Developing Editors in Xtext Framework

Bojana Dimić Surla

Abstract – *The article presents one approach in development of editors as an alternative to standard screen forms for entering data. That approach relates to specifying Xtext grammar of the text that will be entered and generating EMF model and editor on the basis of that specification. Advanced functionalities of the editor, such as generating help during editing and control of correctness of entered data are realized as extensions and constraints on generated EMF model. Such an editor was verified on the example of entering data in library information system. Two editors are shown, one for entering structured data of MARC 21 records and one for entering holdings data.*

Index terms - ***grammmar, Eclipse, EMF, Java***

1. INTRODUCTION

ONE of the most important aspects of information systems is digital content creation, and most common way of creating digital content is by entering data in screen form. In contemporary information systems data input is usually done in screen forms full of visual components, such as labels, text fields, combo boxes, etc.

In this paper we give the suggestion for development of editors that enable different way of entering data, the way that resembles to entering data in text editors such as Microsoft Word and editors for writing source code. In such editor user write data in big text area while editor provides help and control of data correctness.

Those editors are created in Xtext framework without writing source code. They are realised using model-driven software development by specification of data model as Xtext grammar and generating EMF model and editor.

Two editors are presented in this paper. Both are used for entering data in library software system BISIS.

Manuscript received December 20, 2012. The research presented in this paper was supported by Ministry of Science and Technological Development of the Republic of Serbia, Grant III III43007: “Technology Enhanced Infrastructure for eLearning in Serbia”.

Bojana Dimić Surla is with the Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Serbia (e-mail: bdimic@uns.ac.rs).

The first one is editor for entering structured data of MARC 21 bibliographic records, and second is editor for entering holdings data in database.

Those editors represent the improvement of editor in fourth version of system BISIS described in [1], [2] and [3].

2. SOFTWARE ENVIRONMENT

For development of editors described in this paper we were using Eclipse platform. Eclipse was chosen as software environment for specification of the editors and as software environment for running the editors.

Eclipse is an open source project that represents environment for software development. It enables work with various programming languages and on various platforms. Eclipse provides plug-in framework for integration of additional software tool.

EMF - Eclipse Modelling Framework [4] is environment for software modelling and code generation. EMF enables software development by generating source code based on structured data model [5]. EMF model can be built on the basis of UML models, XML schema and set of Java classes [6]. Moreover, EMF represents the support for model-driven software development and is the base for other project in this area such as OpenArchitectureWare (oAW) [7].

OAW is a platform for model-driven software development that provides parsing of models and family of languages and generating source code on the basis of that. Basic characteristics of this platform are that it is developed as a set of plug-ins for Eclipse, it is based on EMF meta-modelling concept and all transformations and generations are specified as workflows. Description of oAW usage is given in [8].

The part of oAW is software tool Xtext for creation of domain-specific languages and generation of Eclipse text editor for those languages. Furthermore, oAW platform provides language Xtend for writing extensions of domain-specific languages, language Check for specifying constraints and Xpand for writing templates for language transformations. Xtend is used for defining content assist i.e. predefined set of acceptable values for input in editor. Constraints written in Check that is similar to Object Constraints Language (OCL) are used for control of correctness of data entered in generated editor.

Xtext uses ANTLR – Another tool for language recognition [9] for implementing parser. ANTLR is a tool that provides framework for language recognition, interpretation, compiling and translation. ANTLR has excellent support for syntax tree construction.

In June 2010 the new version of Xtext 1.0 was released as a part of Eclipse. This version supports development of complex programming languages, editors for those languages as well as complete integrated development environment based on Eclipse [10].

