Az Eclipse Modellezési Keretrendszer bevezetése

Szerzők: Semeráth Oszkár, Nagy András Szabolcs, Graics Bence, Bergmann Gábor

Az EMF-ről

A Wikipedia alapján (http://en.wikipedia.org/wiki/Eclipse_Modeling_Framework): ''Az Eclipse Modeling Framework (EMF) egy Eclipse-alapú modellezési keretrendszer és kódgeneráló eszköz olyan eszközök és alkalmazások építéséhez, amelyek strukturált adatmodellre épülnek.'' Az EMF adatmodellje könnyűsúlyú, mivel csak néhány jól definiált modellezési elemet definiál. Azonban széles körű eszköztámogatással és közösséggel rendelkezik. Például definiálhatja egy nyelv szöveges vagy grafikus szintaxisát és generálhatja a megfelelő szerkesztőket.

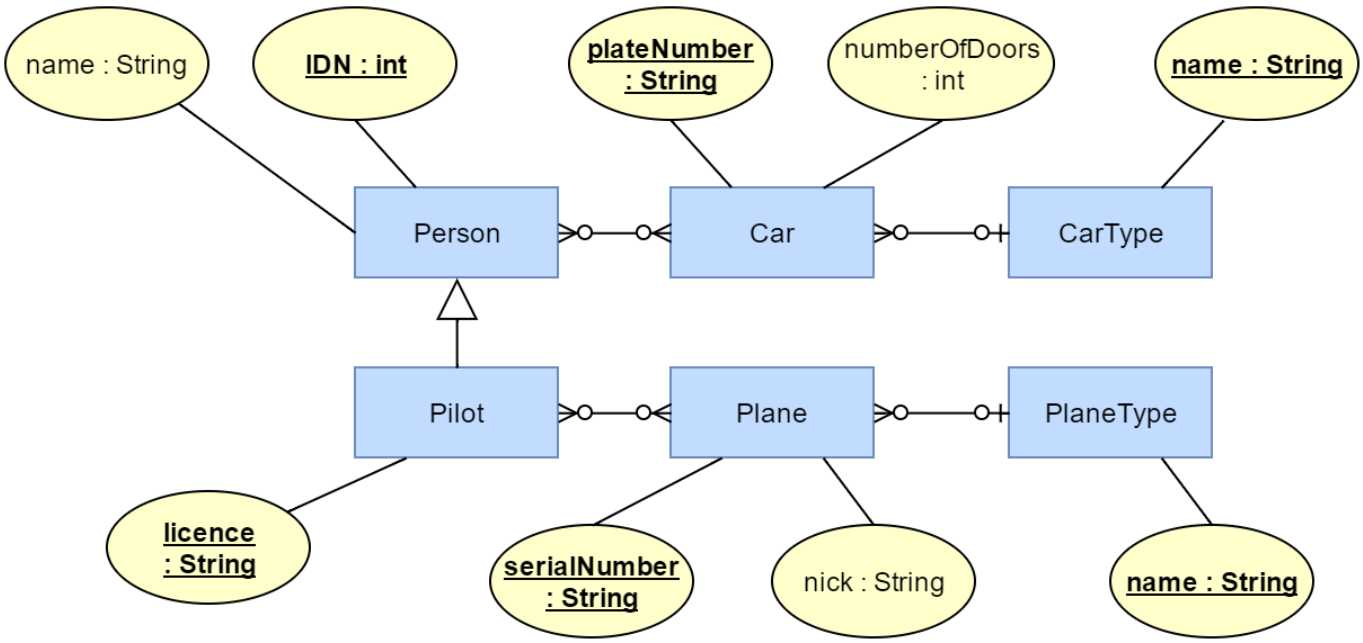
Az EMF képes Java kódot generálni a metamodellből egyetlen gombnyomással. A generált kód egy Java objektum API-t biztosít, amely összhangban van a metamodelllel. Az EMF alapból támogatja az XMI formátumba való szerializálást és XMI fájlokból való deszerializálást.

EMF honlapja: https://projects.eclipse.org/projects/modeling.emf.emf/

A feladat leírása

Ennek a gyakorlatnak a célja, hogy létrehozzon egy testre szabott Egyed-Kapcsolat Diagram (Entity-Relationship Diagram, ERD) metamodellt. Ezek a diagramok segíthetik a szoftverkomponensek fejlesztését, amelyek összetett adatszerkezetekkel dolgoznak.

Az alábbi ábra egy példa egy Egyed-Kapcsolat diagramra.



Előfeltételek

Az Eclipse Modeling Tools kiadás minden szükséges plug-int tartalmaz.

Ecore modell: lépésről lépésre

1. Hozzon létre egy új **Üres EMF projektet** (**Empty EMF Project**) a **File | New | Other... | Eclipse Modeling Framework | Empty EMF Project** lehetőséggel. Nevezze el `hu.bme.mit.mdsd.erdiagram`-nak. Ez egy Java plugin projekt (src mappával és egy MANIFEST.MF fájllal), amely az szükséges ecore függőségeket tartalmazza.

2. A projektben található egy mappa, nevezetesen **model**. Hozzon létre egy új **Ecore Model**-t benne, azzal hogy jobb-klikkel a mappán, majd kiválasztja az | New | Other... | Eclipse Modeling Framework | Ecore Model lehetőséget. Nevezze el `erdiagram.ecore`-nak.

3. Egy új szerkesztő nyílik meg, amely megjeleníti, hogy a modell forrása egy még névtelen üres package. Töltse ki a hiányzó tulajdonságokat az Properties nézetben, kisbetűket használva:

   - Name: `erdiagram`
   - Ns Prefix: `erdiagram` (névtér előtag az XML-ben)
   - Ns URI: `hu.bme.mit.mdsd.erdiagram` (globális azonosítóként működik. Egy másik konvenció az, hogy `http://example.com/erdiagram`-ot használunk)

   Ha az Properties nézet nem látható, akkor jobb-klikkeljen a package elemen az szerkesztőben, és válassza ki a **Show Properties View** lehetőséget. Általában a nem elérhető nézetek megjelenítéséhez használja az **Window | Show View** lehetőséget, vagy használja a Quick Access dobozt.

