Budapesti Műszaki és Gazdaságtudományi Egyetem Méréstechnika és Információs Rendszerek Tanszék Hibatűrő Rendszerek Kutatócsoport Modellalapú tervezés és kódgenerálás szakkör

> Eclipse alapú fejlesztés – Gyakorlat –

> > Darvas Dániel, Horányi Gergő

2012. március

1 Bevezető

Ebben a dokumentumban lépésről lépésre leírjuk a szakköri alkalmon bemutatott gyakorlati példát. Reméljük, hogy e leírás segíteni fogja a házi feladat elkészítését is.

1.1 A feladat

A feladat a Sheldon nevű Eclipse-alapú alkalmazáshoz egy olyan plugin készítése, amely egy Eclipse Forms felületen képes megjeleníteni az egyes vágányszakaszok aktuális foglaltságát az alábbihoz hasonló módon:

| Current occupancy |
|--------------------------------|
| Current track occupancy |
| 1A logical occupancy: FREE |
| 1B logical occupancy: FREE |
| 2A logical occupancy: FREE |
| 2B logical occupancy: FREE |
| 3A logical occupancy: OCCUPIED |
| 3B logical occupancy: FREE |
| 4A logical occupancy: OCCUPIED |
| 4B logical occupancy: OCCUPIED |
| Refresh! |
| |

2 A feladat megoldása lépésről lépésre

2.1 Új plugin létrehozása

1. Indítsuk el az Eclipse alkalmazást! (A kiadott virtuális gépen a helye: *C:\eclipse\eclipse.exe*.) Válasszunk egy új workspace-t, pl. *c:\workspaces\plugintest*.

| Workspace Launcher | $\overline{\mathbf{X}}$ |
|--|-------------------------|
| Select a workspace | |
| Eclipse stores your projects in a folder called a workspace. Choose a workspace folder to use for this session. | |
| Workspace: C:\Workspaces\plugintest | ▶ <u>B</u> rowse |
| Use this as the default and do not ask again | |
| | OK Cancel |

- 2. Ha megjelenik, zárjuk be a *Welcome* ablakot!
- 3. Hozzunk létre egy új projektet!
 - a. Válasszuk a *File > New > Project...* menüpontot.

b. Itt válasszuk ki a *Plug-in Project*-et, hiszen egy új plug-int szeretnénk készíteni a Sheldon programhoz. Kattintsunk a *Next* gombra.

| ∋ New Project | |
|--|----------|
| Select a wizard | > |
| Create a Plug-in Project | |
| <u>Wi</u> zards: | |
| type filter text | |
| Java Project | ^ |
| 🚽 🏶 Java Project from Existing Ant Buildfile | |
| Here General | |
| | |
| 🕀 🗁 Eclipse Modeling Framework | - |
| 🕀 🗁 Ecore Tools | |
| | |
| | |
| | |
| A Back Next > Einish A | Cancel |

c. Írjuk be a projekt nevét: *hu.bme.mit.szakkor.plugintest*. A többi beállítást nem kell megváltoztatni. Kattintsunk a *Next* gombra.

| 😂 New Plug-in Pro | ject 📃 🗆 🔀 | | | |
|---------------------------|--|--|--|--|
| Plug-in Project | | | | |
| Create a new plug-in p | roject | | | |
| | | | | |
| Project name: hu.bm | e.mit.szakkor.plugintest | | | |
| Use default locatio | n | | | |
| Location: C:\Workspi | aces\plugintest\hu.bme.mit.szakkor.plugintest Browse | | | |
| Project Settings | | | | |
| Create a Java proj | ect | | | |
| <u>S</u> ource folder: | src | | | |
| O <u>u</u> tput folder: | bin | | | |
| Target Platform | | | | |
| This plug-in is targete | d to run with: | | | |
| ⊙ <u>E</u> clipse versi | on: 3.7 🕶 | | | |
| <u>⊖ a</u> n OSGi fran | nework: Equinox 😽 | | | |
| Working sets | | | | |
| Add projec <u>t</u> to we | rking sets | | | |
| Working sets: | Select | | | |
| | | | | |
| | | | | |
| | | | | |
| ? | < <u>B</u> ack <u>N</u> ext > <u>Finish</u> Cancel | | | |

d. A következő oldalon megadhatjuk a plug-inünk tulajdonságait, pl. nevét. Emellett egyéb opciókat is beállíthatunk. Vegyük ki a pipát a *"Generate an activator…"* opció elől, erre most nem lesz szükségünk. Kattintsunk a *Finish* gombra.

