SAP ABAP Programozás Alapjai

Második gyakorlat feladatainak egy lehetséges megoldása

A kiadott jegyzetben a feladatok egy lehetséges megoldását mutatom be. A feladat megoldásai során a gyakorlaton használt elnevezési koncepciótól eltértem, de az könnyen leképezhető a hallgató által használtra.

A megoldások ismertetése során a forráskódot angol változónevekkel és angol nyelvű kommentekkel láttam el. A gyakorlat felépítéséhez hasonlóan a feladatok egymásre épülése miatt a magyarázó kommenteket is egymásra épülve ismertetem, azaz egy már ismert magyarázatot nem feltétlenül ismertetek még egyszer egy nagyobb számú feladatnál.

Minden megoldás során az osztály forráskódját és a konzolra írt kimenetet ismertetem!

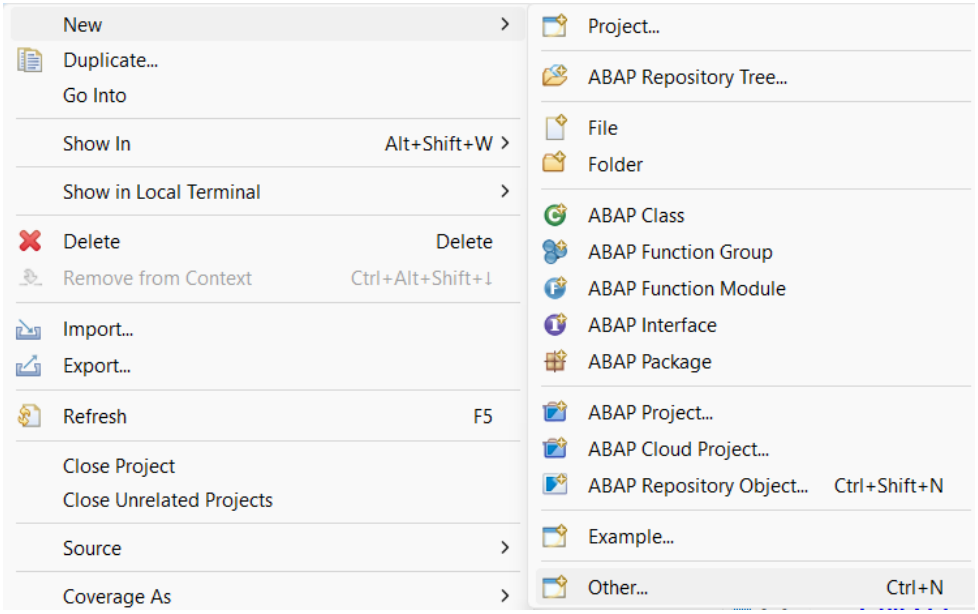
Tartalomjegyzék

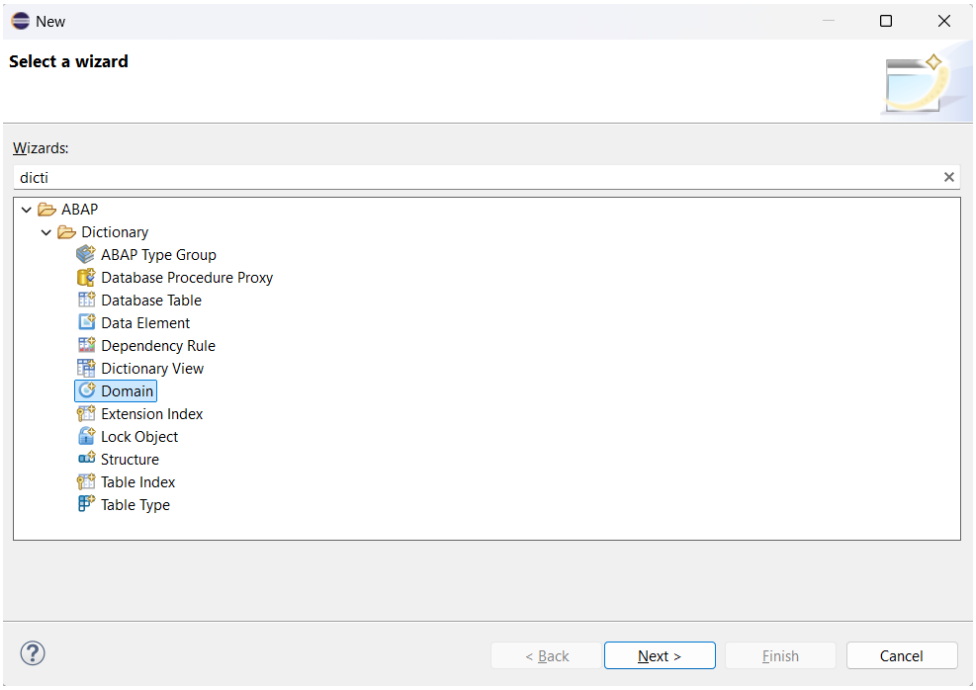
1. Feladat – ABAP Dictionary modellezés hallgatói szemeszter adatokhoz