4. Használja ezt a fa szerkesztőt az első EClass létrehozásához, amely az ERD gyökér modellelemet fogja reprezentálni (jobb-klikk a package-en | New Child | EClass), és nevezze el "EntityRelationDiagram"-nek az Properties nézetben. Ne felejtse el menteni a modellt. Próbálja ki a többi tartalmi menü elemet is.

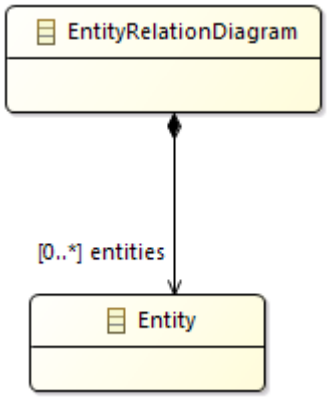
5. Bár lehetséges a modell létrehozása ebben a fa szerkesztőben, van egy kényelmesebb szerkesztő erre a célra. A diagramok hozzáadásához az projektre jobb-kattintva válassza a Configure | Enable ViewModelNature opciót.

6. Kattintson jobb gombbal az ecore fájlra, majd válassza az **Initialize ECore diagram...** lehetőséget. Nevezze el **erdiagram.aird**-nek. Kattintson a Finish gombra. Ezután válassza ki a Design/Entities (vagy Entities in a Class Diagram) reprezentációt, majd az `erdiagram` package-et. Nevezze el az ábrát Entity Relation class diagram-nek.

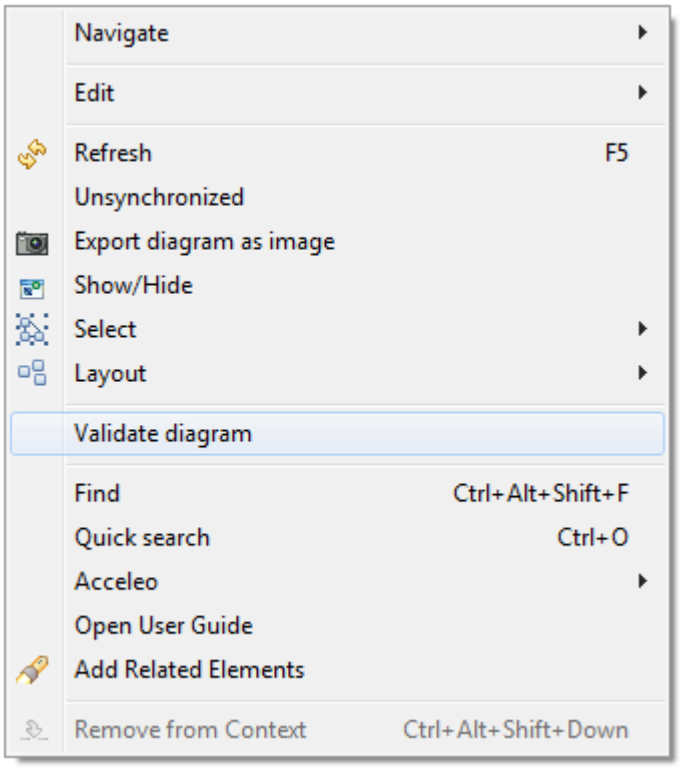
7. Ha az **Empty EMF Project** opciót választja a **File | New | Other... | Eclipse Modeling Framework | Empty EMF Project** menüpontban, akkor az IDE létrehoz egy projekt struktúrát, amely hasonló az aktuális állapothoz.

8. Adja hozzá az első elemet a palettáról az Existing Elements | Add parancs segítségével, majd válassza ki a már létrehozott EClass-ot. Alternatív megoldásként duplán kattintson a vásznon, amikor az ábra először inicializálódik. Vegye figyelembe, hogy a vizuális reprezentáció független az ecore modelltől, azaz az elem törlése a modellből és az ábráról különböző dolgok (az utóbbi nem feltétlenül eredményez elem törlést a modellből).

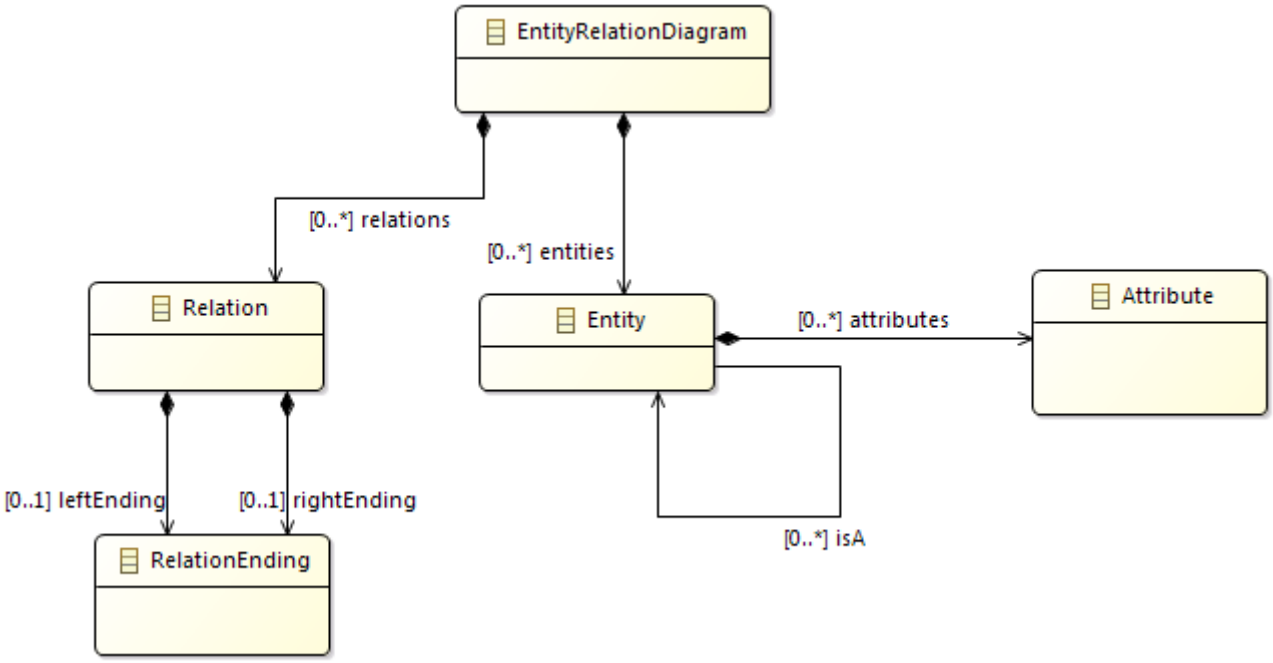
9. Húzza be a metamodell elemeket a palettáról, hogy elkészítse a következő részletet:



10. Ha rákattint egy modellelemre, szerkesztheti annak tulajdonságait a **Property nézet**ben.
    - Adja meg az `EClass`-ok, `EReference`-ek és `EAttribute`-ok neveit.
    - Az `EClass`-okat ebben a nézetben állíthatja be **Abstract** vagy **Interface** típusra.
    - Az `EAttribute` típusa ebben a nézetben állítható be.
    - A kapcsolat multiplicitása `0..*`-ra van állítva.
    - Az `EOpposite` tulajdonság meg kell legyen jelenítve.
    - Az instance modellek objektumai a metamodellben egy *fa hierarchia* szerint kell elhelyezkedjenek a tartalmazási referenciák szerint. Állítsa az `entities` kapcsolatot **Is Containment**-re.
    - **Megjelenés:** szerkesztheti a diagram megjelenését. Például az élek viselkedését.
    - **Speciális beállítások:** Az elemek tulajdonságainak közvetlen szerkesztése.
11. A diagram szerkesztésének hatása az `.ecore` fájlra megfigyelhető, ha megnyitják. Szinkronizálódik, amikor a diagramot mentik.
12. A modell ellenőrizhető a Validate diagram-re kattintva a kontextus menüben (lásd az alábbi ábrát). Az összes generált jelzőt listázza a Problems nézetben.



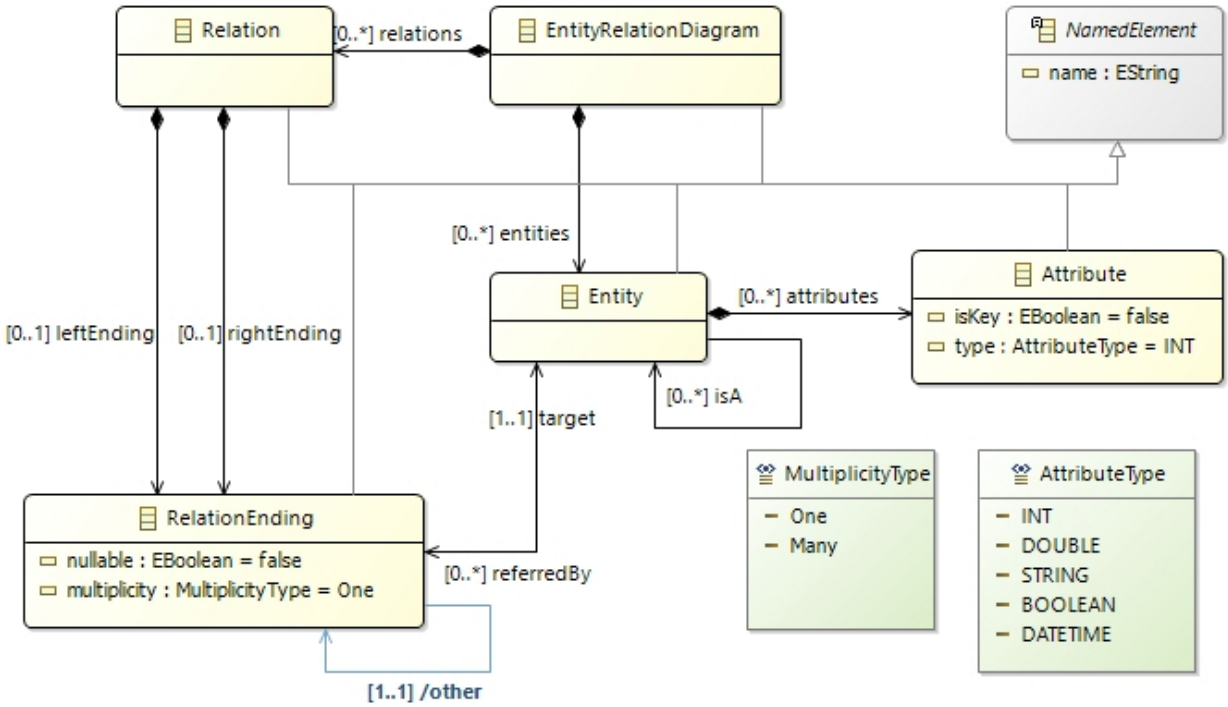
13. Hozza létre az Egyed-Kapcsolat Diagram metamodelljét úgy, mintha osztálydiagram lenne. Egy lehetséges eredmény látható az alábbi ábrán.



14. Adjon hozzá egy `EEnum`-ot a metamodellhez `Multiplicity` néven, és adjon hozzá két literált: `One` és `Many`.

15. Adjon hozzá egy absztrakt `NamedElement` osztályt a metamodellhez, és hozzon létre öröklési kapcsolatokat az összes `EClass`-hoz. Megjegyzés: az `EAttribute` és az `EReference` közti különbség az, hogy az `EAttribute` az `EDataType` típusra vonatkozik, míg az `EReference` az `EClass` típusokat használja.

16. Adjon hozzá `EAttribute`-okat az osztályokhoz, és hozzon létre egy `AttributeType` enumerációt. Az elkészült modell hasonló kell legyen a következő ábrán láthatóhoz:



17. A metamodell nem tartalmaz `EOperations`-öket, mivel ebben a feladatban célunk egy adatmodell definiálása. Fontos megjegyezni, hogy az EMF készítői sem ajánlják az `EOperations` használatát.

Szerkesztő: lépésről lépésre

Ez az példa bemutatja, hogyan lehet osztályokat és egy szerkesztőt generálni az Ecore modellekből.

1. Az ecore fájlok a szakterületi nyelvek tervrajzai. Ahhoz, hogy az Eclipse által nyújtott eszköztámogatást használni lehessen, Java osztály reprezentációkra van szükség ezekből a "dobozokból". Szerencsére ezek az osztályok automatikusan generálhatók.

