MigDB - testy

Testování je důležitou, dalo by se říci až nedílnou součástí, každého úspěšného projektu. Přispívá ke zkvalitnění vyvíjené aplikace a ověřuje, zda byly splněné všechny požadavky zákazníka na systém. Zákazník také často požaduje stoprocentní otestování dodávané aplikace. Tento požadavek se v reálném vývoji aplikací nedá splnit, ale můžeme se mu přiblížit tím, že budeme postupovat podle předem naplánovaných procesů (viz. dokument plánování testů) a hlavně nebudeme testování odkládat až na konec projektu. Je dokázáno, že čím dřív se chyba opraví, tím má menší ekonomické dopady na celý projekt.

Jelikož jsme aplikaci vyvíjeli v iteracích, objeví se zde popis všech testů, které jsme v jednotlivých fázích projektu prováděli.

Testy prvního prototypu

První prototyp, který byl podroben testům, se zabýval modifikací vytvořeného modelu v ecore a generováním SQL příkazů.

Vzhledem k použitému programovacímu jazyku - Java, byl jako framework pro testování zvolen **jUnit**. Tento framework se opírá o tzv. *jednotkové testy*, které lze chápat jako testování jednotlivých částí. Ideální stav je vše pokryté testy (tzv. code coverage) a pro produkční nasazení musí být všechny testy úspěšné.

Testované třídy

následuje seznam testovaných tříd, popis metodiky a úspěšnost

migdb.sql.DialectSQLTest

Tento test si klade za cíl testování správné implementace rozhraní SQLDialect. Cílem je tedy ověřit, že všechny podporované dialekty jsou popsány. V současné době je podporován pouze PostgreSQL, nicméně při přidání dalšího dialektu je při současném návrhu testu snadněji zaručeno, že dojde k otestování všech nezbytných údajů - test se provádí nezávisle na konkrétním dialektu.

Test pokrývá jedinou metodu, kterou výše uvedené rozhraní definuje.

Test je úspěšný.

migdb.ecore.EcoreUtilTest

Během tohoto testu se provádí v současnosti všechny metody třídy EcoreUtil, které mění model. Jedná se o přidání/přejmenování/smazání třídy/atributu (celkem tedy 6 metod). Během testu je postupně vytvořena třída, přidán atribut, přejmenována třída, přejmenován atribut, smazán atribut, smazána třída.

Po každé operaci se testuje, zdali operace byla provedena úspěšně (přibylo co mělo, zmizelo, co mělo).

Test pokrývá veškeré metody, které modifikují model.
Test je úspěšný.

Testy druhého prototypu

V této části vývoje jsme se oprostili od myšlenky vyvíjet aplikaci v programovacím jazyku Java a přistoupili jsme na transformační jazyk. Jako transformační jazyk jsme zvolili QVT Operational, který je součástí EMF (Eclipse modeling framework) ve vývojovém prostředí Eclipse.

Pro testování našich transformací jsme zvolili následující postup. K provádění transformace a k jejímu následnému testování máme k dispozici několik modelů. Prvním z modelů je vstupní model. Vstupní model vyjadřuje počáteční stav, z kterého vycházíme při inicializaci transformací. Druhým modelem je výstupní model. Výstupní model reprezentuje model, který chceme dostat po provedení všech transformací. Dále máme k dispozici soupis změn, které popisují cestu, jak se ze vstupního modelu vytváří výstupní model. Z těchto souborů se dají vytvořit testovací data, která budeme předkládat testované transformaci.

Jednotkové testování (Unit-testing) provádíme tím způsobem, že testujeme zvlášť každý mapping, query i helper, zda dělají to, co od nich očekáváme.

Testování u zákazníka

Před předáním produktu zákazníkovi je plánováno testování v produkčním prostředí.