Paweł Mantur

Implementacja schematów i statycznej kontroli typów w systemie JLoXiM

projekt rozwiązania

2009-11-12

Schematy w bazach danych

- Relacyjne bazy danych:
  tabele, kolumny, typy danych, klucze itp.

Schematy w bazach danych

- Relacyjne bazy danych:
  tabele, kolumny, typy danych, klucze itp.
- XML:
  DTD, XMLSchema

Schematy w bazach danych

- Relacyjne bazy danych:
  tabele, kolumny, typy danych, klucze itp.
- XML:
  DTD, XMLSchema
- Obiektowe bazy danych:
  OMG IDL, ODMG ODL

Rola schematów

- Definiowanie zawartości bazy danych

Rola schematów

- Definiowanie zawartości bazy danych
- Opisywanie zawartości

Rola schematów

- Definiowanie zawartości bazy danych
- Opisywanie zawartości
- Więzy integralności

Rola schematów

- Definiowanie zawartości bazy danych
- Opisywanie zawartości
- Więzy integralności
- Użyteczne przy optymalizacji

Rola schematów

- Definiowanie zawartości bazy danych
- Opisywanie zawartości
- Więzy integralności
- Użyteczne przy optymalizacji
- Umożliwiają kontrolę typów

Komponenty

- System typów

Komponenty

- System typów
- Schemat zewnętrzny:
  język do definiowania schematów

Komponenty

- System typów
- Schemat zewnętrzny:
  język do definiowania schematów
- Reprezentacja wewnętrzna:
  metabaza

Komponenty

- System typów
- Schemat zewnętrzny:
  język do definiowania schematów
- Reprezentacja wewnętrzna:
  metabaza
- Store + schema

Komponenty

- System typów
- Schemat zewnętrzny:
  język do definiowania schematów
- Reprezentacja wewnętrzna:
  metabaza
- Store + schema
- Sygnatury

Komponenty

- System typów
- Schemat zewnętrzny:
  język do definiowania schematów
- Reprezentacja wewnętrzna:
  metabaza
- Store + schema
- Sygnatury
- Aparat statycznej kontroli typów

Komponenty

- System typów
- Schemat zewnętrzny:
  język do definiowania schematów
- Reprezentacja wewnętrzna:
  metabaza
- Store + schema
- Sygnatury
- Aparat statycznej kontroli typów
- Schema code writer

System typów

Typy proste atomowe:

- string
- boolean
- int, decimal, real
- date, time, datetime
- binary
- object

System typów

Restrykcje typów prostych

```
create constraint PositiveInteger restricts int {
  minValue: 0;
  defaultValue: 1;
}
```

System typów

Restrykcje typów prostych

```
create constraint PositiveInteger restricts int {
  minValue: 0;
  defaultValue: 1;
}
```

- restrykcje tworzą nowe typy atomowe

System typów

Restrykcje typów prostych

```
create constraint PositiveInteger restricts int {
  minValue: 0;
  defaultValue: 1;
}
```

- restrykcje tworzą nowe typy atomowe
- opcje zależą od typu bazowego, który ograniczamy

System typów

Restrykcje typów prostych

```
create constraint PositiveInteger restricts int {
  minValue: 0;
  defaultValue: 1;
}
```

- restrykcje tworzą nowe typy atomowe
- opcje zależą od typu bazowego, który ograniczamy
- np. dla typu string: length, regex, maxLength...