   Jobb-kattintson *az ecore fájlra* és válassza ki a **New | Other | Eclipse Modeling Framework | EMF Generator Model** lehetőséget. Az alapértelmezett `erdiagram.genmodel` megfelelő. A következő lépésben válassza ki, hogy a generátor az Ecore modellből generáljon. A harmadik lépésben meg kell adni az Ecore modell URI-ját. Kattintson a Load és Next gombra. Válassza ki az egyetlen elérhető package-et a generáláshoz, majd kattintson a Finish gombra.

2. Ennek eredményeként megnyílik egy másik fa szerkesztő (hasonlóan az ecore szerkesztőhöz). Böngéssze át a beállításokat a Properties szerkesztőben. Jobb-kattintson a gyökéren, és válassza ki a **Generate Model** parancsot. Három package generálódik a forrásmappában. Az első package tartalmazza az általunk létrehozott osztályokat Java interfészként. Emellett két speciális interfész is van: *Factory.java és *Package.java, ezekre később visszatérünk. Nézze meg például a `hu.bme.mit.mdsd.erdiagram/src/erdiagram/EntityRelationDiagram.java` fájlt, és láthatja, hogy semmi furcsa nincs generálva, ez egy interfész alapvető metódusokkal. A második package (erdiagram.impl) az implementációs osztályokat tartalmazza. Az implementációs osztályban néhány szokatlan mező található, de az interfész függvények implementációi egyszerűek. Az erdiagram.util package két segédosztályt tartalmaz, amelyek hasznosak lehetnek bonyolultabb esetekben.

3. A generátor modell fő funkciója az előállított kód irányítása. Például észrevehetjük, hogy az generált package nevek sajnos nem egyeznek meg a projekt nevével. Ennek korrigálásához kattintson a Package elemre, majd keresse meg az *All* | *Base Package* opciót. Változtassa meg a `hu.bme.mit.mdsd` értékre. Ez korrigálja az előállított package neveket a projekt nevével összhangban. Generálja újra a modellkódot, törölje az előzőleg generáolt kódot, és korrigálja a MANIFEST fájlt, mivel az a törölt package-eket próbálja exportálni. Továbbá megváltoztathatja a `src` mappa nevét (például `emf-gen`-re). Ehhez kattintson a .genmodel szerkesztőben a gyökér objektumra, és találja meg a *Model* | *Model Directory* opciót, majd módosítsa a `src` mappa nevét `emf-gen`-re. Azonban ez eltöri a runtime Eclipse-ben a generált fa szerkesztőt, tehát ne csinálja ezt most. További lehetőségek a generátor modellben például osztályonként egy címke tulajdonság kiválasztása (például "name").

4. Ha megváltoztatja az ecore modellt, újra kell töltenie a generátor modellt a **Jobb-kattintás | Reload** opcióval... A generátor modell fájlban végrehajtott változtatások megmaradnak. Vegye figyelembe, hogy bizonyos esetekben ez az újratöltési eljárás nem írja felül a kódgenerálási beállításokat bizonyos modell elemekre (mivel az ecore kézi testreszabását támogatja). Például, ha az ecore modellben az egyik `EReference` `Containment` attribútumát átállítja hamisról igazra, vagy fordítva, a generátor modellben a megfelelő beállítás nem változik automatikusan még az újratöltés után sem. Próbálja ki ezt a *leftEnding* `EReference` Containment attribútumának hamisra állításával. Nézze meg a generátor modellben a *leftEnding* `EReference` tulajdonságait (**Erdiagram | Relation | leftEnding** a generátor modell fa szerkesztőjében). Nyissa meg az Edit nézetet, és vegye észre, hogy a `Children` és `Create Child` attribútumok továbbra is igazra vannak állítva, ami váratlan problémákat okozhat, például a

generált szerkesztő használatakor. Ha azt szeretné, hogy a generátor modell összhangban legyen az ecore modellel, akkor az ilyen változtatások után kézzel szerkesztheti ezeket a tulajdonságokat, vagy törölheti és újra létrehozhat egy generátor modellt az ecore modell alapján.

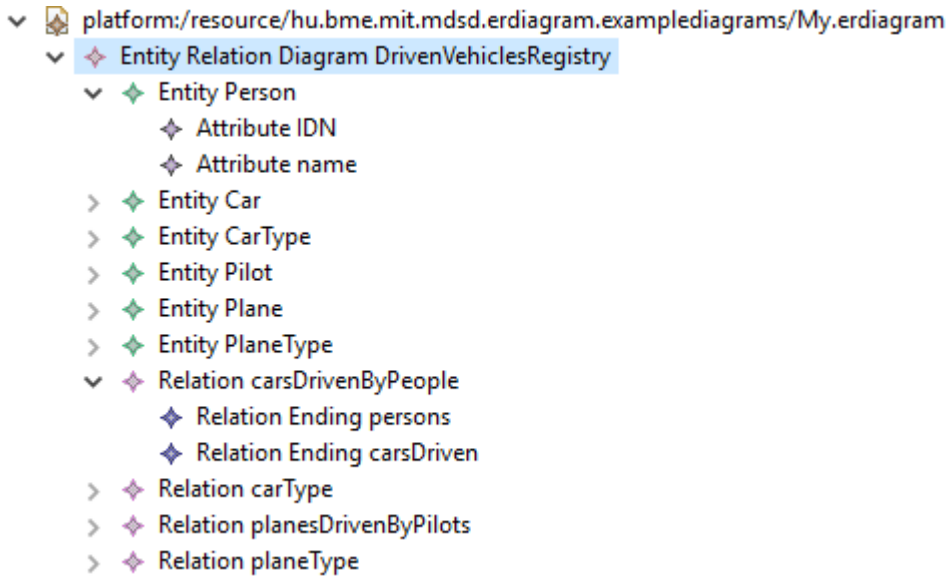
5. Generáljon egy **szerkesztőt**. Jobb-kattintson a genmodel fájl gyökérére, és először generálja az edit és editor funkciókat.