3. Editor for MARC 21 records

A MARC 21 bibliographic record describes one publication (book, serial, article, etc.) and has exactly defined structure. MARC 21 record starts with record leader followed by fields labeled with three digits. There are two types of fields, control fields and data fields. Control field can contain either singular data about publication, either set of fixed-length data on specific position. Data field has up to two indicators and set of subfields marked with letter or digit. Subfields contain bibliographic data as their content.

```
Record:
 leader=Leader (controlFields+=ControlField)* (dataFields+=DataField)* ;
Leader:
 "LDR"  content=LeaderContentType;
LeaderContentType:
        recordLength=ContentType              // 00-04
        recordStatus=ContentType              // 05
        typeOfRecord=ContentType              // 06
        bibliographicLevel=ContentType        // 07
        typeOfControl=ContentType                                 // 08
        characterCodingScheme=ContentType                         // 09
        indicatorCount=ContentType                               // 10
        subfieldCodeCount=ContentType                            // 11
        baseAddressOfData=ContentType                            // 12-16
        encodingLevel=ContentType                                // 17
        descriptiveCatalogingForm=ContentType                    // 18
        multipartResourceRecordLevel=ContentType                 // 19
        lengthOfLengthOfFieldPortion=ContentType                 // 20 : 4
        lengthOfStartingCharacterPositionPortion=ContentType     // 21 : 5
        lengthOfImplementationDefinedPortion=ContentType         // 22 : 0
        undefined=ContentType;                                   // 23 : 0
ControlField: NEW_LINE  name=ControlFieldNameEnum
(characterPositions+=CharacterPosition)*;

DataField: NEW_LINE name=DataFieldNameEnum  ind1=ContentType
ind2=ContentType (subfields+=Subfield)*;
CharacterPosition:  content=ContentType;
Subfield:  name=SubfieldNameEnum  content=STRING;
Native NEW_LINE : "('\r')+";
Native ContentType: "('0'..'9'|'a'..'z'|'A'..'Z'|'#'|'|'|'['|']'|'-
'|'_'|'*'|'£')*";
Native WS: "(' '|'\t'|'\n')+ {$channel=HIDDEN;}";
Enum ControlFieldNameEnum: cf001="001" | cf003="003" | cf004="004" |
cf005="005" | cf008="008";
Enum DataFieldNameEnum: df010="010" | df013="013" | df015="015" |
df016="016" | df020="020"
      | df022="022" | df100="100"| df110="110"| df210="210"| df240="240" |
df245="245" |df246="246"
      | df260="260" |df300="300" | df500="500" | df650="650" | df852="852"
| df853="853";
Enum SubfieldNameEnum: sf0="$0" |sfa="$a" | sfb="$b" | sfc="$c" | sfd="$d"
|sff="$f" |sfg="$g" | sfh="$h"
| sfv="$v" | sfz="$z" | sf2="$2" | sf6="$6" | sf8="$8";
```

Listing 1. Xtext grammar for MARC 21 records

Existing editors for MARC 21 records that are available at Library of Congress Internet site [11] can be divided into two groups. The first group includes those editors that provide standard screen forms with labels and text fields (Concourse [12]). The second group relates to editors that support input of MARC 21 record as a free text (MarcEdit [13]). Editor described in this chapter belongs to second group, except that it supports the control of entered data which is not the case with other editors in this group. Described approach of development of editor for MARC 21 records is discussed in [14].

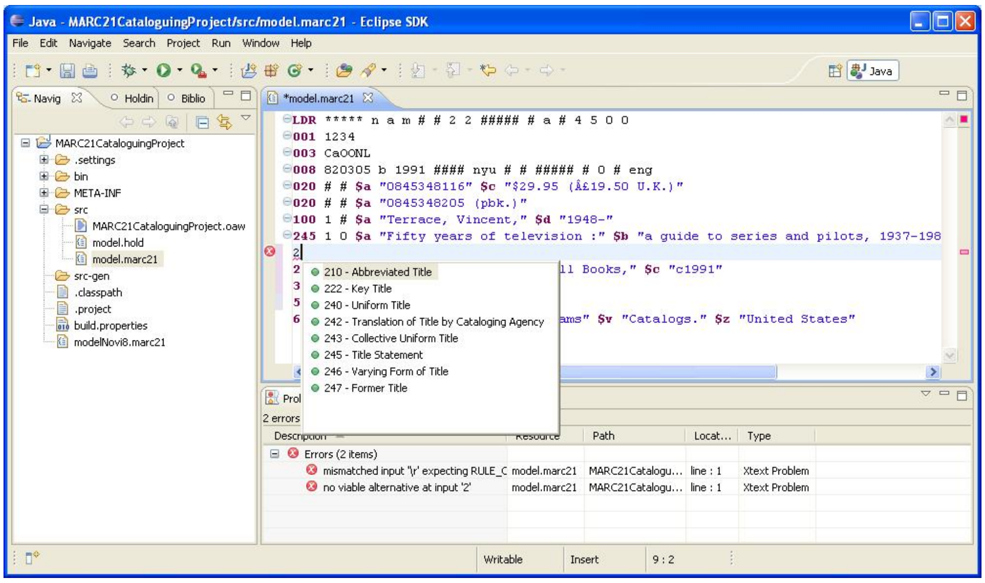
The first step in realisation of editor is specification of grammar for MARC 21 records in Xtext environment. This specification is shown in Listing 1.

