

MICROFLOW OPTIMIZATION

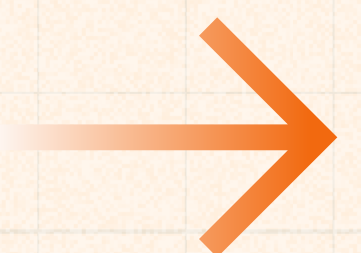
The Key to a Faster Mendix
Application

Behind Every Slow Mendix App Is
a Microflow Waiting to Be Fixed.



WHAT DO WE MEAN BY MICROFLOW OPTIMIZATION?

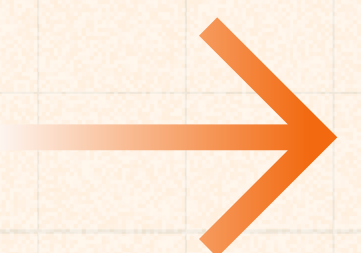
Microflow optimization is the process of improving **performance, efficiency, and scalability** in Mendix applications by refining **business logic execution**.



WHY MICROFLOW OPTIMIZATION MATTERS

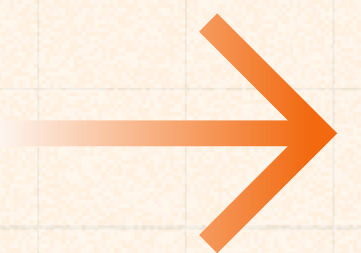
- Microflows execute your core business logic
- They are often the #1 source of performance bottlenecks
- Directly impacts user experience & responsiveness
- Affects system resource utilization (CPU, memory, DB calls)
- Determines how well your app scales under load

Optimized microflows = Faster apps, happier users, and lower infrastructure costs.



HOW DO WE OPTIMIZE MICROFLOWS?

By following the **right set of best practices**. These best practices **cover everything you need** to build efficient, scalable microflows.



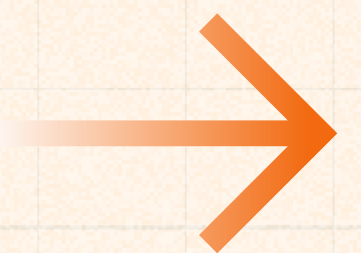
DATABASE OPERATION BEST PRACTICES

How to optimize database interactions in Mendix microflows:

- ✓ Batch operations instead of committing changes one by one
- ✓ Collect and commit all changes at the end of a process
- ✓ Use “Retrieve by Association” to fetch related objects efficiently
- ✓ Use XPath/OQL instead of retrieving entire datasets & filtering in-memory
- ✓ Apply database indexing for faster queries
- ✓ Use Non-Persistable Entities for temporary data to reduce database load

Mistakes to Avoid:

- ✗ Committing every change individually (slows down performance).
- ✗ Fetching entire tables instead of retrieving only required data.
- ✗ Not leveraging indexing for frequent queries.



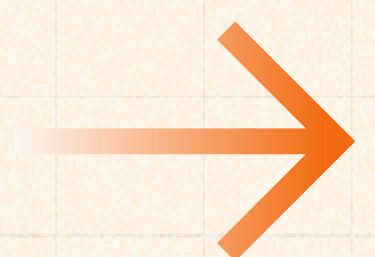
LOOP PROCESSING OPTIMIZATION

How to process large datasets efficiently:

- ✓ Use built-in list operations instead of manual loops
- ✓ Batch process large datasets instead of iterating through them one by one
- ✓ Filter data early (preferably at the database level)
- ✓ Use database aggregation instead of calculating everything in-memory
- ✓ Avoid nested loops ($O(n^2)$ operations kill performance!)

Mistakes to Avoid:

- ✗ Running unnecessary loops over entire datasets.
- ✗ Applying filtering at the microflow level instead of the database.
- ✗ Performing real-time aggregations inside microflows instead of database queries.