System typów

Klasy

- pola

System typów

Klasy

- pola
- metody

System typów

Klasy

- pola
- metody
- wielodziedziczenie

System typów

Klasy

- pola
- metody
- wielodziedziczenie
- role

System typów

Klasy

- pola
- metody
- wielodziedziczenie
- role
- nazwa obiektów będących instancj

System typów

Klasy

- pola
- metody
- wielodziedziczenie
- role
- nazwa obiektów będących instancją
- lista eksportowa

System typów

Klasy

- pola
- metody
- wielodziedziczenie
- role
- nazwa obiektów będących instancją
- lista eksportowa
- więzy integralności

System typów

Liczebności

- [0..1] albo ? - wartość opcjonalna

System typów

Liczebności

- [0..1] albo ? - wartość opcjonalna
- [1..1] - wartość wymagana (domyślne)

System typów

Liczebności

- [0..1] albo ? - wartość opcjonalna
- [1..1] - wartość wymagana (domyślne)
- [0..*] - kolekcja

System typów

Liczebności

- [0..1] albo ? - wartość opcjonalna
- [1..1] - wartość wymagana (domyślne)
- [0..*] - kolekcja
- [1..*] - niepusta kolekcja

System typów

Predefiniowane typy złożone

xml

- wymuszanie struktury XML
- wsparcie dla DOM, XSLT, Xquery

System typów

Predefiniowane typy złożone

xml

- wymuszanie struktury XML
- wsparcie dla DOM, XSLT, Xquery

any – luźna struktuta

- brak narzyconej struktury
- schemat dynamiczny

System typów

Enumeratory

```
create enum AtomicType {
  int, string, bool, datetime, binary = 7
}
```

- wewnętrznie jest to int
- domyślna numeracja od 0

System typów

Constraints (więzy integralności)

- Wewnątrz definicji klasy

```
constraint cardinalityConstraint
"minOccurences <= maxOccurences"
on maxOccurences, minOccurences;
```

System typów

Constraints (więzy integralności)

- Wewnątrz definicji klasy

```
constraint cardinalityConstraint
"minOccurences <= maxOccurences"
on maxOccurences, minOccurences;
```

- Poza definicją klasy

```
constraint cardinalityConstraint
restricts FieldType
"minOccurences <= maxOccurences"
on maxOccurences, minOccurences;
```

Schemat zewnętrzny

- Nieoficjalana nazwa JLoXiM ODL

Schemat zewnętrzny

- Nieoficjalna nazwa JLoXiM ODL
- Rozszerzenie istniejącej gramatyki

Schemat zewnętrzny

- Nieoficjalna nazwa JLoXiM ODL
- Rozszerzenie istniejącej gramatyki
- Tworzenie/usuwanie klas, enumeratorów, restrykcji, więzów

Schemat zewnętrzny

- Nieoficjalna nazwa JLoXiM ODL
- Rozszerzenie istniejącej gramatyki
- Tworzenie/usuwanie klas, enumeratorów, restrykcji, więzów
- Gtramatyka dostępna na wiki: http://jloxim.mimuw.edu.pl/redmine/wiki/jloxim/SchemaGrammar

Metabaza

- Część składu obiektów, gdzie pamiętamy definicje wprowadzone za pomocą schematu zewnętrznego

Metabaza

- Część składu obiektów, gdzie pamiętamy definicje wprowadzone za pomocą schematu zewnętrznego
- Zdefiniowanie schematu dla składu obiektów poprzez określenie typu (zazwyczaj klasy) dla korzenia

Metabaza

- Część składu obiektów, gdzie pamiętamy definicje wprowadzone za pomocą schematu zewnętrznego
- Zdefiniowanie schematu dla składu obiektów poprzez określenie typu (zazwyczaj klasy) dla korzenia
- Możliwość automatycznego dodania przestrzeni systemowej (metabazy) do schematu zewnętrznego

Organizacja metabazy

- Odpytywanie metabazy:

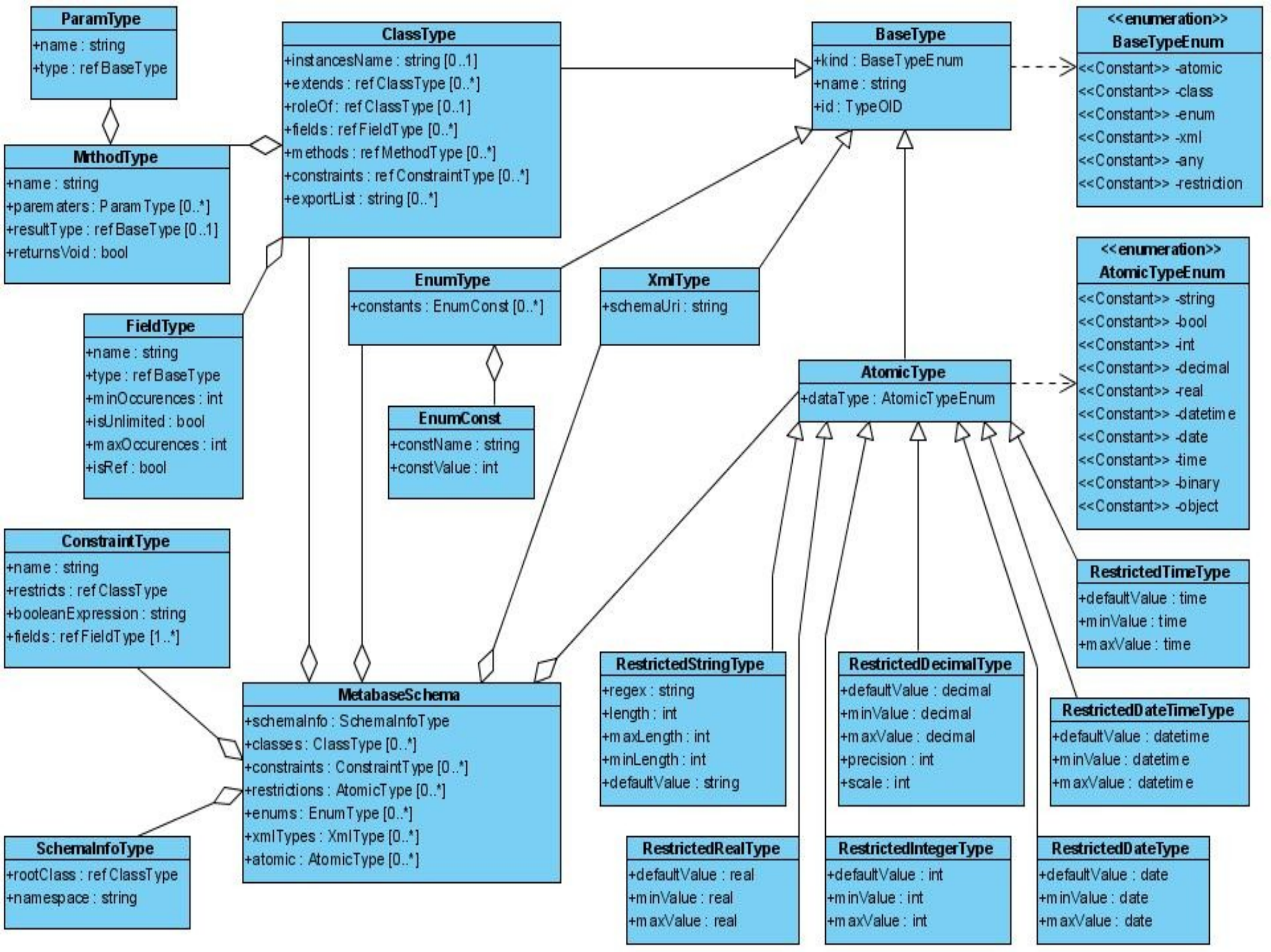
```
sys.classes where name = „MojaKlasa”
sys.schemaInfo.rootClass.name
```

Organizacja metabazy

- Odpytywanie metabazy:

```
sys.classes where name = „MojaKlasa”
sys.schemaInfo.rootClass.name
```

- Schemat metabazy w postaci diagramu UML...