Rule Record in grammar from Listing 1 describes one MARC 21 record. By this rule, MARC 21 record consists of leader represents by attribute leader of type Leader, set of control fields represents by attribute controlFields that is the list of elements of ControlField type and set of data fields represents by attribute dataFields that is the list of elements of DataField type. The order in which those elements are listed in the grammar is the order in which those elements should appear in the editor.

On the basis of the grammar shown in Listing 1 EMF model of MARC 21 record and editor for input of MARC 21 records were generated. In Xtext environment it is possible to specify constraints on generated EMF model that are used in generated editor for control of entered data. All that constraints were written in single file as an expression in Check language. One example of those constraints is number of field with the same name in the record. Some fields in MARC 21 can appear more than once in the record (so called repeatable fields), while others can appear only once. Check expression that control this condition is:

```
Context DataField ERROR
"Field"+this.name.toString().
subString(2,5)+"is not repeatable!"  :
isFieldOccurenceOK(this);
```

This expression specifies the massage that will appear in the editor in context DataField if expression *isFieldOccurenceOK* is false.



```
LDR ***** n a m # # 2 2 ##### # a # 4 5 0 0
001 1234
003 CaOONL
008 820305 b 1991 #### nyu # # ##### # 0 # eng
020 # # $a "0845348116" $c "$29.95 (Â£19.50 U.K.)"
020 # # $a "0845348205 (pbk.)"
100 1 # $a "Terrace, Vincent," $d "1948-"
245 1 0 $a "Fifty years of television :" $b "a guide to series and pilots, 1937-198
2
2                                   ll Books," $c "c1991"
3
5
6                                   ams" $v "Catalogs." $z "United States"
```

210 - Abbreviated Title
222 - Key Title
240 - Uniform Title
242 - Translation of Title by Cataloging Agency
243 - Collective Uniform Title
245 - Title Statement
246 - Varying Form of Title
247 - Former Title

2 errors

Description	Resource	Path	Locat...	Type
Errors (2 items)				
mismatched input '\r' expecting RULE_C	model.marc21	MARC21Catalogu...	line : 1	Xtext Problem
no viable alternative at input '2'	model.marc21	MARC21Catalogu...	line : 1	Xtext Problem

Figure 1. Editor za MARC 21 zapise

On generated EMF model of MARC 21 records it is possible to specify extensions that are used in the editor for content assist i.e. generate list of possible values for input on cursor position. Those extensions are written in Xtend language. One example of those extensions is the content assist for entering name of the field in the editor that is specified with the following expression:

```
List[Proposal]
completeDataField_name(emf::EObject
ctx,String prefix) :
 {newProposal("010 - Library of
Congress Control Number", "010 # #"),
  ...
  newProposal("210 - Abbreviated
Title", "210"),
  newProposal("222 - Key Title",
"222"),
  newProposal("240 - Uniform Title",
"240"),
  newProposal("242 - Translation of
Title by Cataloging Agency", "242"),
  newProposal("243 - Collective
Uniform Title", "243"),
  newProposal("245 - Title Statement",
"245"),
  newProposal("246 - Varying Form of
Title", "246"),
  newProposal("247 - Former Title",
"247")
 ...};
```

The appearance of the generated editor is shown in Figure 1. Figure 1 illustrates input of field name with list of possible values offered to user.

