Uživatelská příručka

K používání frameworku je nutné mít nainstalované prostředí Eclipse a databázi PostgreSql. Testy jsou spustitelné jen z Unixových prostředí – bohužel se nám nepodařilo zprovoznit MWE2 workflow v OS Windows.

Obsah přiloženého CD je nutné zkopírovat na disk například do složky /home/uzivatel/migdb/, aby bylo možné framework otestovat.

Vývojové prostředí

Vývojové prostředí Eclipse funkční v OS Linuxem Mint Olivia je uloženo na přiloženém CD ve složce eclipse-migdb. V této složce je zip s vývojovým prostředím, který je nutné rozbalit. Po spuštění vývojového prostředí je nutné vybrat si jméno workspace. Pokud vývojové prostředí startuje bez výběru workspace je doporučené vytvořit si workspace přes menu File->Switch Workspace->další a zadat název workspace, například workspace-migdb. Po otevření workspace je nutné naimportovat projekty z CD přes menu File->Import. V dialogovém okně Select je nutné vybrat General->Existing Projects into workspace. V dialogovém okně Import Projects je nutné vybrat volbu Select root directory a stistknout tlačítko Browse. Po nalezení kořenové složky se zkopírovaným obsahem CD (zde v návodu /home/uzivatel/migdb/) je nalezeno 8 projektů, které se mají naimportovat. Import je zakončen tlačítkem Finish. V případě problému s importem projektů je na CD ještě zip s projekty, který je možné přenést, rozbalit a naimportovat uvedeným způsobem.

Před dalším postupem je nutné projekty přeložit viz. menu Project->Clean (je nutné, aby bylo zaškrtnuto build automatically) a potvrdit.

Obnovení linků

K prácí s linky slouží složka scripts, která musí být nakopírována na úroveň migdb projektů.

migdb_config.cfg

Konfigurační soubor obsahující cesty k eclipse a projektu. Před použitím je nutné zkopírovat si migdb_config.cfg do svého domovského adresáře a nastavit si cesty k eclipsu a midgb repositáři. Tento soubor je jediným konfiguračním souborem, který je při práci se scripty potřebný, v případě přidání dalšího konfiguračního souboru je nutné upravit script check_config.sh

Použití

Osobně jsem si raději vytvořil aliasy v domovské složce .bash_aliases pro volání scriptů odkudkoliv. Aliasováním nebo přidáním složky scripts do PATH je

zajištěno, že budete vždy používat aktuální scripty.

Script renew_links

Tento script slouží k obnově relativních linků. Tento script obnoví linky, před testováním workflow musí být spuštěn.

Otestování

K otestování slouží workflow zmiňované v kapitole Testování projektu Migdb. Výstupní soubory všech testů je možné najít ve složce output-tests umístěné v projektu migdb.testing.run. Vstupní data pro testy jsou definovány v jednotlivých workflow. Kromě již zmíněných workflow byly připraveny workflow test_app_matcher1_dip.mwe2 a test_app_matcher2_dip.mwe2 v balíčku migdb.testing.app.oracle.run a workflow test_code_generator_dip.mwe2 v balíčku migdb.testing.migdb_executer.run sloužící k otestování výsledků této diplomové práce. Každé workflow je spustitelné přes své kontextové menu Run as->1 MWE2 workflow nebo přes nabídku Run-> Run as->1 MWE2 workflow.

Struktura workflow:

-- nutné unikátní

```
module migdb.test_code_generator_dip

import org.eclipse.emf.mwe.utils.*
… --importy

var OUTPUT_BASE_DIR = "output-tests/gen_SQL_dip"
var IN_APP_OPS = "${QVTO_DIR}/tests/operations/app"
var IN_APP_STR = "${QVTO_DIR}/tests/structures/app"
… -- proměnné, nejdůležitější jsou OUTPUT_BASE_DIR s výstupy
… -- proměnné IN jsou složky obsahující vstupní data
… -- proměnná typu testTransform by neměla být měněna

-- samotná definice workflow
TestWorkflow{
   -- komponenta mazající původní obsah kořenové složky
      component = org.eclipse.emf.mwe.utils.DirectoryCleaner {
            directory = "${OUTPUT_BASE_DIR}"
      }

      … -- nutná registrace metamodelů
      bean=org.eclipse.emf.mwe.utils.StandaloneSetup {
            platformUri=".."
            registerGeneratedEPackage = "mm.app.AppPackage"
      }

      -- testovací komponenta, pro účely ozkoušení je dobré měnit jen parametry
      -- qvtInput. Parametr qvtComparison slouží k otestování výstupu, při je
      -- zadání jako prázdný je automaticky test splněn
      -- tento test je nutný k vytvoření schematu
      component = TestComponent{
            transformationFile = "${ORM_TRANSFORMATION}"
            outputParentUri = "${OUTPUT_BASE_DIR}"
```

```
            qvtInput = "${IN_APP_STR}/structure_tab_parent_joined.qvto"
            qvtComparison =""//COMPARISONS_RDB_STRUCTURE - just note
            qvtComparison ="${EMPTY_ERRORS}"
            testDescription = "schema"
      }

      -- komponenta sloužící k vygenerování kódu schematu SQL
      component = CodeGenComponent2 {
            withoutModel = false
            slot = "${RDB_STRUCTURE_SLOT}"
            generator = SchemaGenerator{
                  filename = "${SCHEMA_FILE_STRING}.sql"
            }
            outputPath = "${OUTPUT_BASE_DIR}/schema"
      }

       -- test k vygenerování operací
       -- parametry qvtComparison musí být zadané
       component = TestComponent{
            transformationFile = "${TEST_TRANSFORMATION}"
            outputParentUri = "${OUTPUT_BASE_DIR}"
            qvtInput = "${IN_APP_STR}/structure_tab_parent_joined.qvto"
            qvtInput = "${IN_APP_OPS}/addParent1.qvto"
            qvtComparison = ""//COMPARISONS_APP_STRUCTURE - just note
            qvtComparison = ""//COMPARISONS_RDB_STRUCTURES_DIR - just note
            qvtComparison = ""//COMPARISONS_RDB_OPS_DIR - just note
            qvtComparison = "${EMPTY_ERRORS}"
            testDescription = "${OUTPUT_BASE_DIR} simple adding parency FK"
      }


      -- komponenta generující kód pro operace
      component = CodeGenComponent2 {
            withoutModel = false
            slot = "${RDB_OPS_SLOT}"
            generator = Generator {
                  filename = "AddParent.sql"
            }
            outputPath = "${OUTPUT_BASE_DIR}"
      }
}
```

Výstupní soubory jsou tedy naleznutelné v projektu migdb.testing.run v jeho potsložce output-tests. Pro každou TestComponent v lokaci specifikované parametrem outputParentUri a pro CodeGenComponent2 v lokaci specifikované parametrem outputPath.

K tomu, aby bylo možné definovat TestComponentu je nutné vytvořit testovací data, tato data se dají vytvořit podle ukázkových příkladů v složce main/tests v projektu migdb.qvto.

Struktura souboru aplikační struktury:

```
//nutné importy
import builder_app;
import queries_app;
// specifikace typu modelu transformace
modeltype APP uses "http://www.collectionspro.eu/jam/mm/app";

//specifikace transformace
transformation addClass1_required(out inoutModel : APP);
```

```
//tělo transformace
main(){
	//vytvoření entity _integer
	var integer : PrimitiveClass := _integer();
	//vytvoření třídy Tab s idProperty idTab
	var tab : StandardClass := _class("Tab", _idProperty("idTab", integer),
OrderedSet{});
	//vytvoření aplikační struktury
	_appStructure(OrderedSet{integer, tab});
}
```

Struktura souboru s seznamem operací

```
//nutné importy
import builder_app;
import queries_app;
// specifikace typu modelu transformace
modeltype APP uses "http://www.collectionspro.eu/jam/mm/app";

//specifikace transformace
transformation addParent1(out inoutModel : APP);

//tělo transformace
main(){
	//vytvoření operace addParent
	var addParent : APP::ops::AddParent :=_addParent("Man", "Person");
	//vytvoření kořenového elementu Operations
	_appOperations(OrderedSet{addParent});
}
```

Konstruktory použité v obou příkladech souborů jsou uloženy v souboru builder_app.qvto. Je vhodné vytvářet testovací soubory v shodných složkách jako již existující testovací data. V opačném případě je nutné vytvořit symbolické linky na importované soubory.