| Content Enter the data required to g Properties ID: h Yersion: I Ngme: P Provider: Execution Environment: 32 Options Generate an activative | enerate the plug-in. w.bme.mit.szakkor.plugintest .0.0.qualifier Ylugintest avaSE-1.6 | | ponments |
|---|---|--------------------|----------|
| Properties ID: h Version: 1 Ngme: P Provider: Execution Environment: 3 Options Generate an activation | w.bme.mit.szakkor.plugintest .0.0.qualifier Ylugintest avaSE-1.6 | | onments |
| Properties ID: h Yersion: 1 Name: P Provider: Execution Environment: Options Separate an activation | u.bme.mit.szakkor.plugintest .0.0.qualifier Jugintest avaSE-1.6 | | onments |
| ID: h Yersion: 1 Name: P Provider: Execution Environment: 3 Options | u. bre. mit. szakkor, plugintest 0.0. qualifier Plugintest avaSE-1.6 | Envig | onments |
| Version: 1 Ngme: P Provider: 2 Execution Environment: 3 Options 3 | .0.0.qualifier Ylugintest avaSE-1.6 | | onments |
| Ngme: P Provider: Execution Environment: Ja Options | Nugintest avaSE-1.6 | Envig | onments |
| Provider: Execution Environment: 3 Options | avaSE-1.6 | Envig | onments |
| Execution Environment: | avaSE-1.6 | Envig | onments |
| Options | | | |
| Generate an activator | | | |
| | a Java class that controls the plu | iq-in's life cycle | |
| Activator: hu.bme.mi | t.szakkor.plugintest.Activator | | |
| This plug-in will make co | ntributions to the UI | | |
| Enable API Analysis | | | |
| Disk officer Acadimities | | | |
| Would you like to create a r | rich client application? | Vec | No |
| woold you like to create an | nen ellene appliedelon: | 0.702 | 0 ng |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| 2 | Rack March S | Finich | Cancel |

e. Ha az Eclipse rákérdez, hogy kívánjuk-e használni a *Plug-in Development* perspektívát, kattintsunk nyugodtan *Yes*-re.

| 😂 Ope | en Associated Perspective? |
|-------------|--|
| 2 | This kind of project is associated with the Plug-in Development perspective. |
| ~ | This perspective is designed to support efficient development and integration of plug-in projects. It adds the Plug-ins and Error Log views which are particularly useful. |
| | Do you want to open this perspective now? |
| <u>R</u> em | ember my decision |
| | Yes No |

- f. Máris elkészült a plug-inünk projektje. Azonban egyelőre még semmilyen kódot nem tartalmaz, mindössze a plug-in tulajdonságait és viselkedését leíró *manifest.mf* és a plug-in fordítását leíró *build.properties* fájlt.
- 4. Mi most nem egy általános Eclipse plug-int szeretnénk fejleszteni, hanem a Sheldon programot szeretnénk kibővíteni. Ehhez egy új *Target Definition* felvételére lesz szükség.
 - a. A *Package Explorer* nézetben kattintsunk jobb gombbal a projekt nevére. Válasszuk a *New > Target Definition* lehetőséget.

| New | ļ | 🛚 🎲 Plug-in Project |
|-----------------------|---------------------|----------------------------|
| Go Into | | 🎼 Feature Project |
| Open in New Window | | 🃬 Project |
| Open Type Hierarchy | F4 | 👚 Task |
| Show In | Alt+Shift+W | , 🔄 🚮 Component Definition |
| Copy | Ctrl+C | 📸 Product Configuration |
| 🗎 Copy Qualified Name | | 📸 Target Definition |
| 💼 Paste | Ctrl+V | 📸 Package |
| 💢 Delete | Delete | 🞯 Class |
| Remove from Context | Ctrl+Alt+Shift+Down | 💕 Interface |
| Build Path | | , 💕 Source Folder |

b. Adjunk egy nevet a Target Definitionnek: *sheldon.target*. Válasszuk ki, hogy egy *Template* alapján inicializálja a targetet, méghozzá a *Base RCP (Binary Only)* template-et használva.

| New Target Definition | _ 🗆 🗙 |
|---|------------|
| Target Definition | |
| Create a new target definition. | \bigcirc |
| Enter or select the parent folder: | |
| hu.bme.mit.szakkor.plugintest | |
| □ → hu.bme.mit.szakkor.plugintest □ → settings □ → bin □ → META-INF □ → src | |
| Eile parmat lehelden kavast | |
| nie na <u>m</u> e: sneidon.target | |
| Initialize the target definition with: | |
| Nothing: Start with an empty target definition | |
| Ourrent Target: Copy settings from the current target platform | |
| O Iemplate: Base RCP (Binary Only) ✓ | |
| ? Einish | Cancel |

- c. Adjuk hozzá a Target Definitionhöz a Sheldon mappáját! Ehhez a *Locations* rész *Add...* gombjára kattintsunk.
- d. Válasszuk ki a *Directory* opciót, majd kattintsunk a *Next*-re.

| 😂 Add Conter | nt | | | _ 🗆 🛛 |
|--|---------------------------------|---------------------|-------------------|------------------|
| Add Content | | | | |
| Select a source | of plug-ins. | | | |
| | | | | |
| Directory | | | | |
| Features | | | | |
| Software Site | 9 | | | |
| | | | | |
| | | | | |
| | | | | |
| A directory in the to the target del | e local file systen inition. | n. The plug-ins fou | nd in the directo | ry will be added |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

e. Adjuk meg a Sheldon program mappáját, ami a virtuális gépen a következő: *C:\Tools\Sheldon\eclipse*. Kattintsunk a *Finish* gombra.

