings.php to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

118. Best Practices for SQL Query Performance in Drupal

What is it?

Best Practices for SQL Query Performance in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Best Practices for SQL Query Performance in Drupal in a global enterprise Drupal rollout.
2. Used Best Practices for SQL Query Performance in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

119. Using Symfony DBAL for Advanced Database Work

What is it?

Using Symfony DBAL for Advanced Database Work is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Symfony DBAL for Advanced Database Work in a global enterprise Drupal rollout.
2. Used Using Symfony DBAL for Advanced Database Work to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

120. Writing Complex Select Queries with DBAL

What is it?

Writing Complex Select Queries with DBAL is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Writing Complex Select Queries with DBAL in a global enterprise Drupal rollout.
2. Used Writing Complex Select Queries with DBAL to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

121. Handling Database Exceptions and Error Logging

What is it?

Handling Database Exceptions and Error Logging is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Handling Database Exceptions and Error Logging in a global enterprise Drupal rollout.
2. Used Handling Database Exceptions and Error Logging to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

122. Preventing SQL Injection in Drupal Queries

What is it?

Preventing SQL Injection in Drupal Queries is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Preventing SQL Injection in Drupal Queries in a global enterprise Drupal rollout.
2. Used Preventing SQL Injection in Drupal Queries to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

123. Advanced Database Indexing Strategies in Drupal

What is it?

Advanced Database Indexing Strategies in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Advanced Database Indexing Strategies in Drupal in a global enterprise Drupal rollout.
2. Used Advanced Database Indexing Strategies in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

124. Creating and Using Materialized Views in Custom Tables

What is it?

Creating and Using Materialized Views in Custom Tables is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating and Using Materialized Views in Custom Tables in a global enterprise Drupal rollout.
2. Used Creating and Using Materialized Views in Custom Tables to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

125. Working with Temporary Tables in Drupal

What is it?

Working with Temporary Tables in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Working with Temporary Tables in Drupal in a global enterprise Drupal rollout.
2. Used Working with Temporary Tables in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

126. Building Reporting Dashboards using Raw Queries

What is it?

Building Reporting Dashboards using Raw Queries is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Building Reporting Dashboards using Raw Queries in a global enterprise Drupal rollout.
2. Used Building Reporting Dashboards using Raw Queries to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

127. Migrating Legacy Data via SQL into Drupal Entities

What is it?

Migrating Legacy Data via SQL into Drupal Entities is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Migrating Legacy Data via SQL into Drupal Entities in a global enterprise Drupal rollout.
2. Used Migrating Legacy Data via SQL into Drupal Entities to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
```

```
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

128. Using Drupal Queue + DB for Asynchronous Processing

What is it?

Using Drupal Queue + DB for Asynchronous Processing is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Drupal Queue + DB for Asynchronous Processing in a global enterprise Drupal rollout.
2. Used Using Drupal Queue + DB for Asynchronous Processing to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

129. Working with Views Database Integration

What is it?

Working with Views Database Integration is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Working with Views Database Integration in a global enterprise Drupal rollout.
2. Used Working with Views Database Integration to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
```

```
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

130. Using db_query_placeholder() Correctly

What is it?

Using db_query_placeholder() Correctly is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using db_query_placeholder() Correctly in a global enterprise Drupal rollout.
2. Used Using db_query_placeholder() Correctly to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

131. Creating Drupal Custom SQL Views Programmatically

What is it?

Creating Drupal Custom SQL Views Programmatically is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Drupal Custom SQL Views Programmatically in a global enterprise Drupal rollout.
2. Used Creating Drupal Custom SQL Views Programmatically to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

132. Executing Stored Procedures from Drupal

What is it?

Executing Stored Procedures from Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Executing Stored Procedures from Drupal in a global enterprise Drupal rollout.
2. Used Executing Stored Procedures from Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

133. Using Database Transactions in Multi-step Forms

What is it?

Using Database Transactions in Multi-step Forms is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Database Transactions in Multi-step Forms in a global enterprise Drupal rollout.

2. Used Using Database Transactions in Multi-step Forms to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

134. Database Testing with SQLite in SimpleTest/KernalTests

What is it?

Database Testing with SQLite in SimpleTest/KernalTests is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Database Testing with SQLite in SimpleTest/KernalTests in a global enterprise Drupal rollout.
2. Used Database Testing with SQLite in SimpleTest/KernalTests to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

135. Writing PHPUnit DB Tests with Drupal DB Layer

What is it?

