EOL Specification

11/08/2002 10:09

A specification of the Exteca Ontology Language

Version History

9 April 2002 First Draft
11 Aug 2002 Brought up to date with current implementation

Authors

Llewelyn Fernandes
Mauro Talevi
Neetu Jain

Contents

Contents....2
Introduction....3
BNF Specification....4
Terminology....4
Core EOL....4
Ontology....4
Metadata....4
MetaDataAttribute....5
Imports....6
Import....6
Associations....6
Association....6
SubassociationOf....6
Attributes....7
Attribute....7
Concepts....7
Concept....7
SignpostOf....7
ConceptDefinition....7
SubclassOf....8
InstanceOf....8
ConceptContraint....8
ExistsLink....8
ForallLinks....9
Property....9
Link....9
Others....9
Content Extensions for EOL....10
Ontology....10
ConceptContent....10
Resources....11
Resource....11
Rules....11
Rule....11
RuleLogic....11
ResourceUrl....11
ContentId....11

Introduction

This document provides a full specification of the Exteca Ontology Language. For examples and a more tutorial-based approach please see the EOL Guide.

BNF Specification

Terminology

This specification uses the following notation

S ::= D	Symbol S defined by definition D
(E)*	repeat expression E zero or more times
(E)+	repeat expression E one or more times
(E)?	optional (0 or 1)
{A|B}	either A or B
bold	signifies a non-terminal symbol
normal	signifies a terminal string

Core EOL

The Core EOL provides facilities for modelling the semantics of the domain that the ontology represents.

Ontology

Defines an ontology. The id is a short identifier for the ontology. The URI is a unique identifier for the ontology.

Ontology ::=
<ontology id=**OntologyId** uri=**OntologyUri** version**=Integer**>
Metadata
Imports
Associations
Attributes
Concepts
</ontology>

Metadata

Defined information about the ontology itself. This is based on the Dublin Core metadata set.

Metadata ::=
<metadata [**MetaDataAttribute**=**String**]*>
(**MetaDataTitle**)?
(**MetaDataCreator**)?
(**MetaDataSubject**)?
(**MetaDataDescription**)?
(**MetaDataPublisher**)?
(**MetaDataContributor**)?
(**MetaDataDate**)?
(**MetaDataType**)?
(**MetaDataFormat**)?
(**MetaDataIdentifier**)?
(**MetaDataSource**)?
(**MetaDataLanguage**)?
(**MetaDataRelation**)?
(**MetaDataCoverage**)?
(**MetaDataRights**)?
(**MetaDataModificationHistory**)?
</metadata>

MetaDataTitle ::=

```
        <metadata-title>String</metadata-title>

MetaDataCreator ::=
        <metadata-creator>String</metadata-creator>

MetaDataDescription ::=
        <metadata-description>String</metadata-description>

MetaDataSubject ::=
        <metadata-description>String</metadata-description >

MetaDataPublisher::=
        <metadata-publisher>String</metadata-publisher>

MetaDataContributor::=
        <metadata-contributor>String</metadata-contributor>

MetaDataDate::=
        <metadata-date>String</metadata-date>

MetaDataType::=
        <metadata-type>String</metadata-type>

MetaDataFormat::=
        <metadata-format>String</metadata-format>

MetaDataIdentifier::=
        <metadata-identifier>String</metadata-identifier>

MetaDataSource::=
        <metadata-source>String</metadata-source>

MetaDataLanguage::=
        <metadata-language>String</metadata-language>

MetaDataRelation::=
        <metadata-relation>String</metadata-relation>

MetaDataCoverage::=
        <metadata-coverage>String</metadata-coverage>

MetaDataRights::=
        <metadata-rights>String</metadata-rights>

MetaDataModificationHistory ::=
        (MetaDataModification)*
[Not yet implemented]

Modification ::=
        <metadata-modification>
                <date>String</date>
                <modifier>String</modifier>
        </metadata-modification>
[Not yet implemented]
```

MetaDataAttribute

Allows the specification of metadata as attributes rather than elements. For example

```
<metadata title="food" description="food ontology"/>
```

Imports

[Not currently implemented]

A collection of Import elements.

