ENTERPRISE MENDIX DOMAIN MODELING

The deathbed of failure or the cornerstone of success.

Your domain model is where it all begins and where things can go terribly wrong

golden earth

WHAT IS ENTERPRISE DOMAIN MODELING?

It’s the **architectural backbone** of your Mendix application. A well-structured domain model ensures:

- ✓ Scalability
- ✓ Performance
- ✓ Maintainability
- ✓ Data Integrity and Security

golden earth

WHAT IS ENTERPRISE DOMAIN MODELING?

- The foundation of every **Mendix application**
- Directly impacts data **integrity & performance**
- Determines long-term maintainability
- Enables or constrains future scalability
- Defines how well your app **reflects business logic**

A solid domain model = fewer refactors, fewer performance issues, and better app longevity.

4

THE 4 PILLARS OF A STRONG DOMAIN MODEL

1. Entity Relationships: **Inheritance**
2. Entity Relationships: **Associations**
3. **Attribute Configuration**
4. **Strategic Indexing**

Nail these, and your Mendix app will scale smoothly. Mess them up, and… well, let’s just say you’ll need aspirin.

golden earth

5

BEST PRACTICES & MISTAKES TO AVOID

golden earth

ENTITY RELATIONSHIPS
INHERITANCE

- ✓ Keep inheritance **shallow** (2-3 levels max)
- ✓ Use inheritance **only for true "is-a" relationships**
- ✓ Consider **associations instead** when applicable
- ✓ **Shallow hierarchies** perform better
- ✓ **Document inheritance choices** (future-you will thank you)

Anti-Patterns to Avoid

- ✗ Deep inheritance chains (harder to manage, impacts performance)
- ✗ Overusing inheritance when associations work better
- ✗ Ignoring documentation—leads to confusion down the line

ENTITY RELATIONSHIPS ASSOCIATIONS

- ✓ Use **bidirectional associations only when needed**
- ✓ Define and document **cardinality** (one-to-many, many-to-many)
- ✓ Plan for **data growth** in one-to-many relationships

Anti-Patterns to Avoid

- ✗ Creating unnecessary bidirectional relationships
- ✗ Failing to account for **scalability & data growth**
- ✗ Not optimizing indexes, leading to slow queries

WHY DOMAIN MODELING MATTERS

Your Mendix app is only as strong as its foundation.

- Poor domain models = **scalability nightmares.**
- Bad configurations = **slow performance.**
- No strategic indexing = **your database fights against you.**

Let's break down the **right** way to approach this.

ATTRIBUTE CONFIGURATION THE SILENT PERFORMANCE KILLER

Bad configurations = **bloated storage, slow queries, and data inconsistencies.**

- ✓ Use **specific data types** (not Strings for everything!)
- ✓ Financial values? Stick to Decimal, Integer, or Long.
- ✓ Validate **at the attribute level** – avoid unnecessary microflows.
- ✓ Use **calculated attributes** to reduce real-time processing.
- ✓ Set **length constraints** for strings to avoid bloated storage.

Anti-Patterns to Avoid

- ✗ Using Strings for everything (kills performance).
- ✗ Ignoring validation at the attribute level.
- ✗ Not defining field limits – leads to inefficiencies.

10

STRATEGIC INDEXING
HOW TO MAKE YOUR APP FASTER

Indexes **make or break performance.**

- ✓ Add indexes on attributes frequently used in:
  - **Filters**
  - **Sorting**
  - **Retrieval constraints**
- ✓ Monitor index usage in production.
- ✓ Balance indexing with **write performance** too many = slow writes.
- ✓ Adjust indexing **based on real queries**.
- ✓ Document **why an index exists** (future-you will thank you).

Common Mistakes:

- ✗ Indexing everything – **slows down inserts/updates.**
- ✗ Forgetting to **review and adjust indexes** over time.
- ✗ Not documenting why an index was added (leading to unnecessary deletions).

golden earth

LET'S TALK

for a consultation or connect with me directly!

hello@goldenearth.io

www.goldenearth.io

golden earth