Understanding the Code: EML and Methods

What is EML (Entity Manipulation Language)?

In standard ABAP, we use **SQL** (`SELECT` , `UPDATE`) to talk to the database. In RAP, we use **EML** (`READ ENTITIES` , `MODIFY ENTITIES`) to talk to the **Business Object**.

Why? Because RAP handles the buffer, drafts, and transaction state for us. If we used direct SQL, we would bypass all the safety checks and draft logic.

Concept: The Transactional Buffer (The "Shopping Cart")

The Analogy: Think of the **Buffer** like a **Shopping Cart** on Amazon.

- When you click "Add to Cart", the item isn't yours yet. It's just in a temporary list.
- Only when you click **"Checkout" (Save)** does the order go to the database.

In RAP:

- **SQL (`SELECT`)** reads from the **Database** (The Warehouse). It sees old, saved data.
- **EML (`READ ENTITIES`)** reads from the **Buffer** (The Shopping Cart). It sees the **current, unsaved data** the user is typing right now.

Rule: Always use EML inside your behavior class. If you use SQL, you will miss the user's latest changes!

---

1. The `setCurrency` Method (Determination)

Goal: "If the user forgets the currency, set it to EUR."

```
METHOD setCurrency.
  " 1. Read the current data from the buffer
  READ ENTITIES OF ZIW_Products IN LOCAL MODE
    ENTITY ZIW_Products
      FIELDS ( prd_currency ) WITH CORRESPONDING #( keys )
    RESULT DATA(products).

  " 2. Filter: Keep only products where currency is empty
  DELETE products WHERE prd_currency IS NOT INITIAL.
  CHECK products IS NOT INITIAL.

  " 3. Update the buffer with 'EUR'
  MODIFY ENTITIES OF ZIW_Products IN LOCAL MODE
    ENTITY ZIW_Products
      UPDATE
      FIELDS ( prd_currency )
      WITH VALUE #( FOR product IN products
                    ( %tky = product-%tky  " Use the Transaction Key (handles Draft/Active)
                      prd_currency = 'EUR' ) ).
ENDMETHOD.
```

Simple Explanation:

1. **Look** at what the user typed (`READ ENTITIES`).
2. **Check** if the currency is missing.
3. **Write** 'EUR' back to the form (`MODIFY ENTITIES`).

---

2. The `validatePrice` Method (Validation)

Goal: "Stop the user from saving a negative price."

```
METHOD validatePrice.
  " 1. Read the price
  READ ENTITIES OF ZIW_Products IN LOCAL MODE
    ENTITY ZIW_Products
      FIELDS ( prd_price ) WITH CORRESPONDING #( keys )
    RESULT DATA(products).

  LOOP AT products INTO DATA(product).
    IF product-prd_price <= 0.
      " 2. Tell RAP this record FAILED
      APPEND VALUE #( %tky = product-%tky ) TO failed-ziw_products.

      " 3. Create a nice error message
      APPEND VALUE #( %tky = product-%tky
                      %msg = new_message_with_text( ... text = 'Price must be > 0' )
                    ) TO reported-ziw_products.
    ENDIF.
  ENDLOOP.
ENDMETHOD.
```

Simple Explanation:

1. **Check** the price.
2. If it's bad, put it in the `failed` table (this stops the save).
3. Put a message in the `reported` table (this shows the red box on screen).

---

3. The `applyDiscount` Method (Action)

Goal: "Reduce price by 10% when button is clicked."

```
METHOD applyDiscount.
  " 1. Read current price
  READ ENTITIES ... RESULT DATA(products).

  " 2. Calculate new price in memory
  LOOP AT products ASSIGNING FIELD-SYMBOL(<product>).
    <product>-prd_price = <product>-prd_price * '0.90'.
  ENDLOOP.

  " 3. Update the app with new price
  MODIFY ENTITIES ... UPDATE FIELDS ( prd_price ) ...

  " 4. Return the result so the UI knows to refresh
  result = VALUE #( ... ).
ENDMETHOD.
```

Simple Explanation: It's a classic "Read -> Calculate -> Write" cycle, but using EML to ensure the Draft is updated correctly.

---

4. The `early_numbering_create` Method

Goal: "Generate ID automatically."

```
METHOD early_numbering_create.
  " 1. Find the last used ID
  SELECT MAX( product_id ) FROM zproduct_demo INTO @DATA(max_id).

  LOOP AT entities ASSIGNING FIELD-SYMBOL(<entity>).
    max_id = max_id + 1.

    " 2. Map the temporary ID (%cid) to the new permanent ID (product_id)
    APPEND VALUE #( %cid      = <entity>-%cid
                    %is_draft = <entity>-%is_draft " <--- CRITICAL for Drafts!
                    product_id = max_id ) TO mapped-ziw_products.
  ENDLOOP.
ENDMETHOD.
```

Simple Explanation: The app sends a "temporary ID" (`%cid`). We calculate the "real ID" (`product_id`) and tell the framework: "Hey, this temporary ID is now this real ID." **Crucial:** We must pass `%is_draft` so the system knows if we are creating a Draft or a Real record.