| 🗢 Add | Content | _ 🗆 🔀 |
|---------------------|---|-----------------------------------|
| Add Dir Select a | ectory directory of plug-ins to add to the target. | |
| Location: | C:\Tools\Sheldon\eclipse | ▼ B <u>r</u> owse ⊻ariables |
| ? | < <u>B</u> ack <u>N</u> ext > | Einish Cancel |

f. Az ablak tetején kattintsunk a Set as Target platform linkre.

| | | | (|
|------------------------|---|---|---|
| Set as Target Platform | 1 | ? | |

2.2 Kontribúció egy nem eclipse-es kiterjesztési ponthoz

Ebben a fejezetben egy, a Sheldonban definiált kiterjesztési ponthoz készítünk egy egyszerű kiterjesztést. Ennek segítségével a plug-inünkben a későbbiekben felhasználhatjuk a Sheldonban lévő vasúti objektumokat.

- 1. Keressük ki a *Package Explorerben* a *META-INF\mainfest.mf* fájlt és kattintsunk rá duplán.
- 2. A szerkesztő alján válasszuk ki az *Dependencies* fület.

iew Dependencies Runtin

- 3. Itt kell megadnunk a plug-inünk működéséhez szükséges további plug-ineket. Mivel a kiterjesztési pont definíciója a *hu.bme.mit.modeltrainapp* plug-inben szerepel, ezért ezt mindenképp hozzá kell adnunk.
 - a. Kattintsunk a Required Plug-ins szekcióban az Add... gombra.
 - b. Válasszuk ki a *hu.bme.mit.modeltrainapp* plug-int. Kattintsunk az *OK* gombra.
 - i. Ehhez a plug-in nevének első pár betűjét be kell gépelnünk.

ii. Ha több, azonos nevű, eltérő verziójú plug-int látunk, közülük válasszuk a legfrissebbet (pl. r1003).

| Plug-in Selection | |
|---|-----------|
| Select a Plug-in: | • |
| hu. | |
| Matching items: | |
| Intermetation of the second state of the se | |
| hu.bme.mit.modeltrainapp | |
| ? | OK Cancel |

4. A szerkesztő alján válasszuk ki az Extensions fület.

| ntime | Extensions | Extension |
|-------|------------|-----------|
| | | |

- 5. Itt tudjuk majd megadni, hogy melyik kiterjesztési pontokhoz szeretnénk kiterjesztést készíteni. Először a *hu.bme.mit.modeltrainapp.extensionpoint.TrainControllerPlugin* kiterjesztési ponthoz készítünk kiterjesztést.
 - a. Kattintsunk az All Extensions szekcióban az Add... gombra.

b. Válasszuk ki a *hu.bme.mit.modeltrainapp.extensionpoint.TrainControllerPlugin* kiterjesztési pontot, majd kattintsunk a *Finish* gombra.

| New Extension | _ 🗆 🗙 |
|---|-------------|
| Extension Point Selection | |
| Create a new hu.bme.mit.modeltrainapp.extensionpoint.TrainControllerPlugin extension. | -) |
| Extension Points Extension Wizards | |
| Extension Point filter: | |
| - hu.bme.mit.modeltrainapp.extensionpoint.TrainControlGUI | |
| Show only extension points from the required plug-ins Extension Point Description: <u>hu.bme.mit.modekrainapp.extensionpoint.TrainContr</u> (or discription explicitle) | ollerPlugin |
| Available templates for hu.bme.mit.modeltrainapp.extensionpoint.traincontrollerpli | ugin: |
| () < Back Next > Finish | Cancel |

c. Adjuk meg az újonnan felvett kiterjesztés adatait: ID: *hu.bme.mit.szakkor.plugintest* Name: *Plugin test*

| All Extensions | ↓ <mark>a</mark> 🖻 | Extension Details |
|--|--------------------|--|
| Define extensions for this plug-in in the following section. | | Set the properties of the selected extension. Required fields are denoted by "*". |
| type filter text | | ID: bu hme mit szakkor plugintest |
| | Add | Name: Plugin test |
| | Remove | |

d. Emellett ez a konkrét kiterjesztési pont elvár még egy "klienst" is, egy osztályt, amely megvalósít egy megadott interfészt. Ezt a kiterjesztési pont nevén jobb gombbal kattintva megjelenő menüből a *New > client* kiválasztásával vehetjük fel.

| L L | | | <u> </u> |
|---------|---|----------|----------|
| New | • | 🗙 client | AC |
| Delete | | | Re |

e. Adjuk meg a kliensosztályt. Ehhez az *object* mezőbe kell beírnunk a megfelelő osztály teljes nevét. Ilyet még nem hoztunk létre, de erre képes az Eclipse is automatikusan. Adjuk meg a következő nevet az *object* mezőben: *hu.bme.mit.szakkor.plugintest.LocalRepository*.