Domainek létrehozása a leírás alapján

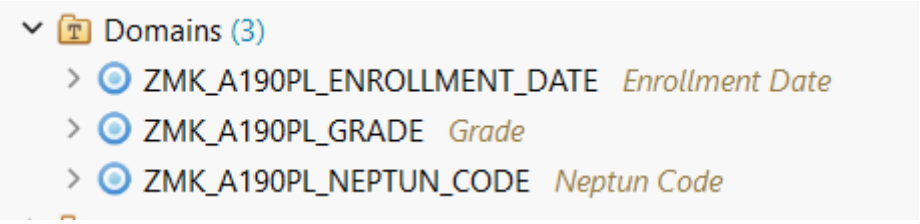
A domaineket az ADT eszközben hoztuk létre.

Létrehozás egyik lehetséges módja a projekten felhívott kontext menü, majd a New -> Other... választása

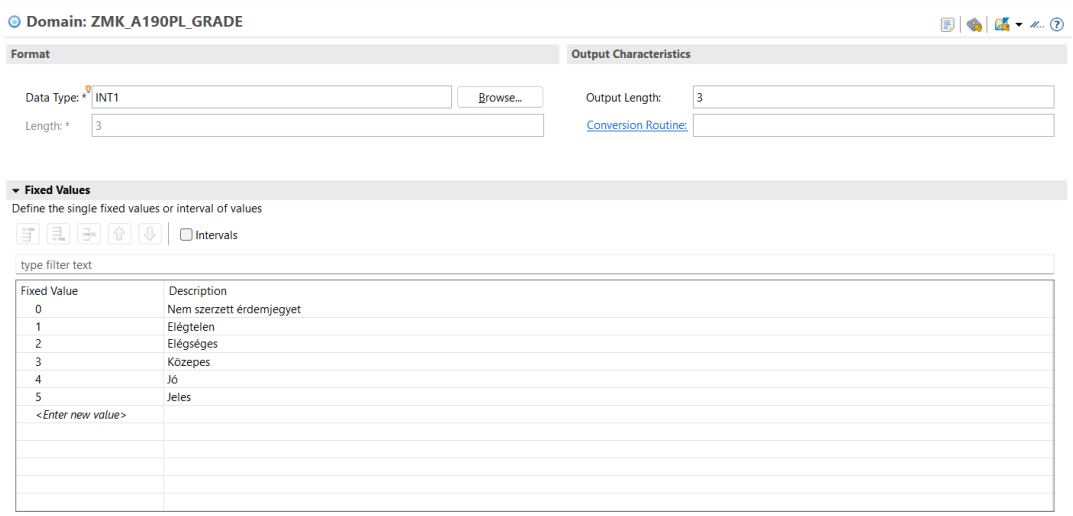




Példa létrehozott domainekre:

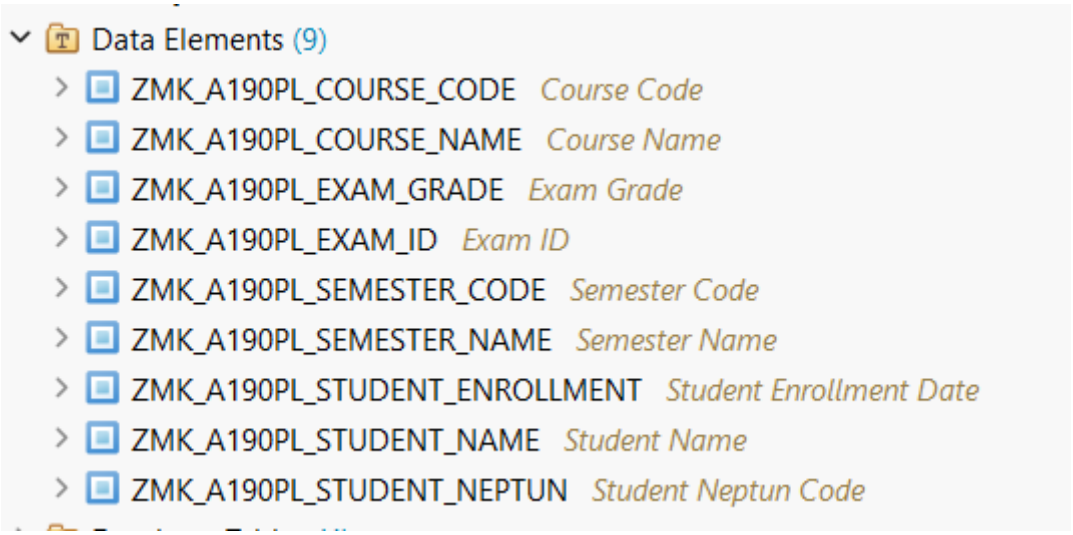


Példa az érdemjegy domain esetén:



Data elementek létrehozása a leírás alapján

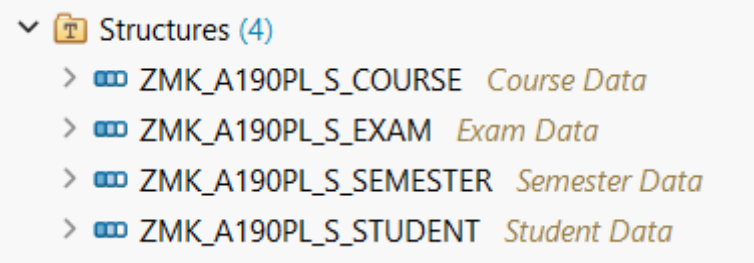
Az adatelemeket ADT-ben hoztuk létre, a domainekhez hasonló elérési módon. A létrehozott data elementek egy lehetséges listája:



Data elementek esetén vagy létrehozott domain-re, vagy beépített típusra hivatkoztunk, a gyakorlaton közösen kitalált adatmodellezési megközelítéskor hozott döntések alapján.

Struktúrák és tábla típusok létrehozása

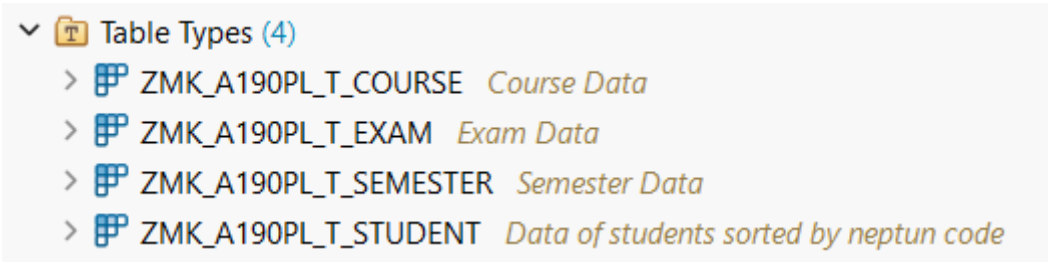
ADT-ben forráskód nézetben volt mód adatszótárban struktúrák definiálására, melyhez a domain és data-elementhez hasonló módon először az „structure” típusú adatszótárbeli elemet hoztuk létre. A létrehozott struktúrák egy lehetséges listája:



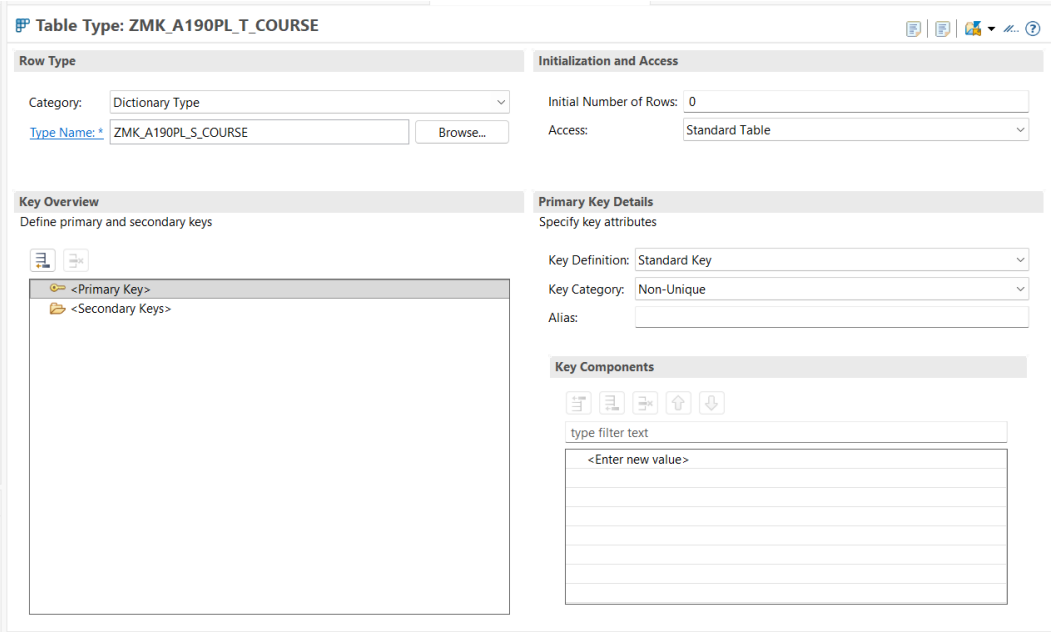
Példa forráskód egy struktúrára:

```
@EndUserText.label : 'Course Data'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
define structure zmk_a190pl_s_course {
  code : zmk_a190pl_course_code;
  name : zmk_a190pl_course_name;
}
```

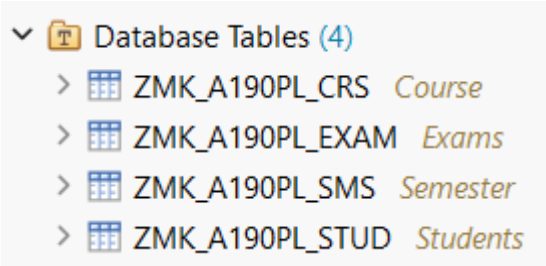
Táblatípusokat az ADT-ben űrlap segítségével hoztunk létre, hasonlóan a korábbi adatszótárbeli elemekhez. A létrehozott táblatípusok egy lehetséges listája:



Példa egy táblatípus meghatározására:



Adatbázis táblázatok létrehozása ADT-ben forráskód létrehozásával történt, miután a korábbi adatszótárbeli elemekhez hasonlóan az adatbázis tábla objektumát létrehoztuk. Egy lehetséges lista az adatbázis táblázatokról:



Egy példa adatbázistábla forráskódja

```
@EndUserText.label : 'Course'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #A
@AbapCatalog.dataMaintenance : #RESTRICTED
define table zmk_a190pl_crs {
  key client : abap.clnt not null;
  code      : zmk_a190pl_course_code;
  name      : zmk_a190pl_course_name;

}
```

