



Martin Jonen Consulting

www.abapconf.org | #abapconf

December 5th 2024

abap
conf2024

```
DATA(ls_structure) TYPE IS_STRUCTURE.  
ASSERT ls_structure IS NOT INITIAL.  
GET_SINITC IMPORTING ls_sadt > DATA(ls_sadt).  
DELETE ls_sadt-ATTRIBUTES USING #23.  
  
LOOP AT it_property_binding ASSUMING  
    ls_attribute = VALUE #10.  
    IF it_property_binding->IS_ATTRIBUTE.  
        ls_attribute->NAME = it_property_binding->NAME.  
        ls_attribute->BINDING = it_property_binding->NAME.  
        ls_attribute->RESULT = it_property_binding->NAME.  
        ls_attribute->TYPE = it_property_binding->NAME.  
        IF ls_property_binding->IS_ATTRIBUTE = 'X'.  
            ls_attribute->IS_KEY = 'X'.  
        ENDIF.  
        INSERT ls_attribute INTO TABLE ls_attributes.  
    ENDOBJ.  
ENDLOOP.
```

Unit Test your Business Objects in RAP

Martin Jonen

13:30 - 14:20 CET

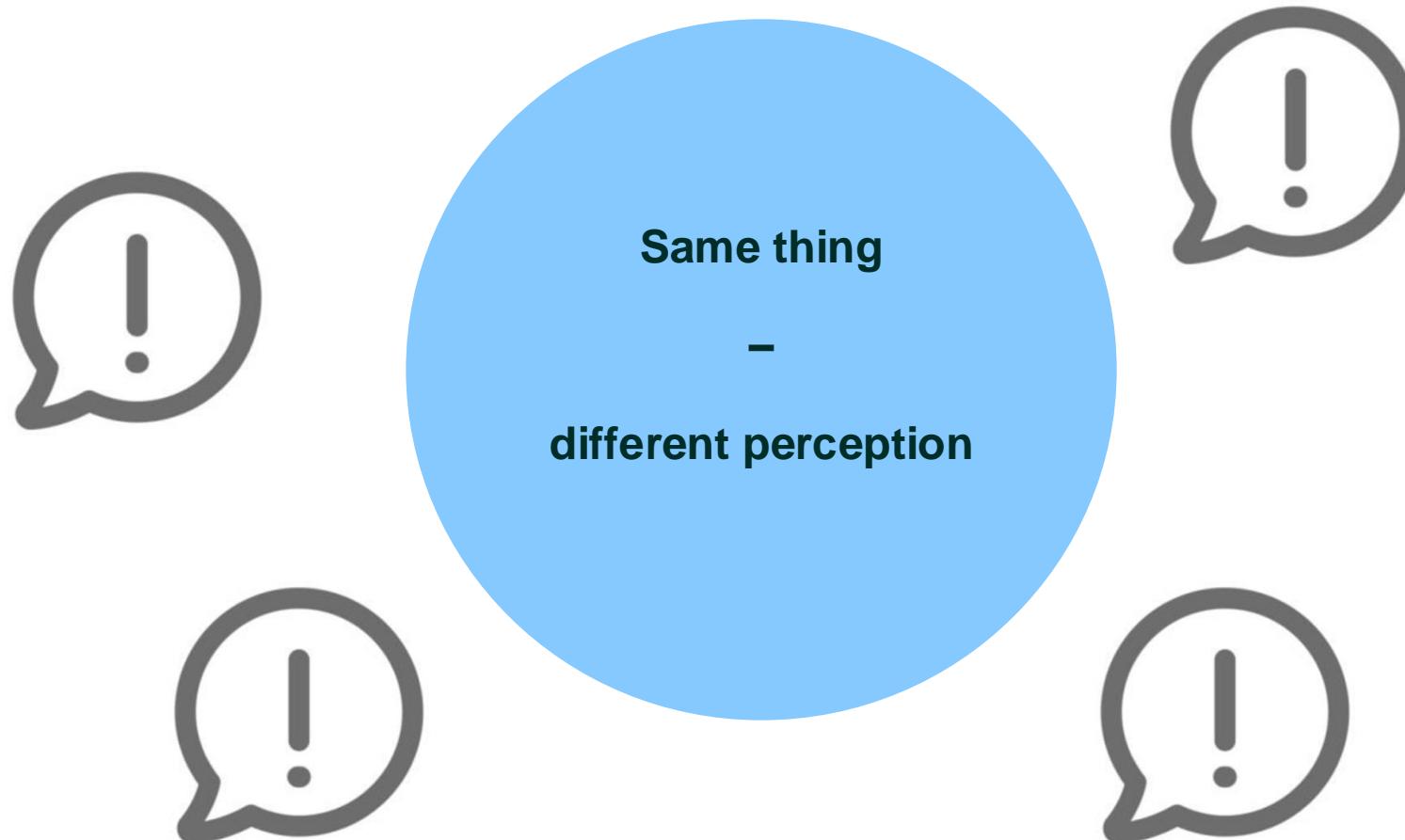
Channel 1

English





What is your first association with this blue dot?





```
31     def __init__(self, path=None, debug=False):
32         self.file = None
33         self.fingerprints = set()
34         self.logdups = True
35         self.debug = debug
36         self.logger = logging.getLogger(__name__)
37         if path:
38             self.file = open(os.path.join(path, 'request.log'), 'a')
39             self.file.seek(0)
40             self.fingerprints.update([l.strip() for l in self.file])
41
42     @classmethod
43     def from_settings(cls, settings):
44         debug = settings.getbool('SUPERSEED_DEBUG')
45         return cls(job_dir(settings), debug)
46
47     def request_seen(self, request):
48         fp = self.request_fingerprint(request)
49         if fp in self.fingerprints:
50             return True
51         self.fingerprints.add(fp)
52         if self.file:
53             self.file.write(fp + os.linesep)
54
55     def request_fingerprint(self, request):
56         return request_fingerprint(request)
```



«Power of three»

Three opinions
(Pair programming)



Three different
solution designs





Martin Jonen

SAP Consultant & Developer | SAP Clean Code Coach

My journey so far ...

- Freelanced **SAP Consultant & Developer** (+14 years)
- Certified **SAP ABAP Clean Code Coach**
(Collaboration with Damir Majer & Prof. Dr. Christian Drumm)
- **Author** (TDD course with Espresso Tutorials)
- Founder of the “**ABAP Conscious Clean Coder Hub**”
- **Hobbies:** Community, Running, Yoga



ASE Coaching



Martin Jonen Consulting

www.abapconf.org | #abapconf

December 5th 2024

abap
conf2024

```
DATA(ls_structure) TYPE IS_STRUCTURE.  
ASSERT ls_structure IS NOT INITIAL.  
GET_SINITC IMPORTING ls_sadt > ls_structures.  
DELETE ls_sadt-ATTRIBUTES USING #23.  
  
LOOP AT lt_property_binding ASSUMING ls_structures.  
    ls_attribute = VALUE #10.  
    ls_attribute-name = ls_structures-name.  
    ls_attribute-binding = ls_structures-binding.  
  
    IF ls_property_binding-is_key = 'X'.  
        ls_attribute-is_key = 'X'.  
    ENDIF.  
    INSERT ls_attribute INTO TABLE ls_sadt-ATTRIBUTES.  
ENDLOOP.
```

Unit Test your Business Objects in RAP

Martin Jonen

13:30 - 14:20 CET

Channel 1

English





At the end of this presentation

... You understood ...

- The **UNIT Test framework** in ABAP OO
- How to write **UNIT tests** for “Business Behaviors” in **Test After Development** style
- How to **mock ENTITY (READ) access** by example
- How to use **clean code principles** to structure your code



How do we get there?

... Without studying tons of theoretical stuff up front!

- By using an example of **SAP Steampunk** by Christian Drumm
- By creating test cases on base of existing coding (**Test After Development**)
- By using SAP standard classes to **mock ENTITY access** to decouple dependencies on DB data



What is this not about!

... sorry for that!

- No TDD, but we part of the **TDD cycle** (hell, yeah!!!)
- No mocking of Business Object itself
- No ENTITY (UPDATE) mock examples
- No RAP or OO beginners workshop



Upcoming (Olé!)



Now let's dive into it ...



Our setup

... Thank goodness we don't have to build everything ourselves!



- Espresso Tutorials – **ABAP Steampunk** (<https://t.ly/jJljD>)
- GitHub Repo “modern-abap-curriculumhttps://github.com/ceedee666/modern-abap-curriculum.git)
 - Clone the Repo (Cloud based – use ABAP trial environment)
 - Get started!