- f. Kattintsunk az *object** linkre, majd a *Finish* gombra.
- g. Ez után megnyílik a frissen elkészült *LocalRepository.java* fájl. Ez fogja tartalmazni a vasúti objektumaink tárolóit, melyet a Sheldon program majd az *initialize* függvény meghívásával inicializál. Készítsük el az osztály alábbi, nagyon egyszerű implementációját:

```
package hu.bme.mit.szakkor.plugintest;
```

```
import hu.bme.mit.modeltrainapp.extensions.TrainControllerPlugin;
import hu.bme.mit.modeltrainapp.model.TrainObjectRepository;
public class LocalRepository implements TrainControllerPlugin {
    private static TrainObjectRepository repo = null;
    public LocalRepository() {
    }
    @Override
    public void initialize(TrainObjectRepository _repo) {
        repo = _repo;
    }
    public static TrainObjectRepository getRepo() {
        return repo;
    }
}
```

h. Már alig várjuk, hogy teszteljük első Eclipse plug-inünket. Ezért egészítsük ki az *initialize* függvényt a következő sorral:

```
System.out.println("Loaded signals: " + repo.getSignals().size());
```

Ennek segítségével az inicializálásról tudomást szerzünk, ha a konzolt figyeljük.

- i. Mentsünk minden fájlt! (Ctrl+Shift+S vagy a menüsoron: 💼)
- 6. Még egy beállítást meg kell tennünk, hogy futtatni tudjuk a plug-inünket: be kell állítanunk, hogy mit kell futtatni.
 - a. Kattintsunk a *Run > Run configurations...* menüpontra!

b. Válasszuk a fastruktúrában az *Eclipse Applicationt*, majd a *Main* fülön a *Program to run* szekcióban jelöljük be a *Run a product* lehetőséget, a listából pedig válasszuk ki a *hu.bme.mit.modeltrainapp.product* elemet.

| Run Configurations | | | X |
|---|--------------------------------------|---|-------------------|
| Create, manage, and run cor Create a configuration to launch an t | nfigurations Eclipse application. | | |
| Image: Second | Name: Eclipse Application | Plug-ins Configuration Tracing Configuration Image: Second Sec | t Common |
| Filter matched 10 of 10 items | | | Apply Revert |
| ? | | | <u>R</u> un Close |

- c. Kattintsunk a *Close* gombra.
- 7. Most már futtathatjuk az alkalmazásunkat. Ehhez kattintsunk a *Run > Run* menüpontra vagy a menüsoron a **O** gombra.
 - a. Ha az Eclipse rákérdez, milyen konfigurációt futtassunk, válasszuk az *Eclipse Application* lehetőséget (hiszen ezt állítottuk be az előbb a céliainknak megfelelően).
- Ha mindent jól csináltunk, elindul a Sheldon alkalmazás, illetve a konzolablakban láthatjuk a diagnosztikai kimenetünket a plug-inek betöltése során.

| 🥺 Error Log 🖉 Tasks 🔝 Problems 🔗 Search 💷 Console 🛛 |
|---|
| Eclipse Application [Eclipse Application] C:\Program Files\Java\jre6\bin\javaw.exe (2012.03.) |
| !MESSAGE Signalbox stylesheet has loaded successfully. |
| Loading extensions |
| Extension loading |
| Initializing |
| Loaded signals: 4 |
| Extension loaded. |

9. Zárjuk be a futó Sheldon alkalmazást.

2.3 Kontribúció a grafikus felhasználói felülethez

Most megpróbálunk egy új nézetet hozzáadni a Sheldon program felületéhez.

Valójában egy nézet hozzáadásához több kiterjesztési ponthoz is kell majd implementációt készítenünk: egyet a nézet definiálására, egyet a nézet megjelenítésére, egyet pedig a nézet megjelenítésére szolgáló eszköztár-elemhez.

- 1. Vegyük fel a nézet megjelenítéséhez szükséges függőségeket!
 - a. Keressük ki a *Package Explorerben* a *plugin.xml* fájlt és kattintsunk rá duplán.

b. A szerkesztő alján válasszuk ki az Dependencies fület.

iew Dependencies Runtir

- c. Kattintsunk a Required Plug-ins szekcióban az Add... gombra.
- d. Vegyük fel egymás után az alábbi plug-ineket:
 - i. org.eclipse.ui
 - ii. org.eclipse.core.runtime
 - iii. org.eclipse.ui.forms

org.eclipse.ui (3.7.0)
 org.eclipse.core.runtime (3.7.0)
 org.eclipse.ui.forms (3.5.101)

- 2. Készítsünk egy új View kiterjesztést!
 - a. A szerkesztő alján válasszuk ki az Extensions fület.

ntime Extensions Extension

- b. Kattintsunk az All Extensions szekcióban az Add... gombra.
- c. Válasszuk ki az *org.eclipse.ui.views* kiterjesztési pontot és kattintsunk a *Finish* gombra.

| New Extension | _ 🗆 🖂 |
|---|--------|
| Extension Point Selection | |
| Create a new org.eclipse.ui.views extension. | |
| | |
| Extension Points Extension Wizards | |
| Extension Point filter: ui.vi | |
| -1 org.eclipse.ui.viewActions | |
| - forg.eclipse.ui.views | |
| | |
| | |
| | |
| | |
| | |
| | |
| Show only extension points from the required plug-ins | _ |
| Extension Point Description: orq.eclipse.ui.views | |
| (no description available) | |
| | |
| | |
| Available templates for oral estince us views: | |
| Available compares for or groups an views. | |
| | |
| | |
| | |
| | |
| | |
| | |
| (?) < Back Next > Finish | Cancel |

d. Itt nem kell a nézethez semmilyen adatot megadnunk.