ParamType
+name : string
+type : ref BaseType
ClassType
+instancesName : string [0..1]
+extends : ref ClassType [0..*]
+roleOf : ref ClassType [0..1]
+fields : ref FieldType [0..*]
+methods : ref MethodType [0..*]
+constraints : ref ConstraintType [0..*]
+exportList : string [0..*]
BaseType
+kind : BaseTypeEnum
+name : string
+id : TypeOID
<<enumeration>>
BaseTypeEnum
<<Constant>> -atomic
<<Constant>> -class
<<Constant>> -enum
<<Constant>> -xml
<<Constant>> -any
<<Constant>> -restriction
MrthodType
+name : string
+parematers : ParamType [0..*]
+resultType : ref BaseType [0..1]
+returnsVoid : bool
EnumType
+constants : EnumConst [0..*]
XmlType
+schemaUri : string
FieldType
+name : string
+type : ref BaseType
+minOccurences : int
+isUnlimited : bool
+maxOccurences : int
+isRef : bool
EnumConst
+constName : string
+constValue : int
<<enumeration>>
AtomicTypeEnum
<<Constant>> -string
<<Constant>> -bool
<<Constant>> -int
<<Constant>> -decimal
<<Constant>> -real
<<Constant>> -datetime
<<Constant>> -date
<<Constant>> -time
<<Constant>> -binary
<<Constant>> -object
AtomicType
+dataType : AtomicTypeEnum
ConstraintType
+name : string
+restricts : ref ClassType
+booleanExpression : string
+fields : ref FieldType [1..*]
MetabaseSchema
+schemaInfo : SchemaInfoType
+classes : ClassType [0..*]
+constraints : ConstraintType [0..*]
+restrictions : AtomicType [0..*]
+enums : EnumType [0..*]
+xmlTypes : XmlType [0..*]
+atomic : AtomicType [0..*]
RestrictedTimeType
+defaultValue : time
+minValue : time
+maxValue : time
RestrictedStringType
+regex : string
+length : int
+maxLength : int
+minLength : int
+defaultValue : string
RestrictedDecimalType
+defaultValue : decimal
+minValue : decimal
+maxValue : decimal
+precision : int
+scale : int
RestrictedDateTimeType
+defaultValue : datetime
+minValue : datetime
+maxValue : datetime
SchemaInfoType
+rootClass : ref ClassType
+namespace : string
RestrictedRealType
+defaultValue : real
+minValue : real
+maxValue : real
RestrictedIntegerType
+defaultValue : int
+minValue : int
+maxValue : int
RestrictedDateType
+defaultValue : date
+minValue : date
+maxValue : date

Organizacja metabazy

- Odpytywanie metabazy:

```
sys.classes where name = „MojaKlasa”
sys.schemaInfo.rootClass.name
```

- Schemat metabazy w postaci diagramu UML
- Schemat metabazy zapisany w JLoXiM ODL na wiki

Store + schema

- Skład obiektów ze świadomością posiadania schematu

Store + schema

- Skład obiektów ze świadomością posiadania schematu
- Kontrola operacji modyfikujących dane

Store + schema

- Skład obiektów ze świadomością posiadania schematu
- Kontrola operacji modyfikujących dane
- Odczytywanie informacji o typach z metabazy

Sygnatury

- Podczas kontroli statycznej nie znamy wartości obiektów, ale znamy ich sygnatury

Sygnatury

- Podczas kontroli statycznej nie znamy wartości obiektów, ale znamy ich sygnatury
- Sygnatury opisują wartości przetwarzane w czasie wykonania

Sygnatury

- Podczas kontroli statycznej nie znamy wartości obiektów, ale znamy ich sygnatury
- Sygnatury opisują wartości przetwarzane w czasie wykonania
- Niosą informację o typie lub możliwych typach obiektów

Sygnatury

- Podczas kontroli statycznej nie znamy wartości obiektów, ale znamy ich sygnatury
- Sygnatury opisują wartości przetwarzane w czasie wykonania
- Niosą informację o typie lub możliwych typach obiektów
- Można symulować wykonanie zapytania wykonując je na metabazie

