Programozás 3. ZH2 - AAAAAA

<For English version, please scroll DOWN>
Készítsen el egy konzolos alkalmazást, melynek segítségével kézilabda játékosokból tudunk csapatokat összeállítani. A játékosokat véletlenszerűen generáljuk, az adatokat több nyelven érkező XML-ben kapjuk meg, ezt elmentjük a helyi adatbázisba, és ezután statisztikát nyerünk ki az adatbázisból.

SOLUTION NEVE: NEPTUN KÓD, NAGYBETŰKKEL (PL. TIW8GX)

DLL1: NEPTUNKÓD.GENERATOR

a) TeamGenerator osztály: Egy `public static XDocument GenerateTeam(int num, string lang)` metódus segítségével a megadott nyelvű XML-t generál le, benne a megadott darabszámú játékossal

```
<player lang="HU">
  <keresztNev>Blanka</keresztNev>
  <vezetekNev>Háfra</vezetekNev>
  <poszt>Left Back</poszt>
</player>
```

```
<player lang="EN">
  <firstName>Blanka</firstName>
  <familyName>Márton</familyName>
  <position>Left Back</position>
</player>
```

```
<player lang="FR">
  <preNom>Nadine</preNom>
  <nom>Háfra</nom>
  <poste>Centre</poste>
</player>
```

b) A generáláshoz a lenti tömböket kell használni (ctrl+C, ctrl+V –vel kimásolható, nem szükséges begépelni):

```
string[] familyNames = { "Szucsánszki", "Schatzl", "Márton", "Kovacsics", "Háfra", "Klujber", "Bíró" };
string[] firstNames = { "Zita", "Nadine", "Gréta", "Anikó", "Noémi", "Katrin", "Blanka" };
string[] positions = { "RightWing", "LeftWing", "Pivot", "Centre", "Left Back", "Right Back", "Goalie" };
Dictionary<string, string[]> nodes = new Dictionary<string, string[]>()
{
    {  "EN", new string[] { "firstName", "familyName", "position"  } },
    {  "HU", new string[] { "keresztNev", "vezetekNev", "poszt"  } },
    {  "FR", new string[] { "preNom", "nom", "poste"  } }
};
```

DLL2: NEPTUNKÓD.DB

a) LanguageNodeAttribute osztály
   1. Konstruktorral beállítható string Language és string NodeName tulajdonságok
   2. Kizárólag tulajdonságokra ráhelyezhető, ráadásul egy tulajdonságra többször is

b) Player entity osztály
   1. int Id – elsődleges kulcs, adatbázis által szolgáltatott növekményes érték
   2. FamilyName, FirstName, Position – max 100 hosszú, megadása kötelező
      Ez a 3 tulajdonság legyen dekorálva egyenként három darab LanguageNode attribútummal, ezzel jelezzük, hogy melyik nyelven hogyan hívjuk az XML-ben az aktuális tulajdonságot
   3. Salary – egész szám
   4. A kötelező dolgokon felül legyen benne egy az összes adatot egy stringben összefűző ToString()
   5. Legyen benne egy `public static Player CreateFromNode(XElement node)` metódus, ami a paraméter XML node-ból (ami lehet lang="HU", lang="EN", lang="FR" fajtájú is!) létrehoz egy Player példányt, véletlenszerű fizetéssel

c) PlayerContext osztály: A szintén ebben a Class Library projektben definiált adatbázis file-ba code first módszerrel generálja le a táblát.
   1. A táblába egy adatot rakjon bele: 8-as azonosítóval Szucsánszki Zitát, „Centre" poszttal és 10000-es fizetéssel.

CONSOLE APP: NEPTUNKÓD.PROGRAM

a) `static void ProcessXmls(PlayerContext ctx)`
   1. A TeamGenerator segítségével generáljon le külön-külön három XML file-t: egy 95 fős angol, egy 42 fős magyar, és egy 23 fős francia csapatot. Tartsuk meg az XDocument referenciákat, de mentsük el mindhárom file-t.
   2. A három XML –ben lévő játékosokat egyesítse egy közös IEnumerable<XElement> gyűjteménybe, és ezen gyűjteményből csináljon egy IEnumerable<Player> gyűjteményt (mindkettőt ciklus nélkül!!!)
   3. Egy foreach ciklussal az összes játékost mentse el az adatbázisba.

b) `private static void QueryDatabase(PlayerContext ctx).`
   Linq segítségével jelenítse meg az alábbiakat (allekérdezés/több lekérdezés használható, ciklus NEM):
   1. A játékosok darabszámát, az első és az utolsó játékos minden adatát
   2. Posztonként az átlagos és a maximum fizetést
   3. Minden poszthoz a legtöbbet kereső első játékost
   4. Közel azonos fizetéssel rendelkező poszt-párokat: fizetés szerint csökkenő sorrendben az első 10 játékos-párt akarjuk látni, akik azonos poszton 1000-nél kevesebb fizetés-különbséggel rendelkeznek. Az eredmény olyan anonymous példányok gyűjteménye legyen, amiben benne van a két játékos és harmadik tulajdonságként a 2 játékos fizetés-összege
   5. Minden poszthoz az összesítve legdrágább első poszt-párt
   6. A Q2 lekérdezés eredményeit mentse el egy tetszőleges formátumú XML állományba! Ehhez az egy alfeladathoz lehet ciklust használni.