e. Kattintsunk jobb gombbal az *org.eclipse.ui.views* elemen és válasszuk a *New* > *category* menüpontot.

| New | ſ |
|------------------|---|
| . Stickyőew | |
| Delete | |
| Chau Description | ŝ |

- f. Adjuk meg az új nézet adatait:
 - i. id: hu.bme.mit.szakkor.plugintest.view1
 - ii. name: Test view
 - iii. class: hu.bme.mit.szakkor.plugintest.TestView
 - iv. allowMultiple: *false*

| Extension Element | Details | |
|--------------------------|---|--------|
| Set the properties of | "view". Required fields are denoted by "*". | |
| id*: | hu.bme.mit.szakkor.plugintest.view1 | |
| name*: | Test view | |
| <u>class*:</u> | hu.bme.mit.szakkor.plugintest.TestVie | Browse |
| category: | | Browse |
| icon: | | Browse |
| fastViewWidthRatio: | | |
| allowMultiple: | false | ~ |
| restorable: | true | ~ |

g. A *class* mezőben most is egy még nem létező osztály nevét adtuk meg. Kattintsunk a *class** hivatkozásra, a megjelenő ablakban pedig a *Finish* gombra.

| 😂 New Java Cla | ss | D |
|--|--|-----------------|
| Java Class Create a new Java | class. | |
| Source fol <u>d</u> er: | hu.bme.mit.szakkor.plugintest/src | Browse |
| Package: | hu.bme.mit.szakkor.plugintest | Bro <u>w</u> se |
| Enclosing type: | | Bro <u>w</u> se |
| Na <u>m</u> e: | TestView | |
| Modifiers: | O public O default O private O protected □ abstract □ final □ static | |
| Superclass: | org.eclipse.ui.part.ViewPart | Browse |
| Interfaces: | | Add |
| Which method stub: | s would you like to create? | |
| | public static <u>v</u> oid main(String[] args) | |
| | ✓ Constructors from superclass | |
| Do you want to add | Ingerited abstract methods comments? (Configure templates and default value <u>here</u>) Generate comments | |
| | | |
| ? | Einish | Cancel |

- h. A TestView osztály implementálását halasszuk későbbre.
- 3. Készítsünk egy új Command kiterjesztést, amely képes megjeleníteni a TestView nézetet!
 - a. Térjünk vissza a *plugin.xml* fájlhoz és válasszuk ki az *Extensions* fület.
 - b. A korábbiakhoz hasonlóan vegyünk fel egy *org.eclipse.ui.commands* kiterjesztést.

| New Extension | | | | _ 🗆 🔀 |
|---|------------------------|------------------|--------|--------|
| Extension Point Sele | tion | | | |
| Create a new Commands | extension. | | | |
| | | | | |
| Extension Points Extens | ion Wizards | | | |
| Extension Point filter: or | g.eclipse.ui.commanc | ls | | |
| -([*] lorg.eclipse.ui.comm | nds | | | |
| Show only extension p | oints from the require | ed plug-ins | | |
| Extension Point Description | n: <u>Commands</u> | | | |
| The org.eclipse.ui.commands extension point is used to declare commands and command categories, using the command and category elements. A command is an abstract representation of some semantic behaviour, but not its actual implementation. This allows different developers to contribute specific behaviour for their individual narts. For example, there might be a "naste" Available templates for commands: | | | | |
| & "Hello, World" comma | and contribution | | | |
| ? | < <u>B</u> ack |] <u>N</u> ext > | Einish | Cancel |

c. Vegyünk fel egy új parancsot az *org.eclipse.ui.commands* elemre jobb gombbal kattintva, a *New > command* menüpontot választva.

| Org.eclipse.ui.commands | 1 | | |
|-------------------------|------------------|----|--------------------------|
| | New | ۰. | x activeKeyConfiguration |
| | Delete | | x category |
| | | | 🗴 command |
| | Show Description | | 🗴 commandParameterType |

- d. Adjuk meg a *command* adatait:
 - i. id: hu.bme.mit.szakkor.plugintest.ShowTestView
 - ii. name: Show Test View
 - iii. defaultHandler: hu.bme.mit.szakkor.plugintest.ShowTestView
- e. Létre kell hoznunk a *defaultHandler*-nél megadott osztályt. Kattintsunk a *defaultHandler* linkre! Most nem hagyunk minden beállítást alapértelmezetten.
 - i. Kattintsunk a *Superclass* mező melletti *Browse* gombra.