Showcase – Calculate average rating

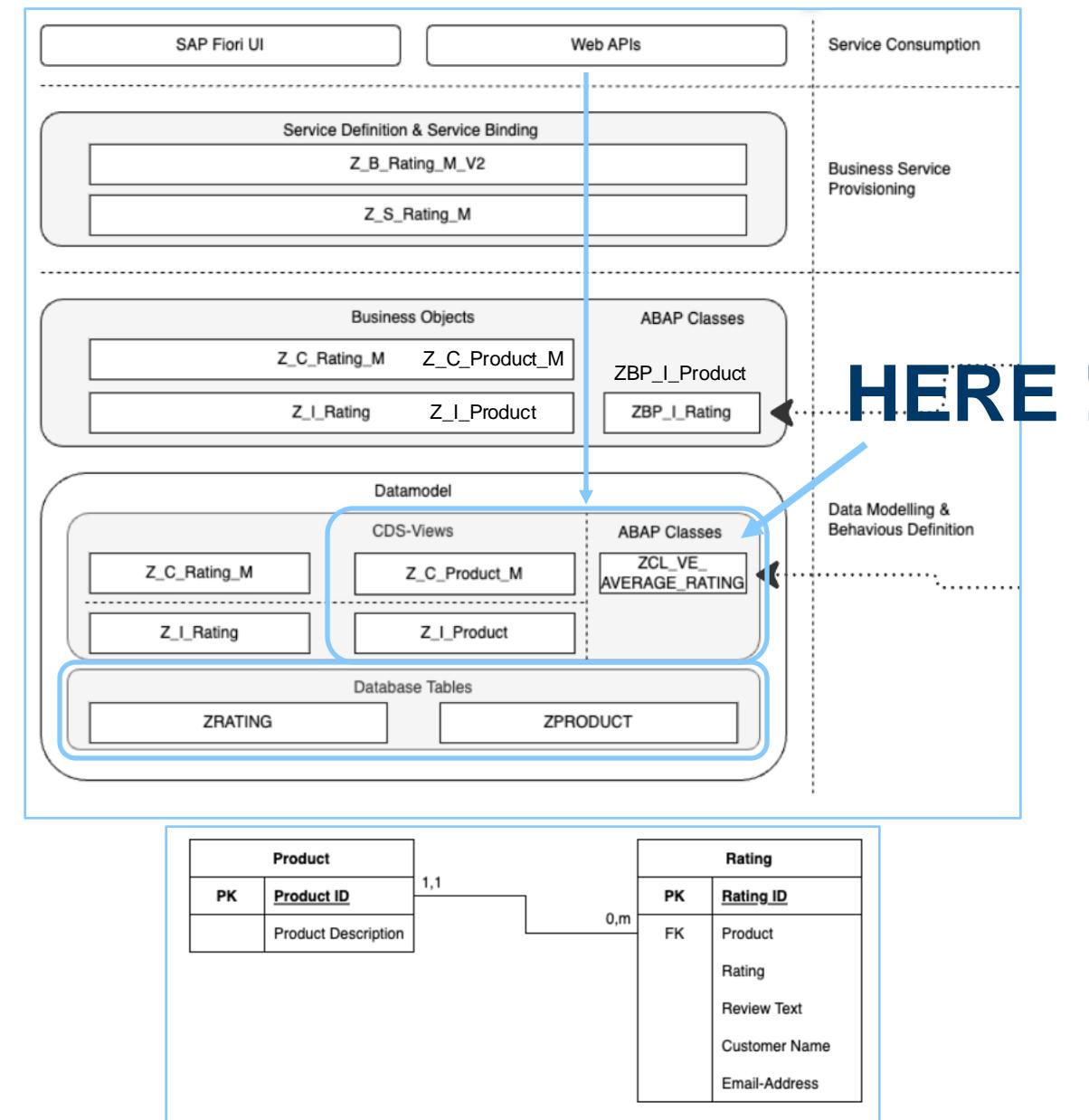
... I like those elbow pads!

Standard ▾

Search Q

Products (6)

<input type="checkbox"/> Product ID	Product Description	Average Rating
<input type="checkbox"/> DXTR1000	Deluxe Touring Bike Black	★★★★★
<input type="checkbox"/> DXTR2000	Deluxe Touring Bike Silver	★★★★★
<input type="checkbox"/> DXTR3000	Deluxe Touring Bike Red	★★★★★
<input type="checkbox"/> EPAD1000	Elbow Pad	★★★★★
<input type="checkbox"/> HELLO	Hello	★★★★★
<input type="checkbox"/> KPAD1000	Knee Pad	★★★★★





- Business logic class

```
► C ZCL_VE_AVERAGE_RATING_MJ
1 CLASS zcl_ve_average_rating_mj DEFINITION
2 PUBLIC
3 FINAL
4 CREATE PUBLIC .
5
6 PUBLIC SECTION.
7   INTERFACES if_sadl_exit_calc_element_read .
8

► C ZCL_VE_AVERAGE_RATING_MJ ► ● IF_SADL_EXIT_CALC_ELEMENT_READ~CALCULATE
1/
18 METHOD if_sadl_exit_calc_element_read~calculate.
19   DATA products TYPE STANDARD TABLE OF Z_C_Product_M_MJ.
20   products = CORRESPONDING #( it_original_data ).
21
22   " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
23   READ ENTITIES OF Z_C_Product_M_MJ
24     ENTITY Product BY \_Rating
25       FIELDS ( Rating )
26       WITH CORRESPONDING #( products )
27       RESULT FINAL(ratings).
28
29   " Calculate the average rating for each product by DB select
30   SELECT Product,
31     AVG( rating AS DEC( 2, 1 ) ) AS average_rating
32   FROM @ratings AS r
33   GROUP BY Product
34   INTO TABLE @DATA(average_product_ratings).
35
36   " Map average ratings to the products (output)
37   LOOP AT products ASSIGNING FIELD-SYMBOL(<product>).
38     READ TABLE average_product_ratings
39       WITH KEY Product = <product>-ProductId
40       INTO DATA(average_product_rating).
41     IF sy-subrc = 0.
42       <product>-AverageRating = average_product_rating-average_rating.
43     ENDIF.
44   ENDLOOP.
45
46   ct_calculated_data = CORRESPONDING #( products ).
47
48 ENDMETHOD.
```



- Business logic class

```
► C ZCL_VE_AVERAGE_RATING_MJ
1 CLASS zcl_ve_average_rating_mj DEFINITION
2   PUBLIC
3   FINAL
4   CREATE PUBLIC .
5
6   PUBLIC SECTION.
7     INTERFACES if_sadl_exit_calc_element_read .
8

► C ZCL_VE_AVERAGE_RATING_MJ ► ● IF_SADL_EXIT_CALC_ELEMENT_READ~CALCULATE
1/
18 METHOD if_sadl_exit_calc_element_read~calculate.
19   DATA products TYPE STANDARD TABLE OF Z_C_Product_M_MJ.
20   products = CORRESPONDING #( it_original_data ).
21
22   " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
23   READ ENTITIES OF Z_C_Product_M_MJ
24     ENTITY Product BY \_Rating
25     FIELDS ( Rating )
26     WITH CORRESPONDING #( products )
27     RESULT FINAL(ratings).
28
```

```
11 @Metadata.allowExtensions: true
12 define root view entity Z_C_Product_M_MJ
13   as projection on Z_I_Product_MJ
14 {
15   key ProductId,
16     ProductDescription,
17
18   @ObjectModel.virtualElementCalculatedBy: 'ABAP:ZCL_VE_AVERAGE_RATING_MJ'
19   @EndUserText.label: 'Average Customer Rating'
20   virtual AverageRating: abap.dec( 2, 1 ),
21
22   /* Associations */
23   Rating : redirected to composition child Z_C_RATING_M_MJ
24 }
```

t by DB select
_rating
ct>).
rating-average_rating.
).

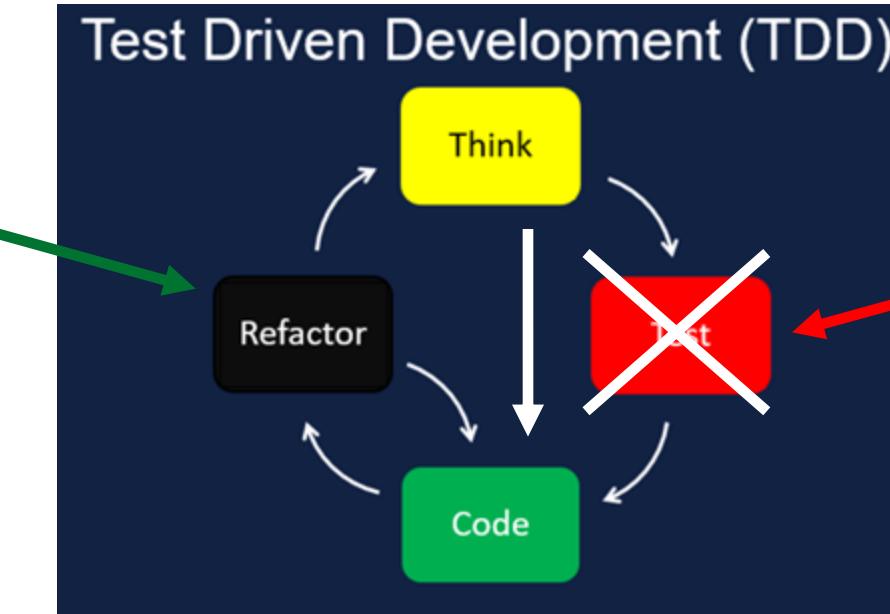


How to test legacy code?