Writing PHPUnit DB Tests with Drupal DB Layer is a critical competency for Drupal 9/10 enterprise development

including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Writing PHPUnit DB Tests with Drupal DB Layer in a global enterprise Drupal rollout.
2. Used Writing PHPUnit DB Tests with Drupal DB Layer to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

136. Extending SQL Queries in Views Handlers

What is it?

Extending SQL Queries in Views Handlers is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Extending SQL Queries in Views Handlers in a global enterprise Drupal rollout.
2. Used Extending SQL Queries in Views Handlers to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

137. Creating DB Backups Programmatically in Drupal

What is it?

Creating DB Backups Programmatically in Drupal is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating DB Backups Programmatically in Drupal in a global enterprise Drupal rollout.
2. Used Creating DB Backups Programmatically in Drupal to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

138. Building Analytics Dashboards with Raw SQL + Charts

What is it?

Building Analytics Dashboards with Raw SQL + Charts is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Building Analytics Dashboards with Raw SQL + Charts in a global enterprise Drupal rollout.
2. Used Building Analytics Dashboards with Raw SQL + Charts to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

139. Advanced SQL Paging & Limiting Techniques

What is it?

Advanced SQL Paging & Limiting Techniques is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Advanced SQL Paging & Limiting Techniques in a global enterprise Drupal rollout.
2. Used Advanced SQL Paging & Limiting Techniques to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

140. Creating Query Extenders for Reusable Queries

What is it?

Creating Query Extenders for Reusable Queries is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Creating Query Extenders for Reusable Queries in a global enterprise Drupal rollout.
2. Used Creating Query Extenders for Reusable Queries to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
```

```
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

141. Debugging SQL Queries in Drupal 9/10

What is it?

Debugging SQL Queries in Drupal 9/10 is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Debugging SQL Queries in Drupal 9/10 in a global enterprise Drupal rollout.
2. Used Debugging SQL Queries in Drupal 9/10 to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

142. Replacing Queries with Custom Plugins in Views

What is it?

Replacing Queries with Custom Plugins in Views is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Replacing Queries with Custom Plugins in Views in a global enterprise Drupal rollout.
2. Used Replacing Queries with Custom Plugins in Views to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
```

```
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

143. Comparing Drupal 7 vs 8/9/10 Database Layer Changes

What is it?

Comparing Drupal 7 vs 8/9/10 Database Layer Changes is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Comparing Drupal 7 vs 8/9/10 Database Layer Changes in a global enterprise Drupal rollout.
2. Used Comparing Drupal 7 vs 8/9/10 Database Layer Changes to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

144. Using Database Log Module for Query Monitoring

What is it?

Using Database Log Module for Query Monitoring is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Using Database Log Module for Query Monitoring in a global enterprise Drupal rollout.
2. Used Using Database Log Module for Query Monitoring to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

145. Tuning Database Layer for High Performance

What is it?

Tuning Database Layer for High Performance is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Tuning Database Layer for High Performance in a global enterprise Drupal rollout.
2. Used Tuning Database Layer for High Performance to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

146. Case Studies of Enterprise SQL Handling in Drupal Projects

What is it?

Case Studies of Enterprise SQL Handling in Drupal Projects is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Case Studies of Enterprise SQL Handling in Drupal Projects in a global enterprise Drupal rollout.

2. Used Case Studies of Enterprise SQL Handling in Drupal Projects to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.

147. Best Practices Checklist for Drupal + SQL Projects

What is it?

Best Practices Checklist for Drupal + SQL Projects is a critical competency for Drupal 9/10 enterprise development including site building, backend APIs, theming, and database mastery.

Real-world Scenarios

1. Applied Best Practices Checklist for Drupal + SQL Projects in a global enterprise Drupal rollout.
2. Used Best Practices Checklist for Drupal + SQL Projects to solve business problems for large complex content-driven websites.

End-to-End Solution

```
$results = \Drupal::database()->select('node_field_data', 'n')
  ->fields('n', ['nid', 'title'])
  ->condition('status', 1)
  ->range(0, 5)
  ->execute()
  ->fetchAll();
foreach ($results as $record) {
  \Drupal::logger('example')->notice($record->title);
}
```

Best Practices

- Follow Drupal coding standards and best practices.
- Use dependency injection and services.
- Document and test custom code.
- Validate inputs and prevent security issues such as SQL injection.