Superclass:

Browse...

ii. Válasszuk ki az *org.eclipse.core.commands.AbstractHandler* osztályt és kattintsunk az *OK*-ra.

| Superclass Selection | _ 🗆 🔀 |
|---|----------------------|
| hoose a type: | - |
| org.eclipse.core.commands.AbstractHandler | |
| <u>A</u> atching items: | |
| AbstractHandler - org.eclipse.core.commands | |
| Workspace matches | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| org.eclipse.core.commands - C:\Tools\Sheldon\eclipse\plugins\org.eclipse.core.commands_3.6. | 0.I20110111-0800.jar |
| | |
| (f) | Cancel |

iii. Kattintsunk a *Finish* gombra.

| 😂 New Java Clas | ss | _ 🗆 🔀 |
|-------------------------|---|----------------|
| Java Class | | |
| Create a new Java | class. | |
| | | |
| Source fol <u>d</u> er: | hu.bme.mit.szakkor.plugintest/src | Browse |
| Pac <u>k</u> age: | hu.bme.mit.szakkor.plugintest | Browse |
| Enclosing type: | | Browse |
| Na <u>m</u> e: | ShowTestView | |
| Modifiers: | gublic Odefault Oprivate Oprotected | |
| | abstract final static | |
| <u>S</u> uperclass: | org.eclipse.core.commands.AbstractHandler | Brows <u>e</u> |
| Interfaces: | 0 org.eclipse.core.commands.IHandler | <u>A</u> dd |
| | | |
| | | Remove |
| Which method stubs | would you like to create? | |
| | public static void main(String[] args) | |
| | Constructors from superclass | |
| Do you want to add | ✓ Inherited abstract methods | |
| | Generate comments | |
| | | |
| | | |
| | | |
| ? | Einish | Cancel |

4. A létrejövő *ShowTestView* osztály *execute* metódusában kell implementálnunk a nézet megjelenítését. Ehhez a fájlban az alábbi kód szerepeljen:

```
package hu.bme.mit.szakkor.plugintest;
import hu.bme.mit.modeltrainapp.rcp.Logger;
import org.eclipse.core.commands.AbstractHandler;
import org.eclipse.core.commands.ExecutionEvent;
import org.eclipse.core.commands.ExecutionException;
import org.eclipse.core.commands.IHandler;
import org.eclipse.core.runtime.IStatus;
import org.eclipse.ui.IWorkbenchPage;
import org.eclipse.ui.PartInitException;
import org.eclipse.ui.PlatformUI;
public class ShowTestView extends AbstractHandler implements IHandler {
     @Override
     public Object execute(ExecutionEvent event) throws ExecutionException {
          try {
               PlatformUI
                          .getWorkbench()
                          .getActiveWorkbenchWindow()
                          .getActivePage()
                          .showView("hu.bme.mit.szakkor.plugintest.view1", null,
                                    IWorkbenchPage.VIEW ACTIVATE);
          } catch (PartInitException e) {
               Logger.log(IStatus.ERROR,
                          "Error occured during the initialization of view.",
                          e);
               Logger.showPopup(IStatus.ERROR, "Error",
                          "Error occured during the initialization of view.");
          }
          return null;
     }
}
```

- a. A "hu.bme.mit.szakkor.plugintest.view1" helyén mindig az aktuális nézet pontos IDjét kell megadni.
- b. A Logger osztály a Sheldon naplózó és hibajelző osztálya, hiba esetén ez fog értesíteni minket a grafikus felületen.
- c. Mentsük is a munkánkat!
- 5. Készítsünk egy új menükiterjesztést, amely képes megjeleníteni a nézetünk megjelenítésére szolgáló gombot!
 - a. Térjünk vissza a *plugin.xml* fájlhoz és válasszuk ki az *Extensions* fület.
 - b. A korábbiakhoz hasonlóan vegyünk fel egy org.eclipse.ui.menus kiterjesztést.
 - c. Vegyünk fel ez alá egy *menuContribution* elemet a korábbiakhoz hasonlóan.

| org.eclipse.ui.menus | | | Up | | |
|----------------------|--------------------|---|------------|----------|----------|
| | New | • | 🗴 group | | F |
| | Delete | | 🗴 menuCont | ribution | <u>i</u> |
| | | | 🗴 widget | | |
| 1 | 📼 Show Description | | 1 | | |

d. A *menuContribution* elem *locationURI* mezőjébe írjuk be a következőt: *toolbar:org.eclipse.ui.main.toolbar*. Ennek az a jelentése, hogy a definiált kontribúció az ablak fő eszköztárához járul hozzá.

| Extension Ele | ement Details |
|------------------------|--|
| Set the proper "*", | ties of "menuContribution". Required fields are denoted by |
| locationURI*; | toolbar:org.eclipse.ui.main.toolbar |
| <u>class:</u> | Browse |
| allPopups: | false |

- e. A menuContribution gyerekeként vegyünk fel egy új toolbar elemet.
- f. A toolbar elem gyerekeként vegyünk fel egy új command elemet.

| interference i | | Up | | |
|--|------------------|------|-----------|---|
| hu.bme.mit.szakkor.plugint | :est.toolbar1 (t | Down | | _ |
| | New | | command | |
| | Delete | | 🗴 control | |

- g. A *command* elem beállításai a következők legyenek:
 - i. commandId: hu.bme.mit.szakkor.plugintest.ShowTestView
 - ii. label: *Show Test View*
 - iii. tooltip: Show Test View

| Extension Elen | nent Details | |
|------------------|--|-----------------|
| Set the properti | es of "command". Required fields are denoted b | у " * ". |
| commandId*: | .bme.mit.szakkor.plugintest.ShowTestView | Browse |
| label: | Show Test View | |
| id: | | |
| mnemonic: | | |
| icon: | | Browse |
| disabledIcon: | | Browse |
| hoverIcon: | | Browse |
| tooltip: | Show Test View | |
| helpContextId: | | |
| style: | push | ~ |
| mode: | | ~ |

- h. Mentsünk minden módosítást. (💼)
- 6. Végre eljutottunk odáig, hogy kipróbálhatjuk, mit alkottunk. Ha futtatjuk a programot (♥) és mindent jól csináltunk, megjelenik az eszköztáron egy *Show Test View* gomb, amire ha rákattintunk, megnyílik az új (és egyelőre töküres) nézetünk.



