Automatyczne generowanie kodu

Marek.Berkan@e-point.pl

4Developers, 26 marca 2010

Zakres wykładu

- **O czym zamierzam opowiedzieć:**
  - **Przyspieszenie tworzenia aplikacji**
  - **Ułatwienie utrzymania aplikacji**
  - **Budowanie kontraktów pomiędzy developerami a innymi uczestnikami projektu**
  - **Wsparcie kontraktów współpracy z systemami zewnętrznymi**
- **O czym nie będę mówił:**
  - **Generatory aplikacji na podstawie bazy danych**
  - **Edytory aplikacji WYSIWYG/CASE**
  - **Czarodzieje (*ang. wizards*)/szablony z IDE**

Wstęp - dla kogo to jest?

- **Duża ilość powtarzalnego kodu**
- **Duża ilość osób w zespole**
- **Systemy przewidziane do wieloletniego rozwoju**

Przykład: kod pobierający dane z bazy tradycyjnie

```
1  String query = "SELECT ... , adres, ... FROM klient WHERE id = 1";
2
3  [...]
4
5  String adres = rs.getString("adres");
```

- **Refaktoring: pole `adres` ma zostać zastąpione dwoma: `adres_linia_1` i `adres_linia_2`**
- **Metoda: cierpliwy „search and replace”**
- **Brak wsparcia ze strony środowiska programistycznego**

Kod pobierający dane z bazy tradycyjnie - zagrożenia

- **Niekompletność zmian**

```
1  q = "SELECT ... , adr" +
2      "es, ... FROM klient WHERE id = 1";
3
4  [...]
5
6  String adres = rs.getString("adres");
```

- **Nadmiarowość zmian**
- **Odroczony czas spostrzeżenia błędu do momentu uruchomienia**

Nie wszystkie części aplikacji są tak samo testowane

Przykład: pobieranie danych z bazy wygenerowanym O/R mappingiem

```
1 Klient k = klientDAO.getByPK(1);
2 String adres = k.getAdres();
```

- **DAO (*ang. Data Access Object*) - obiekty dostępu do bazy danych**
- **POJO (*ang. Plain Old Java Object*) - reprezentują rekordy**

Pobieranie danych z bazy wygenerowanym O/R mappingiem - zyski

1. **Błędy sygnalizowane natychmiast przez kompilator oraz IDE**

```
10
11      Klient klient = klientDAO.getByPK(1);
12      String adres = klient.getAdres();
13
```

2. **Kompletność refaktoringów**

Tasks | Debug | Variables | Breakpoints | Display | Search | History | Problems | Properties | Console | Error Log

Problems (1 error, 8 135 warnings, 0 others (Filter matched 100 of 8136 items))

Description	Resource	Path	Location	Type
The method getAdres() is undefined for the ty	Test.java	/war-country/src/ja	line 12	Java Problem
a is not correctly closed proposed line for clos	edit.vt	/war-country/src/we	line 33	Problem
a is not correctly closed proposed line for clos	edit.vt	/war-country/src/we	line 37	Problem

3. **Ergonomia pracy**

```
rySubsystem(data).getCareOfEditForOneTimeAddr();
```

getCareOfEditForOneTimeAddr() : Boolean - Co
getCustomerRegistrationByIno() : Boolean - Co
getCustomerRegistrationBySelf() : Boolean - Co
getDeliveryAddressIsEditable() : Boolean - Cou

Czego użyć do generacji kodu?

- **Wybór gotowej biblioteki:**
  - **popularna (użytkownicy, wsparcie, forum, listy dyskusyjne)**
  - **dostosowane do systemu budowania aplikacji**
  - **na bieżąco poprawiana i rozwijana**
- **Samodzielna implementacja:**
  - **dostosowana do potrzeb**
  - **wymaga zasobów na implementacje i utrzymanie**

Podstawowe zasady dotyczące implementacji

1. **Wygenerowanego kodu nie umieszczamy w repozytorium**
2. **Wygenerowanego kodu nie edytujemy ręcznie**