Imports ::=
Import*

Import

[Not currently implemented]

Allows us to inherit an existing ontology and extend it. The uri corresponds to the uri in an existing ontology. The prefix is the XML namespace prefix used to refer to elements of the imported ontology in the ontology being constructued.

Import ::=
<import uri=**OntologyUri** prefix**=String/**>

Associations

A collection of Association elements.

Associations ::=
Association*

Association

Defines a type of binary link (relation). The ids for the association and the inverse are the names of links (relations). Restrictions on the domain, range and cardinality can be provided here but are global to the ontology (although they can be restricted further at the concept level). The domain and range refer to concepts and their subconcepts.

Association ::=
<association id=**AssociationId** inverse-id=**AssociationId>**
(**Description)**?
(**Comment)**?
(**SubassociationOf)***
(<mapping domain=**ConceptId** range=**ValueExp/>)**?
(<cardinality count=**CardinalityExp** inverse-count=**CardinalityExp**>)?
</association>

EOL predefines one association. The *hierarchy* association is used to group together associations that should be used when building naviagable hierarchies. You can create a subassociation of hierarchy in order to make an association hierarchical.

SubassociationOf

Allows us to further restrict an existing association. All concepts linked by the subassociation are assumed to be linked by the superassociation.

SubassociationOf ::=
<subassociation-of id=**AssociationId/**>

Note that currently only one level of inheritance is supported.

Attributes

A collection of Attribute elements.

Attributes ::=
Attribute*

Attribute

Defines a type of property. Restrictions on the domain and range can be provided here but are global to the ontology (although they can be restricted further at the concept level). The domain refers to a concept and its subconcepts. Note that each concept can only have a single attribute of a particular id (assumed cardinality of one).

Attribute ::=
<attribute id=**AttributeId** data-type=**DataTypeId**>
(**Description)**?
(**Comment)**?
(<mapping domain=**ConceptId** range=**ValueExp/>)**?
</attribute>

Concepts

A collection of Concept elements.

Concepts ::=
Concept*

Concept

Defines a concept, representing a thing in the ontology. Concepts may be classes (subsets) or instances (members) of a class.

Concept ::=
<concept id=**ConceptId** [name=**String**]**>**
(**Description)**?
(Comment)?
{SignpostOf | ConceptDefinition}
</concept>

SignpostOf

Defines a signpost to another concept. Signposts are non-preferred synomyms pointing to the concept that has the preferred id and contains the definition of the concept.

SignpostOf ::=
<signpost-of concept=**ConceptId**>

ConceptDefinition

Defines the properties and relationships to other concepts and any restrictions.

ConceptDefinition ::=
{(**SubclassOf**)* | (**InstanceOf**>)*}
(**ConceptContraint**)*****
(**Link**)*****
(**Property**)*****

SubclassOf

Declares that this concept is a subclass (subset) of another concept. Implicitly this concept is also a class.

```
SubclassOf ::=
    <subclass-of concept=ConceptId/>
```

subclass-of is considered to be a special type of association that is a subassociation of *child-of* and has an inverse of *has-class* and default cardinality. It is as if we defined the following association:

```
<association id="subclass-of">
    <subassociation-of association="child-of"/>
    <inverse id="has-class"/>
</association>
```

InstanceOf

Declares that this concept is an instance (member) of another concept, which itself must be a class. The cardinality is always one (a concept can only be an instance of one class).

```
InstanceOf ::=
    <instance-of concept=ConceptId/>
```

instance-of is considered to be a special type of association that is a subassociation of *child-of* and has an inverse of *has-instance* and default cardinality. It is as if we defined the following association:

```
<association id="instance-of" inverse-id="has-instance">
    <subassociation-of association="child-of"/>
</association>
```

ConceptContraint

Defines constraints on links between concepts. A constraint must adhere to the constraints defined in the global association definition. The constraints applies to all subclasses and instances of the concept.

```
ConceptConstraint ::=
    ExistsLink |
    ForallLinks
```