6. Jobb-kattintson a projekten, és válassza az **Run as | Eclipse application** opciót. Vagy válassza az **Run | Run configurations...| Eclipse application** lehetőséget az eszköztárban, majd kattintson a *New launch configuration* gombra, opcionálisan töltse ki a paramétereket (például az launch konfiguráció nevét), és nyomja meg a "Run" gombot. Mindkét lehetőség alapértelmezés szerint egy "Eclipse alkalmazás" launch konfigurációt futtat, amely tartalamzza az összes plug-in-ek a munkaterület forrásprojektekben (az előbb generált plug-in-eket), valamint a cél platformon (ez utóbbi alapértelmezett beállítás szerint a plug-inek fejlesztéhez használt Eclipse példányban használt plug-inek halmaza).

7. Hozzon létre egy üres projektet a **File | New | Other... | General | Project** opcióval, és nevezze el **hu.bme.mit.mdsd.erdiagram.examplediagrams**-nek.

8. Hozzon létre egy új Egyed-Kapcsolat Diagramot az új projektben azzal, hogy jobb-kattint a projekten, majd kiválasztja az **New | Other | Example EMF Model Creation Wizard | Erdiagram Model** opciót. A név lehet az alapértelmezett `My.erdiagram`, és a modell objektum (amit szeretnénk szerkeszteni) legyen az **Entity Relation Diagram**.

9. Hozza létre az példánymodellt. A szerkesztő használata meglehetősen egyértelmű és nagyon hasonlít az ecore fa szerkesztőjéhez.



Modell módosítása: lépésről lépésre

Az alábbi példa bemutatja, hogyan lehet a modellt kódból szerkeszteni.

1. Hozzon létre egy új **Plug-in Project**-et a jobb kattintás | New | Plug-in Project opcióval. Nevezze el `hu.bme.mit.mdsd.erdiagram.example`-nek, majd adja hozzá az alábbi függőségeket:

```
```

```
hu.bme.mit.mdsd.erdiagram    The plug-in project's name that holds the ecore
model.
org.eclipse.emf.ecore.xmi    The instance model is serialized as an XMI
document.
```

2. Hozzon létre egy osztályt a forrásmappába:

```
package hu.bme.mit.mdsd.erdiagram.example
name    ErdiagramModels
```

3. Hozzon létre egy modellt a generált Java kóddal. A példánymodell objektumait a generált *Factory singletonon keresztül kell példányosítani és csak az interfészükön keresztül szerkeszthetők, kerülendő a *Impl osztályok használata. Az alábbi módszer egyetlen entitást hoz létre Person névvel és egy Nickname attribútummal:

```
public EntityRelationDiagram createModel() {

    ErdiagramFactory factory = ErdiagramFactory.eINSTANCE;
    EntityRelationDiagram diagram = factory.createEntityRelationDiagram();

    Entity person = factory.createEntity();
    person.setName("Person");

    Attribute name = factory.createAttribute();
    name.setName("Nickname");
    name.setType(AttributeType.STRING);

    person.getAttributes().add(name);
    diagram.getEntities().add(person);

    return diagram;
}
```

4. Hozzon létre egy metódust, amely bejárja a modellt, és kiírja az entitások nevét.

   Az eredmény kiírható az alábbi módon:

```
public void printErdiagram(EntityRelationDiagram erdiagram)
{
    for (Entity entity : erdiagram.getEntities()) {
        System.out.println(entity.getName());
    }
}
```

5. Hozzon létre egy main metódust, amely futtatja a korábbi kódot. Próbálja ki azzal, hogy az osztályra jobb gombbal kattint, majd kiválassza a **Run as | Java Application** lehetőséget.

```
public static void main(String[] args) {
    ErdiagramModels erdiagramModels = new ErdiagramModels();
    EntityRelationDiagram model = erdiagramModels.createModel();
    erdiagramModels.printErdiagram(model);
}
```

6. Valószínűleg nem fog működni. Az Ecore modellek inicializációt igényelnek, amit az alábbi kóddal lehet elvégezni. Megjegyzés: ez nem szükséges, ha az alkalmazást Eclipse alkalmazásként vagy JUnit Plug-in tesztként futtatja.

```
public void init() {
   // For the initialisation of the model.
   // Without this the following error happens:
   //   "Package with uri 'hu.bme.mit.mdsd.erdiagram' not found."
   ErdiagramPackage.eINSTANCE.eClass();
}
```

7. A modell egy xmi fájlba elmenthető és betölthető, amelyet egy `Resource` objektum kezel. Egy resource referenciája egy `URI` objektummal történik (fontos megjegyezni, hogy számos `URI` nevű osztály van; az általunk szükséges az `org.eclipse.emf.common.util` package-ben található). Egy `ResourceSet` objektum `Resource` objektumok egy halmazát kezeli. Írjon egy metódust, amely létrehoz egy `Resource` objektumot, és egy másikat, amely betölt egy `URI` objektummal meghatározott resource-t:

```
public Resource createResource(URI uri) {
    ResourceSet resSet = new ResourceSetImpl();
    Resource resource = resSet.createResource(uri);
    return resource;
}
```

```
public Resource loadResource(URI uri) {
    ResourceSet resSet = new ResourceSetImpl();
    Resource resource = resSet.getResource(uri, true);
    return resource;
}
```

8. Az resource sorosítható a `save()` metódussal:

```
public void saveResource(Resource resource) {
   try {
     resource.save(Collections.EMPTY_MAP);
```

```
    } catch (IOException e) {
       System.out.println("The following error occurred during saving the
resource: "
           + e.getMessage());
    }
}
```

9. A modell elérhető egy `Resource` objektumból a `getContents()` metódus segítségével:

```
public EntityRelationDiagram getModelFromResource(Resource resource) {
    // check the content in production code!
    EntityRelationDiagram root = (EntityRelationDiagram)
resource.getContents().get(0);
    return root;
}
```

10. Tegye ezeket össze egy Java main metódusba. Figyelje meg, hogy az URI-t megkaphatja a jobb kattintás | **Properties** menüponttal.

```
public static void main(String[] args) {
    // init
    ErdiagramModels erdiagramModels = new ErdiagramModels();
    erdiagramModels.init();
    // create
    EntityRelationDiagram model = erdiagramModels.createModel();
    // save
    URI uri =
URI.createFileURI("C:/Users/meres/Desktop/samplemodel.erdiagram");
    Resource resource = erdiagramModels.createResource(uri);
    resource.getContents().add(model);
    erdiagramModels.saveResource(resource);
    // load to a different resource
    Resource resource2 = erdiagramModels.loadResource(uri);
    EntityRelationDiagram model2 =
erdiagramModels.getModelFromResource(resource2);
    // print
    erdiagramModels.printErdiagram(model2);
}
```

Jobb klikk az osztályon, majd válassa a **Run as | Java Application** lehetőséget. Ez egy egyszerű Java alkalmazásként fogja futtatni a kódot, amely létrehoz, ment, betölt és kiír egy modellt.