Obszary zastosowań

- **Model bazy danych**
- **Model wymiany danych przez WebService**
- **Pliki konfiguracyjne**
- **Klucze plików lokalizacyjnych**

Baza danych (O/R-mapping)

- **Większość aplikacji biznesowych korzysta z bazy danych**
- **Dużo tabel i kolumn - ręczne tworzenie kodu jest czasochłonne**
- **Baza danych rozwijanej aplikacji podlega częstym zmianom**
- **Podział ról: projektant bazy i developer**
- **Krytyczna i podatna na drobne błędy**

O/R Mapping - Hibernate

http://www.hibernate.org/

```
1  Session session = HibernateUtil.getSessionFactory().getCurrentSession();
2
3  Klient k = (Klient) session.get(Klient.class, 1);
4  String adres = k.getAdres();
5
6  String q = " FROM " + Klient.class.getName() + " AS klient " +
7      " WHERE klient.adres = :adres AND klient.aktywny = : aktywny ";
8  List<Klient> result = (List<Klient>)session.createQuery(q).
9      setString("adres", "Marszalkowska").
10     setBoolean("aktywny", true).list();
```

- **Plusy:**
  - **Popularność, doskonałe wsparcie**
- **Minusy:**
  - **Niedoskonałości biblioteki generującej kod**
  - **Brak stałych do nazw tabel i kolumn**

O/R Mapping - Torque

http://db.apache.org/torque/

```
1  Klient k = KlientPeer.retrieveByPK(1);
2  String adres = k.getAdres();
3
4  Criteria crit = new Criteria();
5  crit.add(KlientPeer.ADRES, "Marszalkowska").and(KlientPeer.AKTYWNY, true);
6  List<Klient> result = KlientPeer.doSelect(crit);
```

- **Plusy:**
  - **Generowanie stałych do nazw tabel i kolumn**
- **Minusy:**
  - **Stary projekt**
  - **Słabe wsparcie**

XML - JAXB (*ang. Java Architecture for XML Binding*)

http://jaxb.dev.java.net/

```
1  <xsd:element name="klienci" type="t:klienci_type" />
2
3  <xsd:complexType name="klienci_type">
4      <xsd:sequence>
5          <xsd:element name="klient" type="t:klient_type"
6              minOccurs="0" maxOccurs="unbounded" />
7      </xsd:sequence>
8  </xsd:complexType>
9
10 <xsd:complexType name="klient_type">
11     <xsd:attribute name="id" type="t:positive_integer" use="required" />
12     <xsd:attribute name="imie" type="t:text255" use="required" />
13     <xsd:attribute name="nazwisko" type="t:text255" use="required" />
14     <xsd:attribute name="status" type="t:statusKlienta" use="required" />
15     <xsd:attribute name="adres" type="t:text255" use="required" />
16     <xsd:attribute name="birthdate" type="t:dateYYYYMMDD" use="required" />
17 </xsd:complexType>
```

XML - JAXB - wygenerowany kod

```
1  KlientType klient1 = factory.createKlientType();
2  [...]
3  klient1.setStatus(StatusKlientaEnum.T);
4  klient1.setAdres("Marszałkowska");
5
6  KlientType klient2 = factory.createKlientType();
7  [...]
8  klient2.setStatus(StatusKlientaEnum.N);
9  klient2.setAdres("Al. Niepodległości");
10
11 Klienci klienci = factory.createKlienci();
12 klienci.getKlient().add(klient1);
13 klienci.getKlient().add(klient2);
14
15 FileOutputStream out = new FileOutputStream(new File("/tmp/klienci.xml"));
16 marshaller.marshal(klienci, out);
```

XML - JAXB - wygenerowany plik XML

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <klienci>
3      <klient adres="Marszałkowska" birthdate="20090730" id="1"
4        imie="Marek" nazwisko="Abacki" status="T"/>
5      <klient adres="Al. Niepodległości" birthdate="20090730" id="2"
6        imie="Krzysztof" nazwisko="Babacki" status="N"/>
7  </klienci>
```

- **Plusy:**
  - **Zgodność pliku wynikowego ze schematem (m.in. formaty)**
  - **Generowanie stałych do nazw typów i atrybutów**
  - **Stałe z dozwolonej przestrzeni**

Pliki konfiguracyjne - definicja

Część przykładowego pliku konfiguracyjnego z Apache Struts `struts-config.xml`:

```
1 <form-bean name="klientForm" type="org.apache.struts.action.DynaActionForm">
2     <form-property name="imie"     type="java.lang.String" />
3     <form-property name="nazwisko" type="java.lang.String" />
4     <form-property name="adres"    type="java.lang.String" />
5     <form-property name="aktywny"  type="java.lang.Boolean" />
6 </form-bean>
```

Wygenerowany interfejs ze stałymi `klientFormC.java`:

```
1 package struts;
2
3 public interface klientFormC {
4     String FORM_NAME = "klientForm";
5     String imie = "imie";
6     String nazwisko = "nazwisko";
7     String adres = "adres";
8     String aktywny = "aktywny";
9 }
```

Pliki konfiguracyjne - kod

Standardowy sposób odwołań do pól formularza:

```
1  Klient klient = new Klient();
2  klient.setImie(form.getString("imie"));
3  klient.setNazwisko(form.getString("nazwisko"));
4  klient.setAdres(form.getString("adres"));
5  klient.setAktywny((Boolean)form.get("aktywny"));
```

Odwołanie do pól formularza z użyciem wygenerowanych stałych:

```
1  import struts.klientFormC;
2  [...]
3
4  Klient klient = new Klient();
5  klient.setImie(form.getString(klientFormC.imie));
6  klient.setNazwisko(form.getString(klientFormC.nazwisko));
7  klient.setAdres(form.getString(klientFormC.adres));
8  klient.setAktywny((Boolean)form.get(klientFormC.aktywny));
```

Pliki lokalizacyjne - klucze

Część przykładowego pliku lokalizacyjnego `validations.properties`:

```
1 klient_form.adres.required=Pole "Adres" jest wymagane
2 klient_form.adres.maxlength=Pole "Adres" nie może zawierać więcej niż {0} znaków.
```

Wygenerowany interfejs ze stałymi `validationsC.java`:

```
1 package i18n;
2
3 public interface validationsC {
4     String BUNDLE_NAME = "validations";
5     String klient_form_adres_required = "klient_form.adres.required";
6     String klient_form_adres_maxlength = "klient_form.adres.maxlength";
7 }
```

Pliki lokalizacyjne - kod

Standardowy sposób odwołań do kluczy lokalizacyjnych:

```
1  String adres = form.getString("adres");
2  if("".equals(adres.trim())) {
3      errors.add("validations", "klient_form.adres.required");
4  }
5  if(adres.trim().length() > 255) {
6      errors.add("validations", "klient_form.adres.maxlength", 255);
7  }
```

Odwołanie do kluczy lokalizacyjnych z użyciem wygenerowanych stałych:

```
1  import i18n.validationsC;
2  [...]
3
4  String adres = form.getString(klientFormC.adres);
5  if("".equals(adres.trim())) {
6      errors.add(validationsC.BUNDLE_NAME, validationsC.klient_form_adres_required);
7  }
8  if(adres.trim().length() > 255) {
9      errors.add(validationsC.BUNDLE_NAME, validationsC.klient_form_adres_maxlength, 255);
10 }
```

Podsumowanie - korzyści

- **Nie trzeba ręcznie pisać oczywistego i powtarzalnego kodu**
- **Spójność kodu i powtarzalność wzorców**
- **Błędy znajdowane na etapie kompilacji**
- **Wykorzystanie auto-uzupełniania w IDE**
- **Łatwiejsze refaktoringi**
- **Łatwiejsze znajdowanie wielu odniesień do tego samego elementu (czytanie kodu)**

Podsumowanie - utrudnienia

- **Uzależnienie od dodatkowych bibliotek zewnętrznych**
- **Trudniejsze wprowadzenie nowego developera do projektu**
- **Bardziej skomplikowany proces budowania aplikacji**
- **Konieczność tworzenia własnych generatorów do specyficznych elementów**

Czy są jakieś pytania?

Zapraszam do dyskusji