... everyone likes to code clean!

- **Test After Development**

Here, the magic happens!





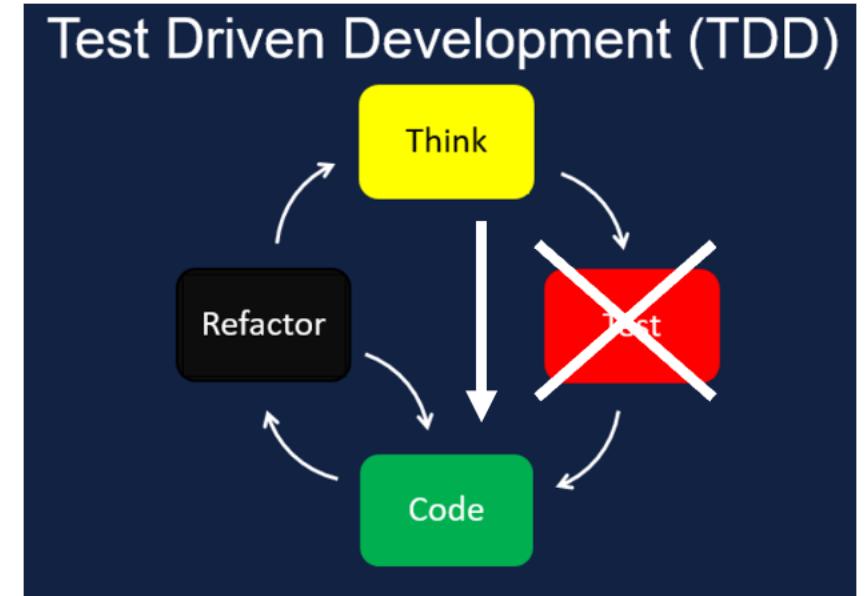
- **Think**

- Where to start?
- Create a **GREEN** test for the existing business logic **including DB access**
- Test average rating for DXTR1000 (= 4.5)

Data Preview

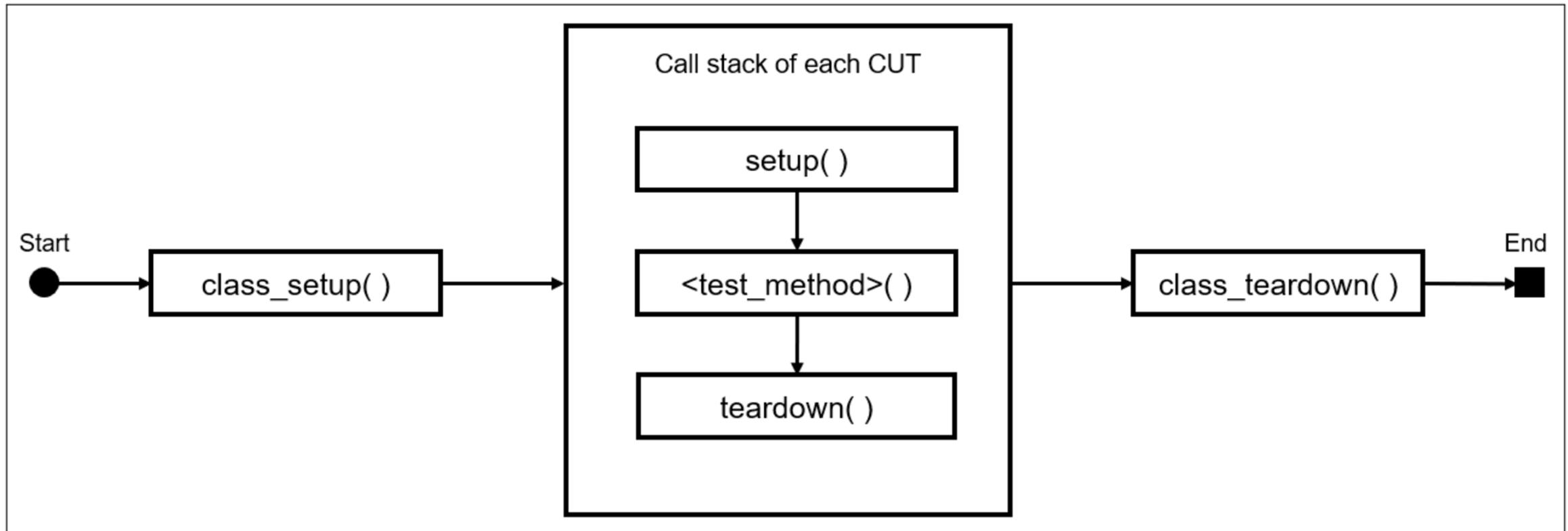
find pattern 6 rows retrieved - 20 ms

RB	CLIENT	RB	RATING_UUID	RB	PRODUCT	RB	NAME	RB	EMAIL	12	RATING	RB	REVIEW
100		E21FD1B74D261E	DXTR1000	Anna Bau	anna@bau					4	The color o		
100		E21FD1B74D261E	DXTR2000	Chris Dr	cd@test.d					5	The best bil		
100		E21FD1B74D261E	DXTR3000	Eve Smith	es@email.					3			
100		E21FD1B74D261E	EPAD1000	Frank Jon	f@jones.m					5	Size varies		
100		E21FD1B74D261E	EPAD1000	Gill Miller	g.miller@c					4			
100		E21FD1B74D261E	DXTR1000	G. Miller	g.miller@c					5			





- **ABAP Unit Test Framework**



Source: <https://developers.sap.com/tutorials/abap-environment-rap100-unit-testing..html>



• GREEN

- Debugger -> In-/Output Data
- DB access -> CDSs are fine

The screenshot shows an ABAP code editor with the following code:

```
1 CLASS ltcl_average_rating DEFINITION FINAL FOR TESTING
2   DURATION SHORT
3   RISK LEVEL HARMLESS.
4
5 PRIVATE SECTION.
6   DATA cut TYPE REF TO if_sadl_exit_calc_element_read.
7
8 METHODS:
9   setup.
10 METHODS:
11   " GIVEN: Input data for Product DXTR1000 WHEN: Calculate average rating THEN: ...
12   should_calculate_correctly FOR TESTING RAISING cx_static_check.
13
14 ENDCLASS.
15
16
17 METHOD setup.
18   cut = NEW zcl_ve_average_rating_mj( ).
19 ENDMETHOD.
20
21
22 METHOD should_calculate_correctly.
23   DATA products      TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.
24   DATA calculated_data TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.
25   DATA expected_values TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.
26
27   " Set Input Data
28   products      = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black' AverageRating = 0 ) ).  
29   calculated_data = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black' AverageRating = 4 ) ).  
30
31   " Set Expected Result
32   expected_values = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black' AverageRating = '4.5' ) ).  
33
34   " Execute calculation
35   cut->calculate( EXPORTING it_original_data      = products
36                     it_requested_calc_elements = VALUE if_sadl_exit_calc_element_read->tt_elements( )
37                     CHANGING ct_calculated_data = calculated_data ).  
38
39   " Compare expected and calculated data
40   cl_abap_unit_assert->assert_equals( exp = expected_values
41                                       act = calculated_data ).  
42
43
44
45 ENDMETHOD.
```

A blue arrow points from the text "AverageRating = 0" in line 29 to the value "0" in the status bar at the bottom of the screen. Another blue arrow points from the text "AverageRating = 4" in line 29 to the value "4" in the status bar. The status bar also shows "Methods: 1 Duration: 1.830 s".