11. Némi inicializáló kód még hiányzik, amely regisztrálja az XMI szerializálót a megadott fájlkiterjesztéshez.

```
public void init() {
    // ...
```

```
    // Defining that the files with the .erdiagram extension should be parsed
as an xmi.
    Resource.Factory.Registry reg = Resource.Factory.Registry.INSTANCE;
    reg.getExtensionToFactoryMap().put("erdiagram", new
XMIResourceFactoryImpl());
}
```

12. Az alkalmazás újrafuttatása után megtalálhatja az újonnan létrehozott .erdiagram fájlt a megadott elérési útvonalon (asztalon, vagy ha a projekt mappát adta meg, akkor lehet, hogy frissíteni kell az F5 gombbal vagy a frissítés gombbal). Ez nem nyitható meg a generált fa szerkesztővel, mivel ez a szerkesztő nincs telepítve az Eclipse-be. Az megnyitásához használhatja a Sample Reflective Ecore Editor-t, indíthat egy runtime Eclipse-et, vagy exportálással telepítheti a plugineket a host-ra.

UI integráció: gyors példa

Fent láthattunk egy egyszerű Java konzolalkalmazást, amely betöltött, manipulált és mentett EMF modelleket. Azonban az EMF objektumok leggyakoribb használata az Eclipse Workbench-ben történik, az Eclipse szerkesztőkbe betöltött EMF modellekkel való interakcióval. Bár az Eclipse plug-in fejlesztés külön kurzust érdemelne, itt gyors és általános bejárást végzünk egy olyan helyzetben, amikor egy tartalommenü műveletet adunk hozzá meglévő EMF szerkesztőkhöz, amely hozzáfér és módosítja az aktuálisan szerkesztett modellt.

A példa plug-in egy tartalommenü műveletet fog regisztrálni, amelyet bármely kiválasztott Entity modellelemre lehet aktiválni, és végrehajtásakor hozzáad egy "FooBar" nevű attribútumot a kiválasztott entitáshoz.

1. Létre fogunk hozni egy Eclipse plug-in projektet. Válassza ki a **File | New | Project... | Plug-in project** lehetőséget, adjon nevet a projektednek, például `hu.bme.mit.mdsd.erdiagram.action`, és az utolsó oldalon a varázslóban kapcsolja ki a "Create a plug-in using one of the templates" opciót. Nyissa meg a `META-INF/MANIFEST.MF` fájlt a létrejött projektben, és adja hozzá plug-in függőségként mind a `hu.bme.mit.mdsd.erdiagram` modell projektet (amely tartalmazza a metamodellt és a generált Java osztályokat) és az `org.eclipse.ui`-t (amelyet ki fogunk bővíteni). Ezeket a függőségeket könnyedén hozzáadhatja a manifest szerkesztő "Dependencies" fülén (konkrétan az "Required Plug-ins" panelen) vagy kézzel a MANIFEST.MF fülön a szöveges szintaxis használatával.

2. Fontos megérteni, hogy három féle bővítményt kell hozzáadnunk az Eclipse felhasználói felületéhez. Először is definiálni fogjuk az "Add FooBar" *Eclipse parancsot*. Másodszor, ilyen parancs megjelenhet az UI-n több helyen is; hozzá lehet rendelni billentyűparancshoz, megjelenhet a fő menüben, eszköztárakon stb.; mi most konkrétan egy felugró menühöz fogunk hozzájárulni, hogy (feltételesen) megjelenjen a kontextusmenükben. Végül, több kódrészlet, úgynevezett *kezelők* (*handlers*) is lehetnek, amelyek valójában végrehajtják a parancs feladatait, amelyek közül mindig egy kiválasztódik a kontextustól függően; itt egyetlen kezelőt fogunk regisztrálni, egyetlen módja lesz a parancs végrehajtásának minden körülmények között.

3. Ez elérhető három bővítési pont (extension point) hozzáadásával, amelyeket a `org.eclipse.ui` plug-in nyújt, nevezetesen az `org.eclipse.ui.menus`, az `org.eclipse.ui.commands` és az `org.eclipse.ui.handlers`. A manifest szerkesztőjében az "Extensions" fülön hozzáadhatja ezeket a bővítményeket; most adjon hozzá egy bővítményt például az `org.eclipse.ui.handlers`-hez (a részletek kitöltése nélkül vagy gyerek elemek létrehozása nélkül), majd mentse el a szerkesztőt.

4. Megjelenik a `plugin.xml` fájl, amely az általunk hozzáadott bővítményeket írja le. Az "Extensions" fül helyett közvetlenül szerkeszthető XML szintaxisban a manifest szerkesztő "plugin.xml" fülén keresztül. Tegye pontosan ezt, és illesze be az alábbi tartalmat. Fontos megjegyezni, hogy az `hu.bme.mit.mdsd.erdiagram.Entity` Java osztályt fully qualified nevével hivatkozzuk (mint egy egyszerű string-re); győződjön meg arról, hogy a név megegyezik a workspace-ben használttal.