A teljes solution-t (minden könyvtárral/file-al együtt) NEPTUNKÓD.ZIP nevű ZIP file-ként kell feltölteni, a Moodle/FF valamelyikére (ilyen sorrendben haladjunk, és a következőre kapcsolódási hiba esetén ugorjunk). Bejelentéseket a Teams csoporton keresztül fogunk tenni, kérdéseket Teams-en keresztül private chat-ben lehet feltenni! Nem megengedett segédeszköz használata, vagy a használat megkísérlése is csalásnak számít!

Jó munkát kívánunk!

Programming 3. ZH - AAAAAA

Create a console app that will help us in creating teams from handball players. The players are randomly generated, the data is received from multi-language XML files, and then saved to the local database, and we can obtain statistics from the database.

SOLUTION NAME: NEPTUN CODE, ALL CAPITALS (E.G. TIW8GX)

DLL1: NEPTUNCODE.GENERATOR

a) TeamGenerator class: contains a `public static XDocument GenerateTeam(int num, string lang)` method to generate the XML of the given language, returns the specified number of players

```
<player lang="HU">
  <keresztNev>Blanka</keresztNev>
  <vezetekNev>Háfra</vezetekNev>
  <poszt>Left Back</poszt>
</player>
```

```
<player lang="EN">
  <firstName>Blanka</firstName>
  <familyName>Márton</familyName>
  <position>Left Back</position>
</player>
```

```
<player lang="FR">
  <preNom>Nadine</preNom>
  <nom>Háfra</nom>
  <poste>Centre</poste>
</player>
```

b) For the generation, use the arrays below (can be copied with ctrl+C, ctrl+V , no need to type manually):

```
string[] familyNames = { "Szucsánszki", "Schatzl", "Márton", "Kovacsics", "Háfra", "Klujber", "Bíró" };
string[] firstNames = { "Zita", "Nadine", "Gréta", "Anikó", "Noémi", "Katrin", "Blanka" };
string[] positions = { "RightWing", "LeftWing", "Pivot", "Centre", "Left Back", "Right Back", "Goalie" };
Dictionary<string, string[]> nodes = new Dictionary<string, string[]>()
{
    {  "EN", new string[] { "firstName", "familyName", "position"  } },
    {  "HU", new string[] { "keresztNev", "vezetekNev", "poszt"  } },
    {  "FR", new string[] { "preNom", "nom", "poste"  } }
};
```

DLL2: NEPTUNCODE.DB

a) LanguageNodeAttribute class
   1. Contains properties string Language and string NodeName that are set in the constructor
   2. Can be applied ONLY to properties, and multiple times for one property

b) Player entity class
   1. int Id – primary key, database-generated auto increment value
   2. FamilyName, FirstName, Position – max 100 characters, obligatory
      These 3 properties should be decorated with the LanguageNode attribute: three attributes for each property; this signals how the property data is called in the different language XML files
   3. Salary – integer
   4. In addition to the obligatory things, the class should contain a ToString() that returns all data concatenated into one string
   5. It should also contain a `public static Player CreateFromNode(XElement node)` method, that converts the parameter XML node ból (that can be lang="HU", lang="EN", lang="FR" as well!) into a Player instance, with random salary value.

c) PlayerContext class: Using the code first approach, generate the table into the database file located in the same Class Library project.
   1. Do generate one data record into the table: with id 8 you should have Zita Szucsánszki, with „Centre" position and a 10000 salary.

CONSOLE APP: NEPTUNCODE.PROGRAM

a) `static void ProcessXmls(PlayerContext ctx)`
    1. Use the TeamGenerator to generate separately three XML files: an english team with 95 players, a Hungarian team of 42 players, and a French team of 23 players. Keep the XDocument instances, but also save the files!
    2. Unite the players from the three XML into a common IEnumerable<XElement> collection, and convert this collection into an IEnumerable<Player> collection (both without loops!!!)
    3. Use a foreach loop to save all players into the database.

b) `private static void QueryDatabase(PlayerContext ctx).`
    Using Linq, determine the followings (You can use subqueries/multiple queries, but NOT loops):
    1. The number of players, all data of the first and last players
    2. The maximum and the average salary for every position
    3. For every position, the players who earn the most
    4. The post-pairs with close salaries: ordered by the descening order of salaries, we want to see the top 10 player-pair, who play in the same position, and the difference between their salaries is smaller than 1000. The result should be a collection of anonymous instances that contain both players and the third property is the addition of the two salaries
    5. For every position, the most expensive post-pair
    6. Save all the results of the Q2 query into an XML of a custom format! Only in this sub-task, you are allowed to use loops!

Compress your full solution (all folders/files included) into a NEPTUNCODE.ZIP compressed file, and upload to Moodle/FF (process in this order, jump to the next item in case of connection troubles). We'll make announcements in the Teams group, and you can ask questions via private chat in Teams! Using not-authorized help or even trying to use not-authorized help is considered cheating!

We wish you a successful test!