ExistsLink

Defines a restriction on the application of a link to the concept and its subconcepts. If the concept is ommitted it expresses a restriction on the number of concepts that can be linked by the association. If the concept is given it expresses a restriction on the number of concepts that can be linked by the association to the given concept and its subconcepts. If the cardinality is omitted a default of ">=1" is assumed.

```
ExistsLink ::=
    <exists-link association=AssociationId [concept=ConceptId]
    [cardinality=CardinalityExp]/>
```

ForallLinks

Defines a restriction on the application of a link to the concept and its subconcepts. Expresses a restriction such that all concepts linked by the association must be a subconcept of the given concept.

ForallLinks ::=
<forall-links association=**AssociationId** concept=**ConceptId/**>

Property

Defines a property of a concept. A property must adhere to the restrictions defined in the global attribute definition. The cardinality is always 1 (we can only have one property of a given name per concept/instance).

Property ::=
<property attribute=**AttributeId** value=**ValueExp/**>

Link

Defines a link between concepts. A link must adhere to the restrictions defined in the global association definition and also in the concept and its superconcepts.

Link ::=
<link association=**AssociationId** concept=**ConceptId** [strength=**Percentage**]/>

Others

Comment :=
<comment>**String**</comment>

Description ::=
<description>**String**</description>

OntologyId ::=
String

OntologyUri ::=
String

ConceptId ::=
{**String |** Top}

AssociationId ::=
{**String |** child-of | has-child}

AttributeId ::=
String

DataTypeId ::=
{String | Integer | Float | Date | Boolean}

ValuesExp ::=
{Value | ValueRange | ValueList}

Value ::=
{**String | Integer | Float | Date | Boolean**}

[Note that currently all values are represented as strings, regardless of type]

ValueRange ::=
Value-**Value**

ValueList ::=
({**Value** | **ValueRange**})*

CardinalityExp ::=
[{> | < | >= | <=}]**Integer**][,**CardinalityExp**]

Percentage ::=
An integer between 0 and 100

String ::=
A sequence of characters possibly with spaces

Integer ::=
A positive or negative integer

Float ::=
A floating point number

Date ::=
A date of the form DDMMYYYY

Boolean ::=
{true | false}

Content Extensions for EOL

These extensions to the Core EOL are provided to allow the addition of content-specific information to the ontology.

Ontology

Defines an ontology. The content extensions add additional tags to the Core ontology definition. However, generally it is advisable to separate the main ontology definition from the content definition by importing the definition into the content ontology.

Ontology ::=
<ontology id=**OntologyId** uri=**OntologyUri** version**=Integer**>
Metadata
Import*
Association*
Attribute*
Concept*
ConceptContent*
</ontology>

ConceptContent

Defines actual resources or rules for mapping resources to concepts.

ConceptContent ::=
<concept-content concept=**ConceptId**>
(**Resources**)*
(**Rules**)*
</concept-content>

Resources

Provide a set of resources of a particular type that are associated with the concept.

```
Resources ::=
    <resources content=ContentId>
        (Resource)*
    </resources>
```

Resource

Provides a reference to a specific resource.

```
Resource ::=
    <resource uri=ResourceUri [name=String]/>
```

Rules

Defines how a specific type of content should be mapped to the concept.

```
Rules ::=
    <rules content=ContentId [name=String]>
        (Rule)*
    </rules>
```

Rule

Defines a pattern which, if found in a resource, will provide evidence that that resource is relevant to the concept.

```
Rule ::=
    <rule logic=RuleLogic/>
```

RuleLogic

Specifies a pattern for matching resources to concepts. The actual form the pattern description takes is dependent on the specific pattern recognition technology being used. Details of the Exteca Pattern Matching rules can be found in a separate document.

```
RuleLogic ::=
    String
```

ResourceUri

A unique locator for the resource.

```
ResourceUri ::=
    String
```

ContentId

A unique identifier for the content type or source.

```
ContentId ::=
    String
```