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>
    <extension
     point="org.eclipse.ui.commands">
      <command
        description="Add an attribute called 'FooBar' to selected Entity"
        id="hu.bme.mit.mdsd.erdiagram.action.AddFooBar"
        name="Add FooBar Attribute">
      </command>
    </extension>
    <extension
     point="org.eclipse.ui.handlers">
      <handler
          class="hu.bme.mit.mdsd.erdiagram.action.AddFooBarHandler"
          commandId="hu.bme.mit.mdsd.erdiagram.action.AddFooBar">
      </handler>
    </extension>
    <extension point="org.eclipse.ui.menus">
      <menuContribution locationURI="popup:org.eclipse.ui.popup.any?
after=additions">
        <command commandId="hu.bme.mit.mdsd.erdiagram.action.AddFooBar"
            style="push" tooltip="Add FooBar Attribute">
            <visibleWhen checkEnabled="false">
              <with
                    variable="activeMenuSelection">
                 <iterate>
                    <adapt
                          type="hu.bme.mit.mdsd.erdiagram.Entity">
                    </adapt>
                 </iterate>
              </with>
            </visibleWhen>
        </command>
      </menuContribution>
    </extension>
</plugin>
```

5. Ahogy az fent látható, a bővítmény contribution-nek van egy láthatósági feltétele, amely biztosítja, hogy a menüelem csak az entitások kontextusmenüjében jelenjen meg. Pontosabban ez működni fog bármely nézetben vagy szerkesztőben (például az alapértelmezett generált EMF fa szerkesztőben vagy az Outline nézetben), ahol a kiválasztás azonosítható, hogy megfelel egy Entity objektumnak. Fontos tudni, hogy bizonyos konkrét szintaxis keretrendszerekben a kiválasztás tartalmazhat olyan jelölési elemet, amely különbözik az alapul szolgáló domain modell EObject-tól; ezekben az esetekben további

becsomagolási/kicsomagolási lépésre van szükség, amely specifikus a megadott konkrét szintaxis szerkesztő technológiához.

6. További testreszabási lehetőségek is vannak (például biztosítani, hogy a parancs le legyen tiltva, ha más elem van kiválasztva; biztosítani, hogy a menüelem megfelelő helyen jelenjen meg a kontextusmenüben, stb.), de most továbblépünk. A plugin.xml mentése után vegye észre, hogy egy figyelmeztetés található a felső részen, a handler contribution `class=` sorában. Itt hivatkozzuk a kezelő logikát megvalósító osztályt, de még nem hoztuk létre azt az osztályt. Ha a kurzort arra a sorra mozgatjuk, és megnyomjuk a Ctrl+1 billentyűkombinációt, megjelenik egy választék a *gyors javítások* (*quick fixes*) közül, amelyek közül a "Create..." lehetőségre lesz szükségünk. A kiválasztásával megjelenik egy "New Java Class" varázsló. Az `IHandler` interfész neve előre ki van töltve, de könnyebb dolgunk lesz, ha hozzáadjuk az `org.eclipse.core.commands.AbstractHandler` ősosztályt is (a "Superclass" mezőben). Így csak egyetlen metódust kell megvalósítani. Nyomjunk a Finish gombra.

7. Vegye figyelembe, hogy az Eclipse környezetben az bővítmény pont (extension-point) mechanizmusát is használják az Ecore package-ek globális regisztrációjához, a fájlkiterjesztések az EMF resource típusokkal való társításához, stb. Ezért nincs szükség az előzőleg leírt lépésekre az Ecore package-ek inicializálásához és az XMIResourceImpl regisztrálásához; közvetlenül használhatjuk az EMF típusokat. Először azonban ki kell csomagolnunk az EMF objektumot az Eclipse UI által biztosított *kiválasztás* mechanizmusának adatszerkezetéből. Helyezze be tehát az alábbi implementációt az `AddFooBarHandler.execute` metódusba:

```
@Override
public Object execute(ExecutionEvent event) throws ExecutionException {
    // unpack selected object
    IStructuredSelection selection = (IStructuredSelection)
HandlerUtil.getCurrentSelection(event);
    Object firstElement = selection.getFirstElement();

    // cast to Entity, which it must be in this case
    Entity entity = (Entity) firstElement;

    // perform business logic
    Attribute newAttribute = ErdiagramFactory.eINSTANCE.createAttribute();
    newAttribute.setName("FooBar");
    entity.getAttributes().add(newAttribute);