2.4 Nézet elkészítése Eclipse Forms API-val

Ebben a fejezetben egy egyszerű nézetet készítünk az Eclipse Forms API segítségével.

- 1. Keressük ki a *TestView.java* fájlt a *Package Explorerben* és kattintsunk rá duplán! Végre megtöltjük tartalommal.¹
- 2. Vegyünk fel két változót az osztályba:

private FormToolkit toolkit;
private Form form;

- a. A *form* változó fogja az űrlapunkat reprezentálni, míg a *toolkit* változó az elemek példányosítását fogja segíteni.
- 3. A createPartControl metódusban inicializáljuk az űrlapot az alábbiak szerint:

```
// Toolkit példányosítása
toolkit = new FormToolkit(parent.getDisplay());
```

// új Form példány készítése
form = toolkit.createForm(parent);

// az űrlap elrendezésének beállítása
form.getBody().setLayout(new ColumnLayout());

```
// az űrlap címének beállítása
form.setText("Example Sheldon Eclipse Forms plugin");
```

¹ Számos osztály neve nem egyértelmű és mind a java.awt, mind az org.eclipse.swt csomagban megtalálható. A java.awt-ben lévő osztályokra NEM lesz szükségünk most, sehol se válasszuk ezt!

4. Ha most kipróbálnánk a plug-inünket, ezt látnánk:

| Test view 🛛 | |
|--------------------------------------|--|
| Example Sheldon Eclipse Forms plugin | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

5. Hozzunk létre egy szekciót az űrlapon!

```
// szekció létrehozása (leírássa, címsorral, kibontva)
Section occup = toolkit.createSection(form.getBody(),
        Section.DESCRIPTION | Section.TITLE_BAR | Section.EXPANDED);
// szekció címének beállítása
occup.setText("Current occupancy");
// szekció leírásának beállítása
occup.setDescription("This section shows the current track occupancy.");
// szekció belsejéhez elemeket tároló objektum létrehozása
Composite occupBody = toolkit.createComposite(occup);
// szekció belső elrendezésének beállítása²
ColumnLayout cl = new ColumnLayout();
cl.maxNumColumns = 1;
occupBody.setLayout(cl);
```

```
// szekciót kezelő belső objektum beállítása
occup.setClient(occupBody);
```

² A ColumnLayout használata javítja a gyakorlaton előkerült problémát, miszerint a Label-ek nem idomulnak a bennük szereplő szöveg szélességének megváltozásához.

6. Ha most kipróbálnánk a plug-inünket, ezt látnánk:

| 🗖 Sheldon 📃 🗆 🔀 |
|--|
| File View Simulator/Reality |
| 🕴 🚥 🔋 🖸 🕖 🌡 📲 🎓 📴 🗄 Show Test View 👘 🛃 * 🖗 * 🔅 🏷 |
| 🖹 🔚 Default perspective |
| Test view 🛛 🗖 |
| Example Sheldon Eclipse Forms plugin |
| Current occupancy This section shows the current track occupancy. |
| |

- 7. Vegyünk fel minden vágányszakaszhoz egy-egy címkét (legalábbis azokhoz, amelyekre a *getGraphicalRepresentation()* nem null értéket ad vissza ezek a GUI-n tényleg megjelenítendő vágányszakaszok).
 - a. Ehhez először vegyünk fel egy az osztályba egy hash-táblát, ami a vágányszakaszok azonosítóit és a címkéket összerendeli.

```
private Map<Integer, Label> trackLabels = new HashMap<Integer, Label>();
```

b. Kérdezzük le a LocalRepository-ból az összes vágányszakaszt (TrackElement objektumot), majd mindegyikhez készítsünk 1-1 címkét.

```
for (TrackElement te : LocalRepository.getRepo().getTrackElements()) {
    if (te.getGraphicalRepresentation() != null) {
        // a Label objektum létrehozása (dummy szöveggel)
        trackLabels.put(te.getId(),
            toolkit.createLabel(occupBody, "..."));
    }
```

```
}
```

8. Ha most kipróbálnánk a plug-inünket, ezt látnánk:

| 🗖 Sheldon | | | - | | | × |
|---|---|---|----|---|---|---|
| File View Simulator/Reality | | | | | | |
| 🕴 🚥 🔋 📭 🕖 📳 🔛 🕋 🔛 🔁 🗄 Show Test View | 1 | Ł | ų. | | 1 | Ģ |
| 🖹 🖪 Default perspective | | | | | | |
| Test view 🛛 | | | | 1 | | |
| Example Sheldon Eclipse Forms plugin | | | | | | ٦ |
| | | | | | | |
| This section shows the current track occupancy. | | | | | | ч |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

