Building End-to-End Multi-Client Service Oriented Applications – *Angular Edition*

Module 04

Entities & Core Functionality

pluralsight
hardcore developer training

Entity Structure

- **Two sets of entities**
  - Business side
  - Client side
- **Each travels up & down its tiers**
- **Passed by services and proxies**
- **Each set of entities has knowledge it needs to service its side of the service wall**
- **Entities will be “equivalent”**
  - Namespace equivalency will be handled in **AssemblyInfo.cs** file

Business Entities

- **Used across data layer, business engine layer, and service layer**
- **Each entity maps to a database table**
- **Used as data contracts on business side**
  - Not the exclusive data contracts though
  - Can still have custom data contracts whenever needed
- **Data access layer will provide ORM mapping if needed**
  - DB-Context class will use EF fluent-language
- **The `EntityBase` base class provides common characteristics**
- **Interface for identifying entity ID (`IIdentifyableEntity`)**
  - Will be used in Data Access module
- **Interface for identifying account ownership (`IAccountOwnedEntity`)**
  - Will be used when adding security to application later

Property Change Notification

- **Compile-time property-change handling**
  - `() => PropertyName` vs `“PropertyName”`
- **Disallow duplicate event wiring**
  - No need to `-=` AND += when/if manually wiring to `PropertyChanged` event
- **Build dirty-setting into property change**

Dirty Tracking

- **Build into property change notification**
- **Provide way to obtain dirty objects down the graph**
- **Able to ask if object graph is dirty at any level (with short-circuiting)**
- **Uses routine to “walk” object graph**
  - Leverages cool lambda technique for reusability

Validation

- **Uses FluentValidation**
  - Simple, yet extensible rules engine
  - Available through NuGet
- **Wired functionality into ObjectBase**
- **Object validates when a property is changed**
- **"`IDataErrorInfo`" is implemented for XAML binding purposes**
  - Reports back broken validation rules for given property
- **Broken rules stored per-object**

- **You will see more on this when we get to UI**

Client Entities

- **Used across proxy layer, web API layer, and UI layer**
- **Used as data contracts on client side**
    - Not the exclusive data contracts though
    - Custom data contracts may exist on both sides
- **Have more intelligence than business entities**
    - Property change notification (binding)
    - Dirty-tracking
    - Validation (Fluent-Validation library)
- **Prepared for data binding to any type of client**
- **The `ObjectBase` base class provides common characteristics**

Unit Testing

- **Test entity characteristics by testing core base**
  - Property change notification
    - Basic notification
    - Against duplicate subscribing
  - Dirty tracking
    - Standard dirty set
    - Child dirtiness
    - Object aggregation & cleaning
    - Object cleaning
  - Validation

Unit Test Naming Conventions

- **Test class name is name of class to be tested followed by “Tests”**
- **Test names take one of two forms**
  - `PascalCase` indicates name of method to be tested
  - `lower_case_underscore_delimited` indicates custom name descriptor

End of module