- **Refactor (Magic time)**

- Extract “functional” method
- “Information hiding”
- “Best of three”

```
27 METHOD if_sadl_exit_calc_element_read~calculate.
28   DATA products TYPE STANDARD TABLE OF Z_C_Product_M_MJ.
29   DATA average_product_ratings TYPE tt_average_product_ratings.
30
31   products = CORRESPONDING #( it_original_data ).
32
33   " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
34   READ ENTITIES OF Z_C_Product_M_MJ
35     ENTITY Product BY \_Rating
36     FIELDS ( Rating )
37     WITH CORRESPONDING #( products )
38     RESULT FINAL(ratings).
39
40   " Calculate the average rating for each product by DB select
41   SELECT Product,
42     AVG( rating AS DEC( 2, 1 ) ) AS average_rating
43   FROM @ratings AS r
44   GROUP BY Product
45   INTO TABLE @average_product_ratings.
46
47   " Map average ratings to the products (output)
48   products = map_average_ratings2products( average_product_ratings = average_product_ratings
49                                         products           = products ).  
50
51   ct_calculated_data = CORRESPONDING #( products ).  
52
53 ENDMETHOD.
```

```
54 METHOD map_average_ratings2products.
55
56   result = products.
57
58 LOOP AT result ASSIGNING FIELD-SYMBOL(<product>).
59   <product>-AverageRating = average_product_ratings[ Product = <product>-ProductId ]-average_rating.
60 ENDLOOP.  
61
62 ENDMETHOD.  
63
```



- Refactor (Magic time)

- Extract “functional” method
- “Information hiding”
- “Best of three”

```
27 METHOD if_sadl_exit_calc_element_read~calculate.
28   DATA products TYPE STANDARD TABLE OF Z_C_Product_M_MJ.
29   DATA average_product_ratings TYPE tt_average_product_ratings.
30
31   products = CORRESPONDING #( it_original_data ).
32
33   " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
34   READ ENTITIES OF Z_C_Product_M_MJ
35     ENTITY Product BY \ Rating
36     FI PRIVATE SECTION.
37     WI TYPES:
38     RE ty_products TYPE STANDARD TABLE OF z_c_product_m_mj WITH DEFAULT KEY,
39
40   " Ca
41   SELE
42   FR
43   GR
44   IN
45
46   " Ma
47   prod
48
49
50
51   ct_calculated_data = CORRESPONDING #( products ).
52 ENDMETHOD.
```

```
54 METHOD map_average_ratings2products.
55
56   result = products.
57
58 LOOP AT result ASSIGNING FIELD-SYMBOL(<product>).
59   <product>-AverageRating = average_product_ratings[ Product = <product>-ProductId ]-average_rating.
60 ENDLOOP.
61
62 ENDMETHOD.
```



- Refactor (Magic time)

- Extract “functional” method
- “Information hiding”
- “Best of three”

```
27 METHOD if_sadl_exit_calc_element_read~calculate.
28   DATA products TYPE STANDARD TABLE OF Z_C_Product_M_MJ.
29   DATA average_product_ratings TYPE tt_average_product_ratings.
30
31   products = CORRESPONDING #( it_original_data ).
32
33   " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
34   READ ENTITIES OF Z_C_Product_M_MJ
35     ENTITY Product BY \ Rating
36     FI PRIVATE SECTION.
37     WI TYPES:
38     RE ty_products TYPE STANDARD TABLE OF z_c_product_m_mj WITH DEFAULT KEY,
39
40   " Ca
41   SELE
42   FR
43   GR
44   IN
45
46   " Ma
47   prod
48   ct_calculated_data = CORRESPONDING #( products ).
```

```
52 ENDMETHOD.
```

```
54 METHOD map_average_ratings2products.
55
56   result = products.
57
58 LOOP AT result ASSIGNING FIELD-SYMBOL(<product>).
59   <product>-AverageRating = average_product_ratings[ Product = <product>-ProductId ]-average_rating.
60 ENDLOOP.
61
62 ENDMETHOD.
```





- Refactor (Magic time)

- Extract “functional” method

```
9 PRIVATE SECTION.
10 TYPES tt_products TYPE STANDARD TABLE OF z_c_product_m_mj WITH DEFAULT KEY.
11 TYPES tt_ratings TYPE TABLE FOR READ RESULT z_c_product_m_mj\product\_rating.
12
13 TYPES: BEGIN OF ty_average_product_rating,
14     Product      TYPE z_c_product_m_mj-ProductId,
15     average_rating TYPE z_c_product_m_mj-AverageRating,
16   END OF ty_average_product_rating.
17
18 TYPES tt_average_product_ratings TYPE STANDARD TABLE OF ty_average_product_rating WITH DEFAULT KEY.
19
20 METHODS map_average_ratings2products IMPORTING average_product_ratings TYPE tt_average_product_ratings
21                               products          TYPE tt_products
22                               RETURNING VALUE(result) TYPE tt_products.
23
24 METHODS read_average_rating4products IMPORTING products      TYPE tt_products
25                               RETURNING VALUE(result) TYPE tt_average_product_ratings.
26
```

```
29 CLASS zcl_ve_average_rating_mj IMPLEMENTATION.
30
31 METHOD if_sadl_exit_calc_element_read~calculate.
32   DATA(products) = CORRESPONDING tt_products( it_original_data ).
33
34   DATA(average_product_ratings) = read_average_rating4products( products ).  
35
36   products = map_average_ratings2products( average_product_ratings = average_product_ratings
37                                     products           = products ).  
38
39   ct_calculated_data = CORRESPONDING #( products ).  
40 ENDMETHOD.  
41
42 METHOD read_average_rating4products.
43   " Read the ratings for the product(s) from the Z_C_Product_M_MJ entity
44   READ ENTITIES OF Z_C_Product_M_MJ
45     ENTITY Product BY \_Rating
46     FIELDS ( Rating )
47     WITH CORRESPONDING #( products )
48     RESULT FINAL(ratings).
49
50   " Calculate the average rating for each product by DB select
51   SELECT Product,
52         AVG( rating AS DEC( 2, 1 ) ) AS average_rating
53   FROM @ratings AS r
54   GROUP BY Product
55   INTO TABLE @result.
56 ENDMETHOD.
```



- **Refactor (Decoupling of DB dependency)**

- **For READ operations:**

Using CDS Test Environment standard class *cl_cds_test_environment*

- **For CUD operations:**

+ SQL Test Environment standard class *cl_sql_test_environment*



- **REFACTOR:
Mocking ENTITY
access**

```
4 CLASS ltcl_average_rating DEFINITION FINAL FOR TESTING
5   DURATION SHORT
6   RISK LEVEL HARMLESS.
7
8   PRIVATE SECTION.
9     CLASS-DATA:
10    cds_test_environment TYPE REF TO if_cds_test_environment,
11                           ratings           TYPE STANDARD TABLE OF zrating_mj.
12
13   CLASS-METHODS:
14     class_setup,
15     class_teardown.
16
17   DATA cut TYPE REF TO if_sadl_exit_calc_element_read.
18
19   METHODS:
20     setup,
21     teardown.
22
23   METHODS:
24     " GIVEN: Input data for Product DXTR1000 WHEN: Calculate average rating THEN: ...
25     should_calculate_correctly FOR TESTING RAISING cx_static_check.
26
27 ENDCLASS.
```



- **REFACTOR:
Mocking ENTITY
access**

```
29
30  METHOD class_setup.
31      " create the test doubles for the underlying CDS entities
32      cds_test_environment = cl_cds_test_environment->create(
33          i_for_entity = 'Z_C_RATING_M_MJ'
34          i_dependency_list = VALUE #( ( name = 'ZRATING_MJ' type = 'TABLE' ) ) .
35      ratings = VALUE #(
36          ( rating_uuid = '1' product = 'DXTR1000' rating = 4 )
37          ( rating_uuid = '2' product = 'DXTR1000' rating = 5 )
38          ( rating_uuid = '3' product = 'DXTR2000' rating = 5 ) .
39
40  ENDMETHOD.
41
40 METHOD class_teardown.
41     " remove test doubles
42     cds_test_environment->destroy( ).
```

```
4 CLASS ltcl_average_rating DEFINITION FINAL FOR TESTING
5     DURATION SHORT
6     RISK LEVEL HARMLESS.
7
8 PRIVATE SECTION.
9     CLASS-DATA:
10         cds_test_environment TYPE REF TO if_cds_test_environment,
11         ratings             TYPE STANDARD TABLE OF zrating_mj.
12
13     CLASS-METHODS:
14         class_setup,
15         class_teardown.
16
17     DATA cut TYPE REF TO if_cdl_exit_calc_element _read.
```

EN: Calculate average rating THEN: ...
ING cx_static_check.