4. EDITOR FOR HOLDINGS DATA

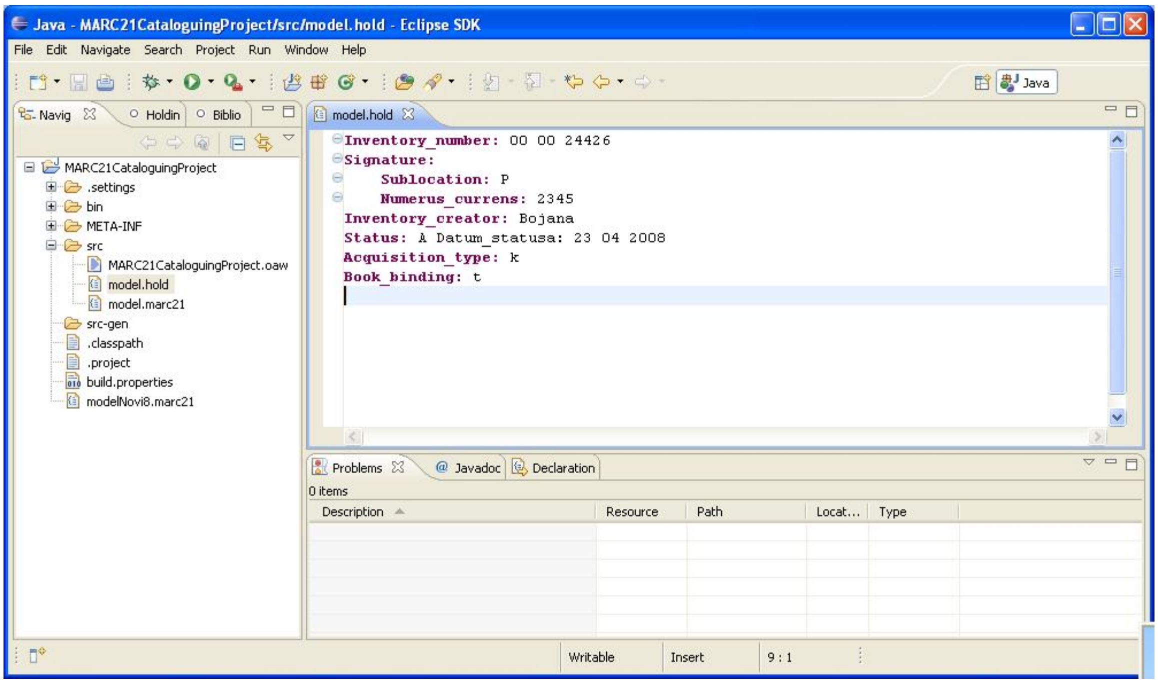
Holdings data relates to single copy of publication in the library. Every copy has its own inventory number, signature, acquisition, etc. Within BISIS library software system those pieces of information are stored in database tables. In this section we give the suggestion of creating editor in Xtext for those types of data.

Listing 2 shows grammar for entering holdings data. The rule Inventory represents data type that describes one copy of bibliographic item. Each piece of information about holdings is presented by its own rule. Elements declared with "=" are mandatory while optional elements are declared with "?=".

Rules that describe single holdings elements consists of the label which is the key word written in the quotes in Xtext grammar and data itself that is entered in the editor. That data can be of simple type such are String or Integer or can be of complex type such is Date which is also presented as a rule in the given grammar.

```
Inventory:
      inventoryNumber=InventoryNumber
      inventoryDate?=InventoryDate
      signature = Signature
      inventoryCreator=InventoryCreator
      status?=Status
      acquistionType?=AcquisitionType
      supplier?=Supplier
      availability?=Availability
      bookBinding?=BookBinding
      note?=STRING;

InventoryNumber: "Inventory_number:" department=INT invBook=INT number=INT
NEW_LINE;
InventoryCreator: "Inventory_creator:" name=STRING NEW_LINE;
InventoryDate: "Inventory_date:" date=Date NEW_LINE;


Signature: "Signature:" NEW_LINE
              sublocation?=Sublocation
              numerusCurrens?=NumerusCurrens
              internalCode?=InternalCode
              udk?=UDK
              format?=Format
              dublet?=Dublet;

Status: "Status:" sifra=ID statusDate=Date NEW_LINE;
Sublocation: "Sublocation:" code=ID NEW_LINE;
NumerusCurrens: "Numerus_currens:" number=INT NEW_LINE;
InternalCode: "Internal_code:" code=ID NEW_LINE;
UDK: "UDK:" number=STRING NEW_LINE;
Format: "Format:" code=ID NEW_LINE;
Dublet: "Dublet:" dublet=ID NEW_LINE;

AcquisitionType: "Acquisition_type:" code=ID NEW_LINE;
Supplier: "Supplier:" name=STRING NEW_LINE;
Availability: "Availability:" code=ID NEW_LINE;
BookBinding: "Book_binding:" code=ID NEW_LINE;

Native NEW_LINE : "('\r'|'\n')+";
Native WS: "(' '|'\t')+ {$channel=HIDDEN;}";
Date: year=INT month=INT day=INT;
```

Listing 2. Grammar of the language for input of holdings data



```
Inventory_number: 00 00 24426
Signature:
    Sublocation: P
    Numerus_currens: 2345
Inventory_creator: Bojana
Status: A Datum_statusa: 23 04 2008
Acquisition_type: k
Book_binding: t
```

Figure 2. Editor for holdings data

On the basis of the grammar shown in Listing 2 EMF model of holdings data and editor for those data were generated. On the generated EMF model we specified constraints by which editor controls the correctness of entered data and shows appropriate message. One example of those constraints is check if entered codes belong to the set of acceptable codes for coded holdings elements, such as status, acquisition type or sub location. Check expression that controls the correctness of code for acquisition type is:

```
   context AcquisitionType ERROR
"Unknown code for acquisition type!" :
acquisitionTypeCodes().contains(this.sifra);
   List[String] acquisitionTypeCodes():
{"a","b","c","d","f"};
```

The expression given above specify the text of an error message that will show in the editor if user enter acquisition type that is not in the set of defined code for this element.