Sygnatury

- Podczas kontroli statycznej nie znamy wartości obiektów, ale znamy ich sygnatury
- Sygnatury opisują wartości przetwarzane w czasie wykonania
- Niosą informację o typie lub możliwych typach obiektów
- Można symulować wykonanie zapytania wykonując je na metabazie
- Zamiast na rzeczywistych wartościach operujemy na sygnaturach

Statyczna kontrola typów

- Symulowanie wykonania: statyczny stos środowisk i statyczny stos rezultatów

Statyczna kontrola typów

- Symulowanie wykonania: statyczny stos środowisk i statyczny stos rezultatów
- Zamiast wartości – sygnatury

Statyczna kontrola typów

- Symulowanie wykonania: statyczny stos środowisk i statyczny stos rezultatów
- Zamiast wartości – sygnatury
- Wnioskowanie o typie wyniku na podstawie tabel decyzyjnych

Statyczna kontrola typów

- Symulowanie wykonania: statyczny stos środowisk i statyczny stos rezultatów
- Zamiast wartości – sygnatury
- Wnioskowanie o typie wyniku na podstawie tabel decyzyjnych
- Nie o wszystkim da się zdecydować podczas kontroli statycznej

Statyczna kontrola typów

- Symulowanie wykonania: statyczny stos środowisk i statyczny stos rezultatów
- Zamiast wartości – sygnatury
- Wnioskowanie o typie wyniku na podstawie tabel decyzyjnych
- Nie o wszystkim da się zdecydować podczas kontroli statycznej
- **Pół**-mocna kontrola typów

Tablice decyzyjne

- Tablica decyzyjna dla każdego operatora

Tablice decyzyjne

- Tablica decyzyjna dla każdego operatora
- Wnioskowanie o typie wyniku na podstawie typu argumentów

Tablice decyzyjne

- Tablica decyzyjna dla każdego operatora
- Wnioskowanie o typie wyniku na podstawie typu argumentów
- Czasami zbiór możliwych typów (wariant)

Tablice decyzyjne

- Tablica decyzyjna dla każdego operatora
- Wnioskowanie o typie wyniku na podstawie typu argumentów
- Czasami zbiór możliwych typów (wariant)
- Czasami odłożenie sprawdzenia poprawności typologicznej do czasu wykonania

Tablice decyzyjne

- Tablica decyzyjna dla każdego operatora
- Wnioskowanie o typie wyniku na podstawie typu argumentów
- Czasami zbiór możliwych typów (wariant)
- Czasami odłożenie sprawdzenia poprawności typologicznej do czasu wykonania
- Łatwa zmiana zachowania aparatu kontroli, poprzez zmianę tabel decyzyjnych

Tablice decyzyjne

- Tablica decyzyjna dla każdego operatora
- Wnioskowanie o typie wyniku na podstawie typu argumentów
- Czasami zbiór możliwych typów (wariant)
- Czasami odłożenie sprawdzenia poprawności typologicznej do czasu wykonania
- Łatwa zmiana zachowania aparatu kontroli, poprzez zmianę tabel decyzyjnych
- Wiele decyzji to kwestia upodobań

Schema code writer

- Eksport metabazy do postaci zewnętrznej

Schema code writer

- Eksport metabazy do postaci zewnętrznej
- Generowanie bibliotek klienckich

Schema code writer

- Eksport metabazy do postaci zewnętrznej
- Generowanie bibliotek klienckich
- Dostarczenie API umożliwiającego pisanie plug-in'ów generujących różne formaty (XML, Java, C#, UML, ...)

Schema code writer

- Eksport metabazy do postaci zewnętrznej
- Generowanie bibliotek klienckich
- Dostarczenie API umożliwiającego pisanie plug-in'ów generujących różne formaty (XML, Java, C#, UML, ...)
- Implementacja eksportu do JLoXiM ODL

Dziękuję.