- **REFACTOR:**
Mocking ENTITY access

```
29
30  METHOD class_setup.
31      " create the test doubles for the underlying CDS entities
32      cds_test_environment = cl_cds_test_environment->create(
33          i_for_entity = 'Z_C_RATING_M_MJ'
34          i_dependency_list = VALUE #( ( name = 'ZRATING_MJ' type = 'TABLE' ) ) .
35      ratings = VALUE #(
36          ( rating_uuid = '1' product = 'DXTR1000' rating = 4 )
37          ( rating_uuid = '2' product = 'DXTR1000' rating = 5 )
38          ( rating_uuid = '3' product = 'DXTR2000' rating = 5 ) .
39      ENDMETHOD.
40
41  METHOD class_teardown.
42      " remove test doubles
43      cds_test_environment->destroy( ).
```

```
40 CLASS ltcl_average_rating DEFINITION FINAL FOR TESTING
41     DURATION SHORT
42     RISK LEVEL HARMLESS.
43
44 PRIVATE SECTION.
45 CLASS-DATA:
46     cds_test_environment TYPE REF TO if_cds_test_environment,
47     ratings           TYPE STANDARD TABLE OF zrating_mj.
48
49 CLASS-METHODS:
50     class_setup,
51     class_teardown.
```

```
52     DATA cut TYPE REF TO if_cdl_exit_calc_element _read.
```

```
53
54  METHOD setup.
55      cut = NEW zcl_ve_average_rating_mj( ).
56
57      " clear the test doubles per test
58      cds_test_environment->clear_doubles( ).
```

```
59
60      " insert test data into test doubles
61      cds_test_environment->insert_test_data( ratings ).
```

```
62      ENDMETHOD.
```

```
63
64  METHOD teardown.
65      ROLLBACK ENTITIES. "#EC CI_ROLLBACK
66      ENDMETHOD.
```

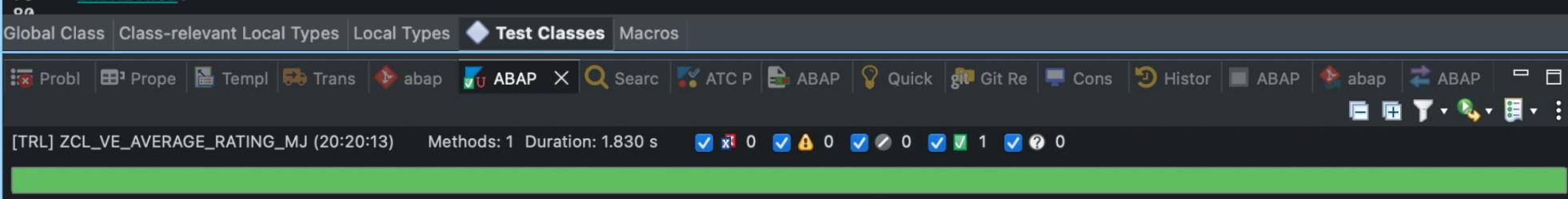
EN: Calculate average rating THEN: ...
ING cx_static_check.



- **REFACTOR:**
Mocking ENTITY access

- **Test method not changed**

```
59 o METHOD should_calculate_correctly.  
60   DATA products      TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.  
61   DATA calculated_data TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.  
62   DATA expected_values TYPE STANDARD TABLE OF Z_C_Product_M_MJ WITH DEFAULT KEY.  
63  
64   " Set Input Data  
65   products      = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black' AverageRating = 0 ) ).  
66   calculated_data = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black' AverageRating = 4 ) ).  
67  
68   " Set Expected Result  
69   expected_values = VALUE #( ( ProductId = 'DXTR1000' ProductDescription = 'Deluxe Touring Bike Black' AverageRating = '4.5' ) ).  
70  
71   " Execute calculation  
72   cut->calculate( EXPORTING it_original_data      = products  
73                           it_requested_calc_elements = VALUE if_sadl_exit_calc_element_read=>tt_elements( )  
74                           CHANGING ct_calculated_data    = calculated_data ).  
75  
76   " Compare expected and calculated data  
77   cl_abap_unit_assert=>assert_equals( exp = expected_values  
78                                         act = calculated_data ).  
79 ENDMETHOD.  
oo
```



The screenshot shows the SAP ABAP IDE interface. The code editor displays the ABAP code for the `should_calculate_correctly` method. The test results are shown in the status bar at the bottom: [TRL] ZCL_VE_AVERAGE_RATING_MJ (20:20:13) Methods: 1 Duration: 1.830 s with 1 error (red checkmark) and 0 warnings (blue checkmark).



How to know what to mock?

```
1 projection;
2 strict ( 1 );
3
4 define behavior for Z_C_Product_M_MJ alias Product
5 {
6   use create;
7   use update;
8   use delete;
9
10  use association _Rating { create; }
11 }
12
13 define behavior for Z_C_Rating_M_MJ alias Rating
14 {
15   use update;
16   use delete;
17
18   use action setStatusToCompleted;
19
20   use association _Product;
21 }
22
23 define view entity Z_C_RATING_M_MJ
24   as projection on Z_I_RATING_MJ
25 {
26   key RatingUUID,
27   Product,
28   Name,
29   Email,
30   Rating,
31   Review,
32   CreatedBy,
33   CreatedAt,
34   LastChangedBy,
35   LastChangedAt,
36   Status,
37   StatusCriticality,
38
39   /* Associations */
40   _Product : redirected to parent Z_C_Product_M_MJ
41 }
```

```
32 cds_test_environment = cl_cds_test_environment->create(
33   i_for_entity = 'Z_C_RATING_M_MJ'
34   i_dependency_list = VALUE #( ( name = 'ZRATING_MJ' type = 'TABLE' ) ) ).
```

```
61 READ ENTITIES OF Z_C_Product_M_MJ
62 ENTITY Product BY \_Rating
63 FIELDS ( Rating )
64 WITH CORRESPONDING #( products )
65 RESULT FINAL(ratings).
```



DEEP DIVE

```
4 define view entity Z_I_RATING_MJ
5   as select from zrating_mj
6   association to parent Z_I_Product_MJ
7 {
8     key rating_uuid      as RatingUUID,
9     product             as Product,
10    name                as Name,
11    email               as Email,
12    rating              as Rating,
13    review              as Review,
14    status              as Status,
```



- Refactoring is FUN and a super Learning “Tool” 😊

```
54● METHOD map_average_ratings2products.  
55  
56     result = products.  
57  
58● LOOP AT result ASSIGNING FIELD-SYMBOL(<product>).  
59     <product>-AverageRating = average_product_ratings[ Product = <product>-ProductId ]-average_rating.  
60   ENDOLOOP.  
61  
62 ENDMETHOD.  
63
```

```
58● METHOD map_average_ratings2products.  
59     result = products.  
60  
61● LOOP AT result ASSIGNING FIELD-SYMBOL(<product>).  
62     <product>-AverageRating = VALUE #( average_product_ratings[ Product = <product>-ProductId ]-average_rating OPTIONAL ).  
63   ENDOLOOP.  
64 ENDMETHOD.  
65
```

«Power of three»



Showcase – Check Rating

... please use numbers between 1 and 5!