Like it was for the MARC 21 record editor it is possible to define content assist for holdings data model. One example of this is a list of all codes defined for coded element.

Figure 2 shows editor for holdings data generated on the basis of the grammar from Listing 2. Labels that are presented as key words in the grammar are highlighted. Next to the label there is an entered data that relates to one copy of a publication.

5. Conclusion

Editors described in this paper are realised as Eclipse plug-ins. Those editors can be extended by advanced user interface functionalities in Eclipse plug-in technology using SWT and JFace packages. Moreover, those editors were built as independent application by using Rich Client Platform technology.

The advantage of suggested solution for building screen forms for entering data is that they are generated by the grammar of the text instead of writing source code. In addition, some advanced functionalities such as content assist and content control are realised on the basis of specification without writing source code.

Another advantage of suggested solution is the way of editing. Entering data in the single text area is closer to the users that use text editors such as Microsoft Word or editors for programming. Nevertheless, going through the entered data as well as copy-paste methods are facilitated in editors described in this paper.

For structured data such as MARC 21 records this solution for editing naturally arises, but this paper showed that it can be used for data that does not have structured form and are stored in database tables.

References

[1] Dimić, B , “XML editor za obradu bibliografske građe”, master theses, Faculty of Sciences, Novi Sad, available at: http://diglib.uns.ac.rs/ndltd/docs/set3/ndltd559/tezaBojanaDimic.doc (accessed 30 November 2010)

[2] Dimić, B. and Surla, D., “XML Editor for UNIMARC and MARC21 cataloguing”, The Electronic Library, 2009, Vol. 27., No 3, pp. 509-28

[3] Dimić, B., Milosavljević, B., and Surla, D., XML schema for UNIMARC and MARC 21 formats. The Electronic Library, 2010, Vol. 28 , No.2, pp. 245-262.

[4] Eclipse Modeling – EMF Home,available at: http://www.eclipse.org/modeling/emf/ (accessed 5 December 2010)

[5] Budinsky, F., Steinberg, D., Merks, E., Ellersick, R and Grose, T., “ Eclipse Modeling Framework: A Developer's Guide“, 2003, Addison Wesley, United States

[6] Powell, A., „Model with the Eclipse Modeling Framework, Part 1: Create UML models and generate code“, 2004, IBM, available at: http://www.ibm.com/developerworks/library/os-ecemf1 (accessed 5 December 2010)

[7] openArchitectureWare.org - Official openArchitectureWare Homepage, available at: http://www.openarchitectureware.org/ (accessed 5 December 2010)

[8] Efftinge, S., Friese, P., Haase, A., Hübner, D., Kadura, C., Kolb, B, Köhnlein, J., Moroff. D., Thoms, K., Völter, M., Schönbach. P., Eysholdt, M., Hübner. D. and Reinisch, S., „openArchitectureWare User Guide, Version 4.3.1“, 2008, openArchitectureWare, availanle at: http://www.openarchitectureware.org/pub/documentation/4.3.1/openArchitectureWare-4.3.1-Reference.pdf (accessed 5 Decmber 2010)

[9] ANTLR Parser Generator, available at: http://www.antlr.org/ (accessed 2 December 2010)

[10] Xtext, available at: http://www.eclipse.org/Xtext/ (accessed 29 November 2010)

[11] MARC Standards, available at: http://www.loc.gov/marc/ (accessed 5 December 2010)

[12] Concourse Software Product, available at: http://www.booksys.com/v2/products/concourse/ (accessed 5 December 2010)

[13] MarcEdit, available at: http://people.oregonstate.edu/~reeset/marcedit/html/index.php
(accessed 28 November 2010)

[14] Dimić Surla, B (2010), “Softverski sistem za katalogizaciju po MARC 21 formatu”, PhD theses, Faculty of Sciences, Novi Sad, available at: http://diglib.uns.ac.rs/ndltd/docs/set4/ndltd608/Disertacija-BojanaDimicSurla.pdf (accessed 9 December 2010)