2. Feladat – Adatbázis feltöltése

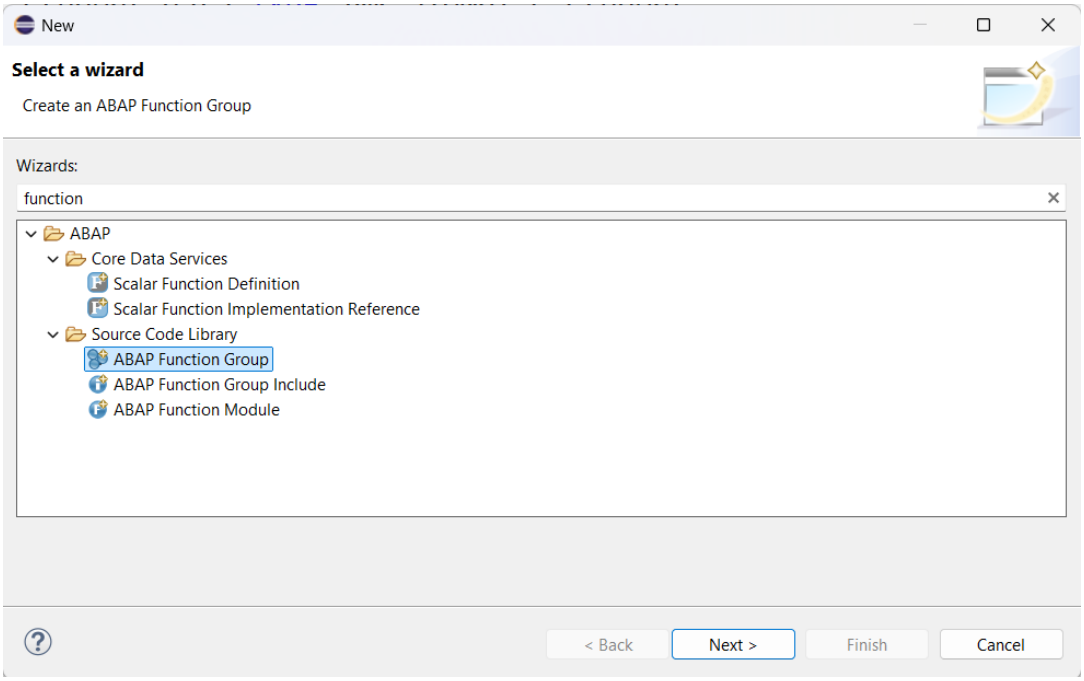
Az ADT konzol alkalmazásként futtatható ABAP osztály létrehozása az első gyakorlaton megtanult módon történt. A példa forráskód:

```
CLASS zmk_cl_uni_2026_init_db DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.

CLASS zmk_cl_uni_2026_init_db IMPLEMENTATION.

  METHOD if_oo_adt_classrun~main.

    DATA: students  TYPE TABLE OF zmk_a190pl_stud,
          courses   TYPE TABLE OF zmk_a190pl_crs,
          semesters TYPE TABLE OF zmk_a190pl_sms,
          exams     TYPE TABLE OF zmk_a190pl_exam.

    students = VALUE #(
        ( neptun = 'ABC123' name = 'Gipsz Jakab'
          enrollment_date = '20250101' )
        ( neptun = 'XYZ555' name = 'Kovács M. Ágnes'
           enrollment_date = '20250101' )
        ( neptun = 'UBULKA' name = 'Ubul'
           enrollment_date = '20250101' )
     ).

    courses = VALUE #(
        ( code = 'ME_GEIAK_001' name = 'Analízis 1.' )
        ( code = 'ME_GEIAK_002' name = 'Analízis 2.' )
        ( code = 'ME_GEIAK_003' name = 'Ethernial Beta Softwers' )
     ).

    semesters = VALUE #(
        ( code = '2025202601' name = '2025/06 őszi félév' )
        ( code = '2025202602' name = '2025/06 tavaszi félév' )
    ).

    exams = VALUE #(
```