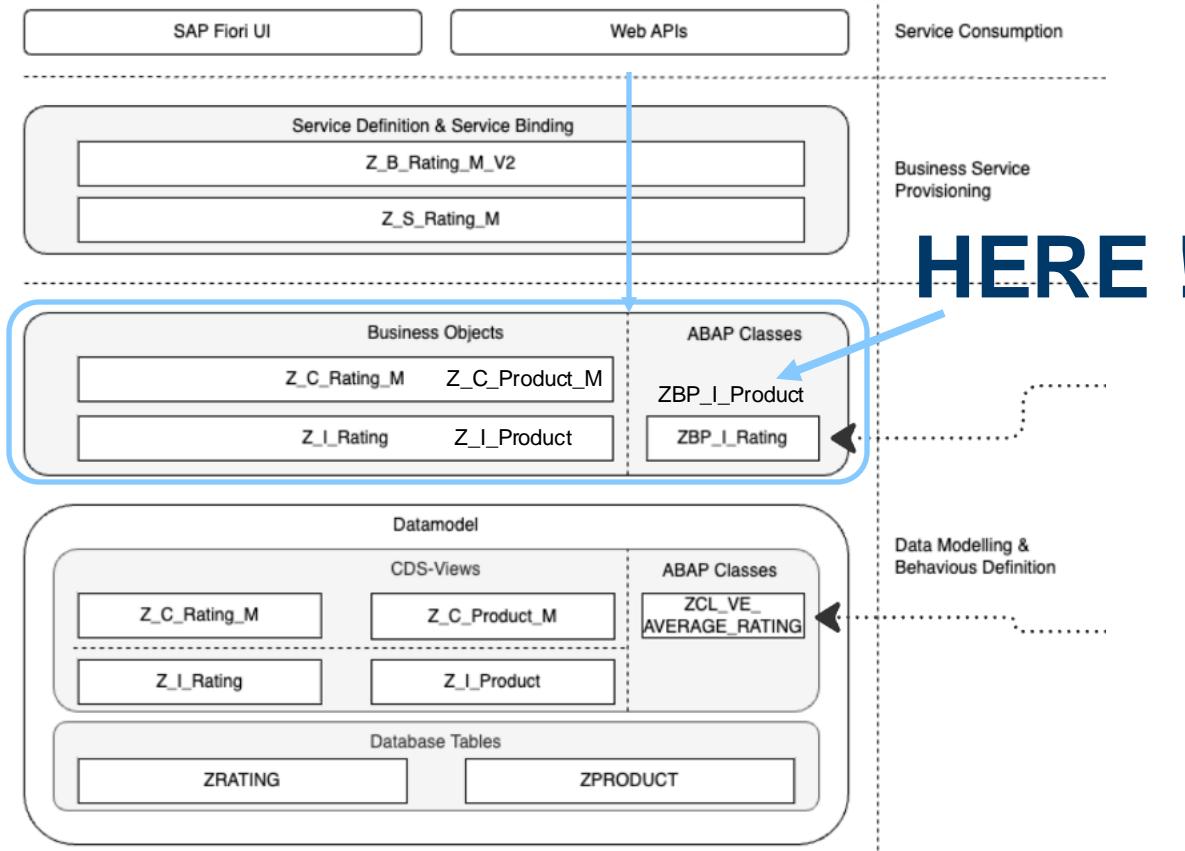
DXTR1000

Rating

UUID: e21fd1b7-4d26-1eef-aace-709fc448db29	Rating: <input type="text" value="7"/>	User Name: CB9980006539
Product ID: DXTR1000	Review: The color of the bike is not very durable.	Time Stamp: Nov 24, 2024, 3:51:00 PM
Name: Anna Bauer	User Name: CB9980006539	
Email: anna@bauer.de	Time Stamp: Nov 24, 2024, 3:51:00 PM	

Other Messages

✖ Rating values must be between 0 and 5.
Rating





Testing the Behavior Definition/Business Object logic

The screenshot shows the SAP Studio IDE interface. The left side features the Explorer view with a tree structure of project components, including Core Data Services, Source Code Library, and others. The right side is a code editor displaying behavior definitions for a class named `zbp_i_product_mj`. The code is annotated with several validation blocks:

```
1 Managed implementation in class zbp_i_product_mj unique;
2 strict ( 1 );
3
4 define behavior for Z_I_Product_MJ alias Product
5 persistent table zproduct_mj
6 lock master
7 authorization master ( instance )
8 {
9   create;
10  update;
11  delete;
12  association _Rating { create; }
13
14  field ( readonly : update ) ProductId;
15  mapping for zproduct_mj corresponding
16  {
17    ProductId          = product_id;
18    ProductDescription = product_desc;
19  }
20
21 }
22
23
24 define behavior for Z_I_Rating_MJ alias Rating
25 persistent table zrating_mj
26 lock dependent by _Product
27 authorization dependent by _Product
28 {
29   update ( features : instance );
30   delete ( features : instance );
31   field ( readonly ) Product;
32   association _Product;
33
34   field ( readonly, numbering : managed ) RatingUUID;
35   field ( readonly ) LastChangedAt, LastChangedBy, CreatedAt, CreatedBy;
36
37   validation checkEmail on save { field Email; }
38   validation checkRating on save { field Rating; }
39
40   determination setStatusNew on modify { create; }
41   determination setStatusCustomerFeedback on modify { field Rating; }
42
43   action ( features : instance ) setStatusToCompleted result [1] $self;
44   mapping for zrating_mj corresponding
45   {
46     RatingUUID      = rating_uuid;
47     LastChangedAt   = last_changed_at;
48     LastChangedBy   = last_changed_by;
49     CreatedAt       = created_at;
50     CreatedBy       = created_by;
51   }
52 }
```



Testing the Behavior Definition/Business Object logic

```
Managed implementation in class zbp_i_product_mj unique;
strict ( 1 );

define behavior for Z_I_Product_MJ alias Product
persistent table zproduct_mj
lock master
authorization master / instance

{
  create;
  update;
  delete;
  association _Rating;
  field ( readonly
    mapping for zrating_mj
    ProductId
    ProductDescriptor
  );
}

CLASS zbp_i_product_mj DEFINITION PUBLIC ABSTRACT FINAL FOR BEHAVIOR OF z_i_product_mj .
ENDCLASS.

CLASS ZBP_I_PRODUCT_MJ IMPLEMENTATION.
ENDCLASS.

define behavior for Z_I_Rating_MJ alias Rating
persistent table zrating_mj
lock dependent by _Product
authorization dependent by _Product
{
  update ( features : instance );
  delete ( features : instance );
  field ( readonly ) Product;
  association _Product;

  field ( readonly, numbering : managed ) RatingUUID;
  field ( readonly ) LastChangedAt, LastChangedBy, CreatedAt, CreatedBy;

  validation checkEmail on save { field Email; }
  validation checkRating on save { field Rating; }

  determination setStatusNew on modify { create; }
  determination setStatusCustomerFeedback on modify { field Rating; }

  action ( features : instance ) setStatusToCompleted result [1] $self;
  mapping for zrating_mj corresponding
  {
    RatingUUID      = rating_uuid;
    LastChangedAt   = last_changed_at;
    LastChangedBy   = last_changed_by;
    CreatedAt       = created_at;
    CreatedBy       = created_by;
  }
}
```



Testing the Behavior Definition/Business Object logic

The screenshot shows the SAP ABAP IDE interface with three code editors:

- Left Editor:** Shows the Z_RATING_DB database model. A class named "Z_I_PRODUCT_MJ" is selected.
- Middle Editor:** Displays the source code for the class "ZBP_I_PRODUCT_MJ". Annotations highlight specific parts of the code:
 - "Managed implementation in class zbp_i_product_mj unique;"
 - "CLASS zbp_i_product_mj DEFINITION PUBLIC ABSTRACT FINAL FOR BEHAVIOR OF z_i_product_mj."
 - "CLASS ZBP_I_PRODUCT_MJ IMPLEMENTATION."
- Right Editor:** Displays the source code for the class "lhc_rating". Annotations highlight specific parts of the code:
 - "CLASS lhc_rating DEFINITION INHERITING FROM cl_abap_behavior_handler"
 - "PRIVATE SECTION."
 - "CONSTANTS:"
 - BEGIN OF rating_status,
 - new TYPE i VALUE 10,
 - customer_feedback TYPE i VALUE 20,
 - completed TYPE i VALUE 30,
 - END OF rating_status.
 - "METHODS check_email FOR VALIDATE ON SAVE IMPORTING keys FOR rating~checkemail."
 - "METHODS check_rating FOR VALIDATE ON SAVE IMPORTING keys FOR rating~checkrating."

Annotations in the middle editor point to the corresponding code in the right editor, illustrating how behavior definitions map to their implementations.



How to UNIT test this scenario?

- **Abstract class is not accessible, but ...**
 - **FRIENDS** are giving us a helping hand

```
ZBP_I_PRODUCT_MJ ▶ LHC_RATING ▶ LTCL_I_PRODUCT_MJ
1 CLASS ltcl_i_product_mj DEFINITION DEFERRED FOR TESTING.
2 • CLASS lhc_rating DEFINITION INHERITING FROM cl_abap_behavior_handler FRIENDS ltcl_i_product_mj.
3
```



• THINK

- Test for Rating 5

```
ZBP_I_PRODUCT_MJ > LHC_RATING > LTCL_I_PRODUCT_MJ
1 CLASS ltcl_i_product_mj DEFINITION DEFERRED FOR TESTING.
2 CLASS lhc_rating DEFINITION INHERITING FROM cl_abap_behavior_handler FRIENDS ltcl_i_product_mj.
3
4 PRIVATE SECTION.
5
6 CONSTANTS:
7   BEGIN OF rating_status,
8     new          TYPE i VALUE 10,
9     customer_feedback TYPE i VALUE 20,
10    completed      TYPE i VALUE 30,
11   END OF rating_status.
12
13 METHODS check_email FOR VALIDATE ON SAVE
14   IMPORTING keys FOR rating~checkemail.
15 METHODS check_rating FOR VALIDATE ON SAVE
16
METHOD check_rating.
  READ ENTITIES OF z_i_product_mj IN LOCAL MODE
    ENTITY rating
      FIELDS ( rating )
      WITH CORRESPONDING #( keys )
    RESULT DATA(ratings).
  LOOP AT ratings ASSIGNING FIELD-SYMBOL(<rating>).
    IF <rating>-rating < 0 OR <rating>-rating > 5.
      APPEND VALUE #( %key = <rating>-%key ) TO failed-rating.
      APPEND VALUE #( %key = <rating>-%key
                     %msg = new_message( id      = 'ZM_RATING_M'
                                         number = '002'
                                         severity = if_abap_behv_message=>severity-error )
                     %element-rating = if_abap_behv=>mk-on ) TO reported-rating.
    ENDIF.
  ENDLOOP.
ENDMETHOD.
```



- **GREEN**

```
► ZBP_I_PRODUCT_MJ ► LTCL_I_PRODUCT_MJ ► SHOULD_ACCEPT_RATING_5
1 CLASS ltcl_i_product_mj DEFINITION FINAL FOR TESTING
2   DURATION SHORT
3   RISK LEVEL HARMLESS.
4
5 PRIVATE SECTION.
6   CLASS-DATA:
7     cut          TYPE REF TO lhc_rating. " Must be static, because local class is static
8
9   CLASS-METHODS:
10    "! Instantiate class under test and setup test double frameworks
11    class_setup,
12    "! Destroy test environments and test doubles
13    class_teardown.
14
15  METHODS:
16    "! Reset test doubles
17    setup,
18    "! Reset transactional buffer
19    teardown.
20  METHODS:
21    " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...
22    should_accept_rating_5 FOR TESTING RAISING cx_static_check.
23
24 ENDCLASS.
```



- **GREEN**

```
► ZBP_I_PRODUCT_MJ ► LTCL_I_PRODUCT_MJ ► SHOULD_ACCEPT_RATING_5
1 CLASS ltcl_i_product_mj DEFINITION FINAL FOR TESTING
2   DURATION SHORT
3   RISK LEVEL HARMLESS.
4
5 PRIVATE SECTION.
6   CLASS-DATA:
7     cut          TYPE REF TO lhc_rating. " Must be static, because local class is static
8
9   CLASS-METHODS:
```

```
METHOD class_setup.
  CREATE OBJECT cut FOR TESTING.
ENDMETHOD.
```

```
15 METHODS:
16   "! Reset test doubles
17   setup,
18   "! Reset transactional buffer
19   teardown.
20 METHODS:
21   " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...
22   should_accept_rating_5 FOR TESTING RAISING cx_static_check.
23
24 ENDCCLASS.
25
```



- GREEN

```
► ZBP_I_PRODUCT_MJ ► LTCL_I_PRODUCT_MJ ► SHOULD_ACCEPT_RATING_5
1 CLASS ltcl_i_product_mj DEFINITION FINAL FOR TESTING
2   DURATION SHORT
3   RISK LEVEL HARMLESS.
4
5 PRIVATE SECTION.
6   CLASS-DATA:
7     cut          TYPE REF TO lhc_rating. " Must be static, because local class is static
8
9   CLASS-METHODS:
```

```
METHOD class_setup.
  CREATE OBJECT cut FOR TESTING.
ENDMETHOD.
```

```
15 METHODS:
16   "! Reset test doubles
17   setup,
18   "! Reset transactional buffer
19   teardown.
20 METHODS:
21   " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...
22   should_accept_rating_5 FOR TESTING RAISING cx_static_check.
23
```

```
42 METHOD should_accept_rating_5.
43   DATA keys    TYPE TABLE FOR VALIDATION z_i_product_mj\|rating~checkrating.
44   DATA failed   TYPE RESPONSE FOR FAILED LATE z_i_product_mj.
45   DATA reported  TYPE RESPONSE FOR REPORTED LATE z_i_product_mj.
46
47   " Call method to be tested
48   keys = VALUE #( ( RatingUuid = 'E21FD1B74D261EEFAACE709FC448FB29' ) ). " Table entry in ZRATING_MJ existing with this UUID
49   cut->check_rating( EXPORTING keys      = CORRESPONDING #( keys )
50                       CHANGING failed    = failed
51                       reported = reported ).
52
53   cl_abap_unit_assert=>assert_initial( failed ).
54   cl_abap_unit_assert=>assert_initial( reported ).
```



- GREEN

```
► ZBP_I_PRODUCT_MJ ► LTCL_I_PRODUCT_MJ ► SHOULD_ACCEPT_RATING_5
1 CLASS ltcl_i_product_mj DEFINITION FINAL FOR TESTING
2   DURATION SHORT
3   RISK LEVEL HARMLESS.
4
5   PRIVATE SECTION.
6     CLASS-DATA:
7       cut          TYPE REF TO lhc_rating. " Must be static, because local class is static
8
9   CLASS-METHODS:
```

```
METHOD class_setup.
  CREATE OBJECT cut FOR TESTING.
ENDMETHOD.
```

```
15   METHODS:
16     "! Reset test doubles
17     setup,
18     "! Reset transactional buffer
19     teardown.
20   METHODS:
21     " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...
22     should_accept_rating_5 FOR TESTING RAISING cx_static_check.
23
```

```
42 METHOD should_accept_rating_5.
43   DATA keys    TYPE TABLE FOR VALIDATION z_i_product_mj\|rating~checkrating.
44   DATA failed   TYPE RESPONSE FOR FAILED LATE z_i_product_mj.
45   DATA reported  TYPE RESPONSE FOR REPORTED LATE z_i_product_mj.
46
47   " Call method to be tested
48   keys = VALUE #( ( RatingUuid = 'E21FD1B74D261EEFAACE709FC448FB29' ) ). " Table entry in ZRATING_MJ existing with this UUID
49   cut->check_rating( EXPORTING keys      = CORRESPONDING #( keys )
50                       CHANGING failed    = failed
51                       reported = reported ).
52
53   cl_abap_unit_assert=>assert_initial( failed ).
54   cl_abap_unit_assert=>assert_initial( reported ).
```

05.12.2024 11:38

[TRL] ZCL_VE_AVERAGE_RATING_MJ (20:20:13) Methods: 1 Duration: 1.830 s ✓ ✘ 0 ✓ ✘ 0 ✓ ✘ 0 ✓ ✘ 1 ✓ ✘ 0



• REFACTOR

- Extract “functional” method

```
27 METHODS is_rating_valid IMPORTING rating          TYPE ze_rating
28                                     RETURNING VALUE(result) TYPE abap_bool.
29
30
64 METHOD check_rating.
65   READ ENTITIES OF z_i_product_mj IN LOCAL MODE
66     ENTITY rating
67       FIELDS ( rating )
68       WITH CORRESPONDING #( keys )
69     RESULT DATA(ratings).
70
71 LOOP AT ratings ASSIGNING FIELD-SYMBOL(<rating>)
72   IF is_rating_valid( <rating>-rating ) = abap_false.
73     APPEND VALUE #( %key = <rating>-key ) TO failed_rating.
74
75     APPEND VALUE #( %key = <rating>-key
76                   %msg = new_message( id      = 'ZM_RATING_M'
77                               number = '002'
78                               severity = if_abap_behv_message=>severity-error )
79                   %element-rating = if_abap_behv=>mk-on ) TO reported-rating.
80   ENDIF.
81 ENDLOOP.
82 ENDMETHOD.
```



```
84 METHOD is_rating_valid.
85   result = xsdbool( rating >= 1 AND rating <= 5 ).
86 ENDMETHOD.
```





- Refactor: Mocking ENTITY access

```
1 CLASS ltcl_i_product_mj DEFINITION FINAL FOR TESTING
2   DURATION SHORT
3   RISK LEVEL HARMLESS.
4
5   PRIVATE SECTION.
6     CLASS-DATA:
7       cut          TYPE REF TO lhc_rating, " Must be static, because local class is static
8       cds_test_environment TYPE REF TO if_cds_test_environment.
9
10    CLASS-METHODS:
11      "! Instantiate class under test and setup test double frameworks
12      class_setup,
13      "! Destroy test environments and test doubles
14      class_teardown.
15
16    DATA:
17      keys          TYPE TABLE FOR VALIDATION z_i_product_mj\`rating~checkrating,
18      rating_mock_data TYPE STANDARD TABLE OF zrating_mj,
19      failed        TYPE RESPONSE FOR FAILED LATE z_i_product_mj,
20      reported      TYPE RESPONSE FOR REPORTED LATE z_i_product_mj.
21
22    METHODS:
23      "! Reset test doubles
24      setup,
25      "! Reset transactional buffer
26      teardown.
27    METHODS:
28      " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...
29      should_accept_rating_5 FOR TESTING RAISING cx_static_check.
30
31 ENDCLASS.
```



- Refactor: Mocking ENTITY access

```
33 CLASS ltcl_i_product_mj IMPLEMENTATION.  
34  
35 METHOD class_setup.  
36   CREATE OBJECT cut FOR TESTING.  
37  
38   cds_test_environment = cl_cds_test_environment->create( i_for_entity = 'Z_I_RATING_MJ'  
39                           i_dependency_list = VALUE #( ( name = 'ZRATING_MJ' type = 'TABLE' ) ) ).  
40 ENDMETHOD.  
41  
42 METHOD class_teardown.  
43   cds_test_environment->destroy( ).  
44 ENDMETHOD.  
45  
46 METHOD setup.  
47   cds_test_environment->clear_doubles( ).  
48 ENDMETHOD.  
49  
50 METHOD teardown.  
51   ROLLBACK ENTITIES.          "#EC CI_ROLLBACK  
52 ENDMETHOD.  
53  
54 METHOD should_accept_rating_5.  
55   " Setup Rating Mock Data  
56   rating_mock_data = VALUE #( ( rating_uuid = 'E21FD1B74D261EEFAACE709FC448FB29' product = 'DXTR1000' rating = '5' ) ).  
57   cds_test_environment->insert_test_data( rating_mock_data ).  
58  
59   " Call method to be tested  
60   keys = VALUE #( ( RatingUuid = 'E21FD1B74D261EEFAACE709FC448FB29' ) ). " Table entry in ZRATING_MJ existing with this UUID  
61   cut->check_rating( EXPORTING keys      = CORRESPONDING #( keys )  
62             CHANGING failed    = failed  
63             reported     = reported ).  
64  
65   cl_abap_unit_assert->assert_initial( failed ).  
66   cl_abap_unit_assert->assert_initial( reported ).  
67 ENDMETHOD.  
68  
69 ENDCLASS.
```

frameworks

_mj\\rating~checkrating,
duct_mj,
product_mj.

executed THEN: ...
check.



- Refactor: Mocking ENTITY access

```
10 CLASS ltcl_i_product_mj DEFINITION FINAL FOR TESTING
11   DURATION SHORT
12   RISK LEVEL HARMLESS.
13
14   PRIVATE SECTION.
15   CLASS-DATA:
16     cut          TYPE REF TO lhc_rating, " Must be static, because local class is static
17     cds_test_environment TYPE REF TO if_cds_test_environment.
18
19
20 CLASS ltcl_i_product_mj IMPLEMENTATION.
21
22 METHOD class_setup.
23   CREATE OBJECT cut FOR TESTING.
24
25   cds_test_environment = cl_cds_test_environment->create( i_for_entity = 'Z_I_RATING_MJ'
26                                         i_dependency_list = VALUE #( ( name = 'ZRATING_MJ' type = 'TABLE' ) ) ).
```

33 METHOD class_teardown.

34 cds_test_environment->destroy().

35 ENDMETHOD.

36

37 METHOD setup.

38 cds_test_environment->clear_doubles().

39 ENDMETHOD.

40

41 METHOD teardown.

42 ROLLBACK ENTITIES. "#EC CI_ROLLBACK

43 ENDMETHOD.

44

45

46 METHOD should_accept_rating_5.

47 " Setup Rating Mock Data

48 rating_mock_data = VALUE #((rating_uuid = 'E21FD1B74D261EEFAACE709FC448FB29' product = 'DXTR1000' rating = '5')).

49 cds_test_environment->insert_test_data(rating_mock_data).

50

51 " Call method to be tested

52 keys = VALUE #((RatingUuid = 'E21FD1B74D261EEFAACE709FC448FB29')). " Table entry in ZRATING_MJ existing with this UUID

53 cut->check_rating(EXPORTING keys = CORRESPONDING #(keys)
54 CHANGING failed = failed
55 reported = reported).

56
57 cl_abap_unit_assert->assert_initial(failed).

58 cl_abap_unit_assert->assert_initial(reported).

59 ENDMETHOD.

60 ENDCLASS.

frameworks

_mj\\rating~checkrating,
duct_mj,
product_mj.

executed THEN: ...
check.



- **THINK -> GREEN (2nd testcase)**

```
27 METHODS:  
28     " GIVEN: Rating Input of value 5 WHEN: Check_Rating is executed THEN: ...  
29     should_accept_rating_5 FOR TESTING RAISING cx_static_check,  
30     " GIVEN: Rating Input of value 6 WHEN: Check_Rating is executed THEN: ...  
31     should_reject_rating_6 FOR TESTING RAISING cx_static_check.  
32
```

```
71 METHOD should_reject_rating_6.  
72     " Setup Rating Mock Data  
73     rating_mock_data = VALUE #( ( rating_uuid = 'E21FD1B74D261EEFAACE709FC448FB29' product = 'DXTR1000' rating = '6' ) ).  
74     cds_test_environment->insert_test_data( rating_mock_data ).  
75     " Call method to be tested  
76     keys = VALUE #( ( RatingUuid = 'E21FD1B74D261EEFAACE709FC448FB29' ) ). " Table entry in ZRATING_MJ existing with this UUID  
77     cut->check_rating( EXPORTING keys      = CORRESPONDING #( keys )  
78                     CHANGING failed    = failed  
79                     reported   = reported ).  
80     cl_abap_unit_assert=>assert_not_initial failed ).  
81     cl_abap_unit_assert=>assert_not_initial reported ).  
82 ENDMETHOD.
```

Check values?



What we learned

... uff!

- **UNIT test framework** in context with business logic in RAP
 - via *Data Definitions*
 - via *Behavior Definition*
- How to **mock ENTITY access** by SAP Standard Class
cl_cds_test_environment
- **Refactoring techniques:**
 - Extract (functional) method (information hiding)
 - «Power of three»
- **Test After Development** for legacy code (not TDD)



What is there to discover

... puh!

- Try it out on your own
 - GitHub Repo: <https://github.com/MaddinJay/ABAPConf24---UNIT-Test-Your-Business-Objects>

Commits

History for ABAPConf24---UNIT-Test-Your-Business-Objects / src on main

All users All time

Commits on Nov 25, 2024

Behavior Implementation Z_I_PRODUCT_MJ - Create Unit Test for rejected rating with value 6 b2d3c5ed-85c0-42f4-a98f-048... committed 5 days ago	e716fb2			
Behavior Implementation Z_I_PRODUCT_MJ - Introducing of Mock Entity Z_I_RATING_MJ b2d3c5ed-85c0-42f4-a98f-048... committed 5 days ago	302d0a9			
Behavior Implementation Z_I_PRODUCT_MJ - Create first Unit Test for existing DB entry in ZRATING_MJ with valid Rating b2d3c5ed-85c0-42f4-a98f-048... committed 5 days ago	e5f4c42			
Behavior Implementation Z_I_PRODUCT_MJ - Create Testclass with FRIENDS concept b2d3c5ed-85c0-42f4-a98f-048... committed 5 days ago	dcac063			
Average Rating - Refactoring Domain and Test Class b2d3c5ed-85c0-42f4-a98f-048... committed 5 days ago	654d44a			
Average Rating - Mock CDS Entity to Unit test Method Calculate b2d3c5ed-85c0-42f4-a98f-048... committed 5 days ago	a9e5131			
Average Rating - Extract Entity Read and Average Rating Select in separate method b2d3c5ed-85c0-42f4-a98f-048... committed 5 days ago	b2436b4			
Average Rating - Refactoring Mapping Average Ratings to Output "Products" b2d3c5ed-85c0-42f4-a98f-048... committed 5 days ago	ef3426f			
Average Rating - Create Test Method + structure method Calculate b2d3c5ed-85c0-42f4-a98f-048... committed 5 days ago	55397ee			
Initial Commit - Copy of Repository https://github.com/ceedee666/modern-abap-curriculum.git b2d3c5ed-85c0-42f4-a98f-048... committed 5 days ago	37e9f6e			

End of commit history for this file





What is there to discover

... puh!

- Mock the CUD ENTITY access and Business Objects itself
 - SHOW CASE - <https://developers.sap.com/tutorials/abap-environment-rap100-unit-testing..html>
 - Search for **/DMO/*TRAVEL*** Behavior Definitions with **UNIT Tests** 😱
 - https://github.com/SAP-samples/abap-cheat-sheets/blob/main/14_ABAP_Unit_Tests.md
- ACCCH-Challenge: Help me to update the Github
 - DM via LinkedIn





Martin Jonen Consulting

www.abapconf.org | #abapconf

December 5th 2024



```
DATA(ls_structure) TYPE IS_STRUCTURE.  
ASSERT ls_structure IS NOT INITIAL.  
get_SINITC IMPORTING ls_struct IS STRUCTURE.  
DELETE ls_struct-attributes USING xs_attributes.  
  
LOOP AT it_property_binding ASSIGNING  
    ls_attribute = VA.  
    IF xs_property_binding-is_key = 'X'.  
        ls_attribute-is_key = 'X'.  
    ENDIF.  
    INSERT ls_attribute INTO TABLE ls_struct.  
ENDLOOP.
```

Thank you!

Unit Test your Business Objects in RAP

Martin Jonen

*13:30 - 14:20 CET
Channel 1
English*