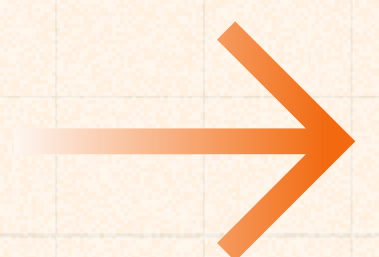
MICROFLOW STRUCTURE & DESIGN BEST PRACTICES

How to structure microflows for clarity & performance:

- ✓ **Follow the Single Responsibility Principle** – One microflow = One clear task.
- ✓ **Create modular, reusable microflows** for repeated operations.
- ✓ **Use decision splitting instead of deeply nested conditions** for readability.
- ✓ **Implement a consistent error-handling strategy** – no silent failures!
- ✓ **Document complex decision logic** – Future developers will thank you!

Mistakes to Avoid:

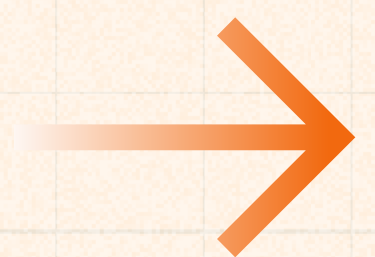
- ✗ Microflows that do too much in one place (hard to debug).
- ✗ Deeply nested decision trees instead of breaking conditions apart.
- ✗ Lack of error-handling – leading to unpredictable behavior.



ADVANCED BEST PRACTICES FOR MICROFLOW OPTIMIZATION

You've learned the fundamentals now let's dive deeper.

- Asynchronous Processing
- Performance Logging
- Optimization Strategy



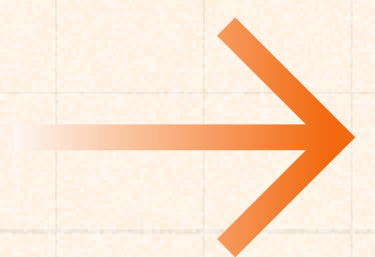
ASYNCHRONOUS PROCESSING

How to optimize time-consuming operations:

- ✓ Use scheduled events for batch processes
- ✓ Create queue-based processing for workload management
- ✓ Implement status tracking for long-running operations
- ✓ Define clear transaction boundaries

Mistakes to Avoid:

- ✗ Running everything synchronously (causes delays for users).
- ✗ Lack of status tracking for long-running jobs.



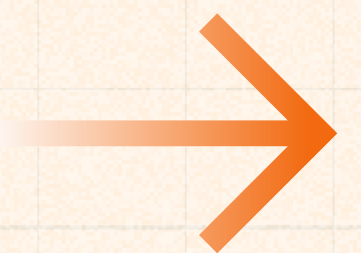
PERFORMANCE LOGGING

Track and measure performance to find bottlenecks.

- ✓ Log execution times for performance-critical microflows
- ✓ Create dashboards for live performance tracking
- ✓ Establish baselines for expected execution times
- ✓ Set up alerts for performance deviations

Mistakes to Avoid:

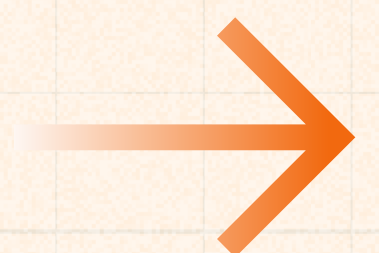
- ✗ No logging makes debugging slow performance harder.
- ✗ No baseline comparison leads to unnoticed degradation over time.



OPTIMIZATION STRATEGY

Follow this process to optimize performance:

1. Identify performance bottlenecks
2. Measure execution times with realistic data loads
3. Optimize database interactions first
4. Refactor loops & use list operations where possible
5. Implement async processing to prevent slow UI response



**SAVE THIS CHECKLIST. SHARE IT WITH
YOUR TEAM. START OPTIMIZING.**

What are your top tips for building high performance microflows?

Drop them in the comments let's build better together



hello@goldenearth.io



www.goldenearth.io