    return null;
}
```

8. Ismételten futtasson egy "Eclipse alkalmazás" launch konfigurációt, amely az összes workspace-beli forrásprojektben és a célplatformon meghatározott plugint tartalmazza; egyszerűen használhatja a korábbi "Eclipse Application" indítási konfigurációt az **Run** eszköztári gomb lenyíló menüjéből. A runtime Eclipse workspace-ében hozzon létre egy új ER diagram példányt az varázsló segítségével, vagy nyisson meg egy korábban létrehozott példánymodell fájlt. Figyelje meg, hogy az "Add FooBar Attribute" megjelenik az Entity objektum kontextusmenüjében a fa szerkesztőben vagy az áttekintési nézetben kiválasztott Entitás esetén, és hogy ez az akció valóban hozzáad egy új attribútumot a modellhez.

9. (Haladó téma) Vegye figyelembe, hogy bár a megvalósított művelet helyesen módosítja az EMF modellt, a változás lehet, hogy nem jelölte meg a szerkesztőt mint *piszkos* (azaz készen áll egy *mentés* műveletre), és nem is jelenik meg az undo/redo veremben sem. Tehát bár az alacsony szintű EMF modell API tökéletes az olyan modellátalakításokhoz, amelyek modellfájlokat várnak be- és kimenetként, és saját maguk kezelik a modell resource-ait, nem működik jól együtt a szerkesztővel betöltött és kezelt modellekkel. Lehetséges megoldás lenne, ha az *EMF Command API*-t használnánk a változtatások megfogalmazására, amely kezeli az undo/redo és a piszkos állapot kezelését, és az editor *editing domain*-jén keresztül kommunikálná ezeket a változtatásokat. Sajnos, ahogyan azt a gyakorlaton is láthatta, az alapvető Command API kézi használata meglehetősen nehézkes, különösen a nem tranzakciós editing domainek esetében.

Általános tippek

- Ha bármi hiba történik az újragenerálás során, és probléma merül fel a kóddal kapcsolatban, két lehetősége van:
  - Ha a dokumentumot nem szerkesztették kézzel, vagy nem értékes, törölje azt. Generálja újra a kódot, és valószínűleg rendben lesz. Ez működik a `Manifest.MF` és a `plugin.xml` esetében is.
  - Más esetekben ne féljen a felülírástól. Például, ha töröl egy elemet az metamodellből, az XMI fájl, ami tartalmazza az példánymodell, hátramaradt címkéket tartalmazhat, amiknek nincs meghatározott típusa. Ez az XMI-t érvénytelenné teszi, de nem szükséges az egész példánymodellt újra kezdeni; egyszerűen törölje az nemkívánatos részeket a kódból kézzel.

A projektek végső állapotát megtalálhatja ebben a repository-ban, az `EMF` branch-en.

Haladó téma: forrásközi hivatkozások

1. A forrásközi hivatkozások bemutatásához hozzon létre egy új Erdiagram modell fájlt `other.erdiagram` néven a fenti varázsló segítségével. Töltse fel legalább egy "Foreign Entity" nevű entitással, majd mentse el a fájlt, és zárja be az szerkesztőt.

2. Nyissa meg a korábban létrehozott `My.erdiagram` fájlt a fa szerkesztőben, kattintson jobb gombbal az üres területre, és válassza ki a "Load resource..." lehetőséget. Válassza ki a "Browse Workspace..." opciót, és válassza ki a `other.erdiagram` fájlt. Figyelje meg, hogy az EMF Resource Set most kibővült egy további resource-szal, amit az resource URI-ja azonosít.

3. Hozzon létre egy kereszthivatkozást az újonnan betöltött resource-hoz egy "Relation" elem kiválasztásával az régi modellben, és megadva "Foreign Entity"-t a "jobb" végpontként.

4. Mentse el a modellt, zárja be a szerkesztőt, majd nyisas meg újra a `My.erdiagram` fájlt. Figyelje meg, hogy az második resource kezdetben nem jelenik meg az resource set részeként. A fa szerkesztőben való böngészés után végül *lustán betöltődik* szükség szerint, például amikor a másik resource-ra mutató Relation (vagy annak *Properties* lapja) először megjelenik.

5. Próbálja ki a korábban létrehozott módosító műveletet, hogy megnézze, hogyan kezeli az EMF modell manipulációs API a forrásközi hivatkozásokat és a lusta betöltést.

6. Nyissa meg a két `.erdiagram` fájl szöveges (XMI) reprezentációját, hogy megnézze, hogyan valósul meg a kereszthivatkozás valójában. Kattintson jobb gombbal az egyes fájlokon, válassza ki az "Open With" opciót, majd a "Text editor" lehetőséget.

7. Ahogy látható, az EMF alapértelmezett XMI szerializációja a kereszthivatkozásokat az elemek tartalmazási hierarchiájának *XPath* kifejezéseként kódolja, amelyek egy adott modellel kapcsolatos útvonalat írnak le. Az útvonal egyes lépései a tartalmazási élek, numerikus pozíciós indexszel ellátva, ha többértékű gyűjteményekről van szó. Az utak lehetnek relatívak az aktuális elemhez vagy az aktuális resource-gyökérhez, vagy a cél resource *resource URI*-jához viszonyítva, azonban az ilyen pozíció alapú hivatkozások *törékenyek*. Ha a hivatkozott resource-t külön megnyitják és módosítják (például a tartalmazási listák átrendezésével), akkor a kereszthivatkozások eltörnek. Két fő megoldás létezik:

    - A lehetőség A: **belső azonosítók** (**intrinsic IDs**), ahol a domain modell tartalmaz olyan információkat, amelyek lehetővé teszik az objektumok egyértelmű azonosítását. Ehhez a lehetőséghez a nyelv készítőnek jelölnie kell egy attribútumot a metamodellben egy adott típus "elsődleges" *kulcsaként*, olyan feltétellel, hogy az attribútum értéke egyedi legyen az *resource-on belül*. Az ilyen azonosítókat az EMF XMI automatikusan használja az objektum azonosítására (resource URI-val kiegészítve, ha kereszthivatkozásról van szó); vegyük figyelembe, hogy ez a hivatkozási rendszer továbbra is törékeny, ha az ilyen azonosítókat a felhasználó megváltoztathatja.

        1. Ha kipróbálná ezt a funkciót, térjen vissza a host Eclipse-hez, nyissa meg az `erdiagram.ecore` metamodell fájlt, válassza ki a "NamedElement" ősosztály "name" attribútumát, majd a Properties lapra lépve állítsa az "ID" értékét igazra. Mentse el az .ecore metamodellt.

        2. Nyissa meg a megfelelő .genmodelt, kattintson jobb gombbal a gyökérre az fa nézetben, majd válassza ki az "Reload..." opciót. Válassza az "Ecore modelt"-t, erősítse meg az előzőleg megadott .ecore URI-t ("Load" és "Next"), majd kattintson a "Finish" gombra. Legalább az modellek kódjának újragenerálása szükséges.

        3. Indítsa el az runtime Eclipse-et az új modell pluginnel. Nyissa meg a `My.erdiagram` fájlt az Erdiagram fa szerkesztőben (jobb kattintás > Open With...), módosítsa a modellt, majd mentse el. Most az `.erdiagram` fájlokat megvizsgálva ellenőrizze, hogy a hivatkozások valóban közvetlenül az *name* attribútumra mutatnak. A kereszthivatkozások már nem sérülékenyek a tartalmi listák egyszerű módosítására, de még mindig eltörhetnek, ha a felhasználóknak lehetőségük van az objektumok átnevezésére.

    - B opció: **külső azonosítók** (**extrinsic IDs**). Néhány modell perzisztencia mechanizmus képes egy stabil és egyedi globális azonosítót (például egy 128 bites véletlenszerű UUID-t, ami gyakorlatilag ütközésbiztos) társítani modellelemekhez, még akkor is, ha azok nem tárolnak közvetlenül ilyen azonosítót. Például ez a blogbejegyzés elmagyarázza, hogy hogyan lehet konfigurálni az EMF XMI resource alapértelmezett implementációját, hogy automatikusan generálja és karbantartsa ezeket a külső UUID-kat az objektumok közötti hivatkozáshoz. A külső azonosítók szerinti kereszthivatkozás nem sérülékeny a domain modell szerkesztésekor.

Hivatkozások

- Útmutató az EclipseSource oldalán: http://eclipsesource.com/blogs/tutorials/emf-tutorial/
- Útmutató a Vogella oldalán: http://www.vogella.com/tutorials/EclipseEMF/article.html