Test Logging Framework Usage Guide

Overview

The `TestLogger` class provides a clean, professional way to debug Flutter widget tests without cluttering your console output with `print()` statements.

- It uses Dart's built-in `dart:developer` logging system.
- The module is in the `test/utils/test_logger.dart.`

Quick Start

1. Enable/Disable Logging

Toggle debugging by changing the `enableDebugLogging` flag:

```
class TestLogger {
  static const bool enableDebugLogging = true; // Set to true for debugging
  // ... rest of class
}
```

2. Available Logging Methods

```
// Section headers (highest priority)
TestLogger.section('Starting filter test');

// Important information
TestLogger.info('Current filter: ${viewModel.currentFilter}');

// Detailed debug information
TestLogger.debug('Todo IDs: [${todos[0].id}, ${todos[1].id}]');

// Log ViewModel state comprehensively
TestLogger.logViewModelState(viewModel, 'After adding todos');

// Log todos with context
TestLogger.logTodos(viewModel.todos, 'Filtered todos visible');

// Check widget finder results
TestLogger.logWidgetFinderResults(find.text('Completed Todo'), 'Completed Todo');

// Analyze widget tree content
TestLogger.logAllTextWidgets(tester, filter: 'Todo');
```

Example Usage in Test

```dart
testWidgets('should filter by completion status', (tester) async {
  TestLogger.section('Starting filter completion status test');

  // Your test setup...
  viewModel.addTodo('Pending Todo');
  viewModel.addTodo('Completed Todo');
  await tester.pump();

  // Log state for debugging
  TestLogger.logViewModelState(viewModel, 'After adding todos');

  // Your test logic...
  final completedTodo = todos.firstWhere((todo) => todo.title == 'Completed Todo');
  TestLogger.debug('Found todo to complete: ${completedTodo.title} (ID: ${completedTodo.id})');

  // More test logic...

  // Debug widget tree if needed
  TestLogger.logAllTextWidgets(tester, filter: 'Todo');
  TestLogger.logWidgetFinderResults(find.text('Completed Todo'), 'Completed Todo');

  TestLogger.section('Filter test completed successfully');
});
```

Log Levels & Output

The framework uses different log levels for better organization:

- **Level 900**: Section headers (most important)
- **Level 800**: Info messages
- **Level 500**: Debug details

Benefits Over print()

✅ Advantages of TestLogger

- **Clean Console**: No output when `enableDebugLogging = false`
- **Professional**: Uses Dart's official logging system
- **Structured**: Different log levels and contexts
- **Easy Toggle**: Single flag controls all debug output
- **Rich Context**: Specialized methods for different data types
- **Maintainable**: Easy to remove or modify logging behavior

❌ Problems with print()

- Always outputs to console (noisy)
- No log levels or filtering
- Difficult to disable selectively
- Not professional for production code
- Hard to maintain and organize

Debugging Workflow

1. **Enable logging** by setting `enableDebugLogging = true`
2. **Run the failing test** to see detailed output
3. **Analyze the logs** to understand what's happening
4. **Fix the issue** based on the debug information
5. **Disable logging** by setting `enableDebugLogging = false`
6. **Run tests again** to verify fix without debug noise

Common Debug Scenarios

ID Collision Issues

```
TestLogger.debug('Todo IDs: [${todos[0].id}, ${todos[1].id}]');
// Check if IDs are truly unique
```

Widget Tree Problems

```
TestLogger.logAllTextWidgets(tester, filter: 'Todo');
// See exactly what text widgets exist in the tree
```

State Management Issues

```
TestLogger.logViewModelState(viewModel, 'After filtering');
// Check if ViewModel state matches expectations
```

Widget Finder Issues

```
TestLogger.logWidgetFinderResults(find.text('Expected Text'), 'Expected Text');
// See how many widgets match your finder
```

Best Practices

1. **Use descriptive contexts** in your log messages
2. **Log before and after critical operations**
3. **Use appropriate log levels** (section > info > debug)
4. **Always disable logging** before committing code
5. **Remove temporary debug code** once issues are resolved