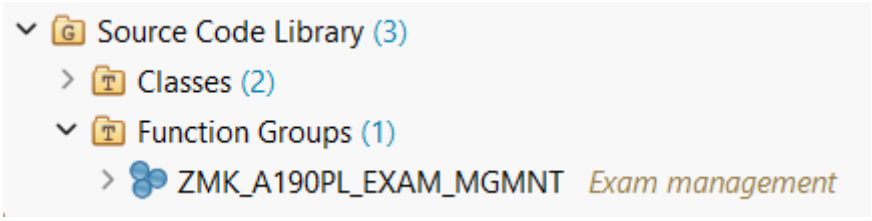
```
        ( exam_id = '000000101' semester_code = '2025202601'
          course_code = 'ME_GEIAK_001' student_neptun = 'ABC123'
          grade = 5 )
        ( exam_id = '000000134' semester_code = '2025202602'
          course_code = 'ME_GEIAK_002' student_neptun = 'ABC123'
          grade = 5 )
        ( exam_id = '000000222' semester_code = '2025202602'
          course_code = 'ME_GEIAK_003' student_neptun = 'ABC123'
          grade = 2 )
        ( exam_id = '000009999' semester_code = '2025202602'
          course_code = 'ME_GEIAK_001' student_neptun = 'UBULKA'
          grade = 0 )
    ).

    " First empty the tables
    DELETE FROM zmk_a190pl_stud WHERE neptun IS NOT INITIAL.
    DELETE FROM zmk_a190pl_crs WHERE code IS NOT INITIAL.
    DELETE FROM zmk_a190pl_sms WHERE code IS NOT INITIAL.
    DELETE FROM zmk_a190pl_exam WHERE exam_id IS NOT INITIAL.

    " Insert data to DB with ABAP SQL
    MODIFY zmk_a190pl_stud FROM TABLE @students.
    MODIFY zmk_a190pl_crs  FROM TABLE @courses.
    MODIFY zmk_a190pl_sms  FROM TABLE @semesters.
    MODIFY zmk_a190pl_exam FROM TABLE @exams.

    COMMIT WORK.

  ENDMETHOD.
ENDCLASS.
```

3. Feladat – Modularizáció funkciós csoport és funkciós modul használatával

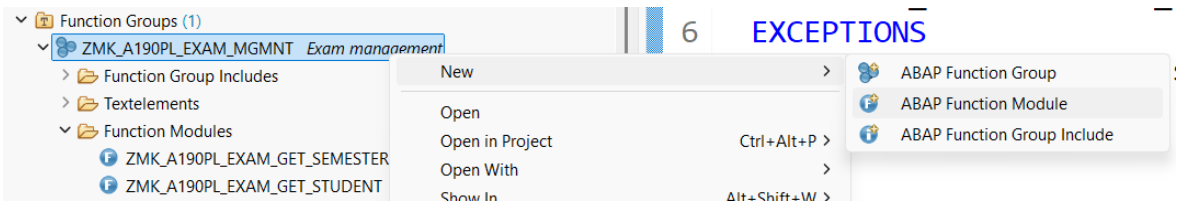
Funkciós csoport létrehozása ADT-ben történt.