9. Készítsünk egy új metódust, ami minden vágányelemhez kitölti a címke szövegét az aktuális foglaltság alapján!

a. Hívjuk meg a *refreshOccupancy* metódust a *createPartControl* metódus végén! 10. Ha most kipróbálnánk a plug-inünket, ezt látnánk:



11. Készítsünk egy gombot, amivel frissíteni tudjuk a foglaltságértékeket:

```
// gomb létrehozása
Button refresh = toolkit.createButton(occupBody, "Refresh!", SWT.NONE);
// eseménykezelő beállítása
refresh.addListener(SWT.Selection, new Listener() {
    @Override
    public void handleEvent(Event event) {
        // a gomb lenyomásakor végrehajtandó metódus
        refreshOccupancy();
    }
});
```

12. Indítsuk el a programot! Nyissuk meg a nézetünket, majd kapcsoljunk szimulátor módba a *Simulator > Simulator with 2 train* menüponttal. Ha most rákattintunk a *Refresh!* gombra, az alábbi képet láthatjuk:

| 🗖 Sheldon | _ 🗆 🛛 |
|---|---------------|
| File View Simulator/Reality | |
| 🕴 💶 🔋 🚺 🕖 🎚 🔛 🎓 🔡 🗄 Show Test View | 1 🖢 = 🖓 = 1 🏷 |
| 😰 🖽 Default perspective | |
| Test view 🛛 | - 8 |
| Example Sheldon Eclipse Forms plugin | |
| Current occupancy | |
| This section shows the current track occupancy. | |
| 1A logical occupancy: FREE | |
| 1B logical occupancy: FREE | |
| 2A logical occupancy: FREE | |
| 2B logical occupancy: FREE | |
| 3A logical occupancy: FREE | |
| 3B logical occupancy: FREE | |
| 4A logical occupancy: FREE | |
| 4B logical occupancy: FREE | |
| Refresh! | |
| | |

a. Látható, hogy a 3A és 4A szakaszon állnak vonatok.

b. Válasszuk ki a *View > Signalbox* menüpontot! Itt grafikusan is láthatjuk a foglaltságokat. Látható, hogy valóban a 3A és 4A szakaszok foglaltak aktuálisan.



Megjegyzés: lehetőség van arra is, hogy a kijelzést automatikusan frissítsük. Minden *ModelTrainObject*-ből leszármazott objektum, így az egyes vágányszakaszok is képesek értesítést küldeni, ha módosult az állapotuk.

Ehhez a track.addModificationListener(listener); metódushívást kell végrehajtanunk, ahol a listener egy IModelTrainObjectModificationListener interfészű objektum. Amennyiben a példában a track objektum változik, úgy meghívja a listener objektum onTrainObjectModification metódusát.

A gyakorlatba átültetve az alábbi feladataink lennének, ha automatikusan frissíteni szeretnénk a kijelzést:

• A TestView osztályt kiegészítjük, hogy implementálja az IModelTrainObject-ModificationListener interfészt.

public class TestView implements IModelTrainObjectModificationListener { ...

• Implementáljuk a módosulást kezelő függvényt.

- }
- Feliratkozunk az inicializálás során az összes vágányszakasz eseménykezelőjére.

```
for (TrackElement te : LocalRepository.getRepository().getTrackElements()) {
    te.addModificationListener(this);
```

}

2.5 Kommunikáció a vezérlők felé

Egy egyszerű felületen keresztül lehetőségünk van arra is, hogy a Sheldonból (egy egy bájtos) üzenetet juttassunk el egy vezérlőhöz. Ehhez a vezérlőt reprezentáló MbedController osztály sendByte(byte message) metódusát használhatjuk.

A vezérlő számára ez az üzenetküldés egy speciális szinkronizációként látszik. E szinkronizáció csatornájának neve kötelezően sheldon, típusa broadcast chan, a küldött adat pedig az int típusú sheldon_content változóból olvasható ki.

Így tudunk például az 1-es ID-jű vezérlőnek egy fix tartalmú üzenetet eljuttatni egy gombra kattintva:

```
Button sendMessage = toolkit.createButton(occupBody, "Send!", SWT.NONE);
sendMessage.addListener(SWT.Selection, new Listener() {
    @Override
    public void handleEvent(Event event) {
        LocalRepository.getMbedControllerByID(1).
            sendByte((byte)42);
    }
});
```

Ha ezt arra szeretnénk felhasználni az 1-es vezérlőben, hogy addig ne induljon el a vezérlő futása, amíg nem kattintunk egy gombra rá a Sheldonban, azt például így tehetjük meg:



Opcionálisan őrfeltételt is írhatunk az élre, pl. sheldon_content == 42. Természetesen ez nem kötelező.

Figyelem! Ezen a csatornán keresztül nem lehet a Sheldon program felé üzenetet küldeni, azaz sheldon! szinkronizációt nem lehet csinálni.