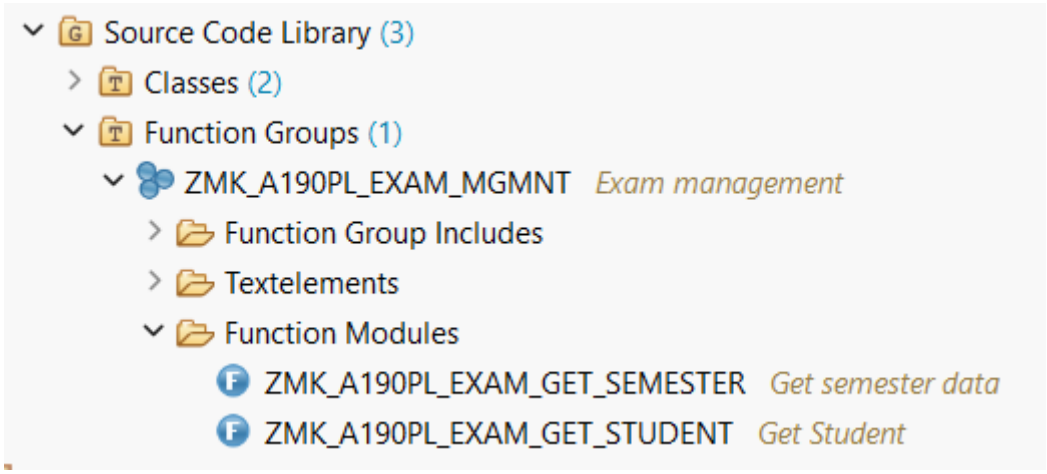
A létrehozott funkciós modulra példa:



Funkciós modulok definiálására legegyszerűbben a funkciós modulon definiált kontext menu ad lehetőséget:



Példa a létrehozott funkciós modulokra:



Forráskód nézetben a funkciós modul interfésze és implementációja megadható.

Példa funkciós modulok.

Hallgató lekérdezése az adatbázisból

```
FUNCTION zmk_a190pl_exam_get_student
  IMPORTING
    neptun_code  TYPE zmk_a190pl_student_neptun
  EXPORTING
    student_data TYPE zmk_a190pl_s_student
  EXCEPTIONS
    student_does_not_exists.

  SELECT SINGLE *
   FROM zmk_a190pl_stud
   WHERE neptun = @neptun_code
   INTO CORRESPONDING FIELDS OF @student_data.
  IF sy-subrc NE 0.
    RAISE student_does_not_exists.
  ENDIF.

ENDFUNCTION.
```

Szemeszter lekérdezése az adatbázisból

```
FUNCTION zmk_a190pl_exam_get_semester
  IMPORTING
    iv_semester_code TYPE zmk_a190pl_semester_code
```

```
  EXPORTING
    es_semester_data TYPE zmk_a190pl_s_semester
  EXCEPTIONS
    semester_does_not_exist.



*---------------------------------------------------------------------*
* Authorization check - Only after exam
*---------------------------------------------------------------------*

*---------------------------------------------------------------------*
* Get semester data from DB
*---------------------------------------------------------------------*
  SELECT SINGLE *
    FROM zmk_a190pl_sms
    INTO CORRESPONDING FIELDS OF es_semester_data
    WHERE code EQ iv_semester_code.
  IF sy-subrc NE 0.
    RAISE semester_does_not_exist.
  ENDIF.

ENDFUNCTION.
```

4. Feladat – Funkciós modul használata

Az ADT konzol alkalmazásként futtatható ABAP osztály létrehozása az első gyakorlaton megtanult módon történt. A példa forráskód:

```
CLASS zmk_cl_uni_2026_gyak3_f2 DEFINITION
  PUBLIC
  FINAL
  CREATE PUBLIC .

  PUBLIC SECTION.

    INTERFACES if_oo_adt_classrun .
  PROTECTED SECTION.
  PRIVATE SECTION.
ENDCLASS.
```

```
CLASS zmk_cl_uni_2026_gyak3_f2 IMPLEMENTATION.


  METHOD if_oo_adt_classrun~main.

    DATA: student_data TYPE zmk_a190pl_s_student.

    CALL FUNCTION 'ZMK_A190PL_EXAM_GET_STUDENT'
      EXPORTING
        neptun_code             = 'ABC123'
      IMPORTING
        student_data            = student_data
      EXCEPTIONS
        student_does_not_exists = 1.
    IF sy-subrc NE 0.
     out->write( 'Hiba történt' ).
    ELSE.
     out->write( student_data ).
    ENDIF.

  ENDMETHOD.
ENDCLASS.
```

A konzolra írt kimenet: