RDFe – expression based translation XML to RDF

Hans-Jürgen Rennau, parsQube GmbH, 2019

Presented at xmlprague 2019

Slides

Slide#1 [Title page]

RDFe is a model and a tool for mapping XML documents to RDF triples. The „e“ stands for expressions – the mapping is based on XPath expressions.

Slide#2 [painters.xml]

Before talking about RDFe, I‘d like to talk about XML and RDF. Here is a simple XML document. In 1936 the painter Rene Magritte created a painting with the title „Clairvoyance“. That’s obvious. Less obvious it is how we rely on intuition. The <painting> element contained by the <painter> element is about a work which the artist created. “Contained” becomes “created by”, the semantic relationship is inferred from the structural relationship. It looks evident – but you would not be able to write a program extracting this information, unless your program is written with a particular document type in mind. Consider – is an element describing a painting necessarily a descendant of the element describing the artist, if such an element occurs in the document?

Slide#3 [paintings.xml]

No! Counter example - here, the element decribing a painting is an ancestor of the element describing the painter. And yet this document is as easy to understand as the previous one. We grasp the perspective conveyed by the document. The perspective is encoded by hierarchy. If the focus is on painters, their paintings are details described by descendant elements. If the focus is on paintings, their creators are details described by descendants of the element describing the painting. Hierarchy reduces complexity, but it introduces an arbitrary choice of focus. The choice streamlines a particular communication, but it is ephemeral. Tree-structure is optimal for messages, but perhaps not for storage of information to be reused in unforseeable ways, in an unforeseeable context.

Slide#4 [painters-zh.xml]

The intuition aforementioned is defeated if we do not understand the natural language underlying the tag names. We are at a loss and do not know how to map the shape to the information contained.

Slide#5 [painters.ttl]

This RDF graph captures the information conveyed by the previous documents. Semantic content, purified and devoid of hierarchy. Structural hierarchy replaced by semantic relationship. A single graph expressing the semantic value of three very different shapes.

Slide#6 [Semantic content and tree shapes]

Three different documents can be mapped to a common graph – a normalizing reduction to semantic content. And expansion of this content into alternative trees, tailored for different needs – each tree shape is like a different channel of publication. If tools can translate in both directions – reduction and expansion – interesting opportunities arise. For example – transformations between the tree formats can be validated by checking the semantic consistency between target and source.

But what gives me the right to say – this document expresses that graph, or the other way around? Perhaps you disagree. When you and I look at XML - can we communicate what RDF we see, and why? We need a model.

Slide#7 [RDF in XML]

First: let us align *RDF resources* with individual XML nodes, accepted as their representation. Call these XML nodes *resource nodes*. They serve as a point of departure when navigating to other XML nodes giving us the property values of that resource. Call those XML nodes *value nodes*. When the property value is itself a resource, the value node is the resource node of another resource. When the property value is literal (e.g. a string), the value nodes can somehow be mapped to the property values. Keep in mind that a literal property value is not necessarily represented by a single XML node. It may be a substring of element content, or a concatenation of various element contents. But we assume for each property an *expression* which navigates from the resource node to the value nodes and maps them to the property values. Call these expressions *value expressions*.

RDF resources are identified by an IRI, so resource nodes must be mapped to IRIs. Sometimes the IRI can be constructed from XML node contents. At other times we must resort to external data sources, enabling a lookup, given some XML values. But in any case, our vehicle is an expression which is evaluated in the context of the resource node and yields the resource IRI. Call this expression the *IRI expression*.

Slide#8 [Key concepts]

To summarize – looking at XML and thinking of RDF, we see resource nodes and value nodes, and we imagine value expresssions and IRI expressions. The gist of it all is a mapping of relationships: semantic relationships between subject and object are mapped to structural relationships between resource nodes and value nodes. And there is no better way of expressing structural relationships within a tree than XPath. XPath is the world champion. So the quintessence of translating XML into RDF is a mapping of RDF property IRIs to XPath expressions. That's what RDFe is about.

Slide#9 [Let RDF pull …]

Let us assume a "pull" perspective. Our painters document described two resources, with three properties each. So we need six value expressions, and two IRI expressions. We write a document called a semantic map, which attaches to resources and properties receptacles for expressions (@iri and @value attributes). We fill the receptacles and let RDF pull the strings of XPath …

Slide#10 [pull … the strings of XPath]

That's it, essentially. Submitting to an RDFe processor XML documents along with a semantic map, we get the RDF graph capturing the semantic content of the documents.

Slide#11 [A very basic semantic map 1]

Let us take a closer look at semantic maps. A semantic map has a target constraint. The processor will never apply the map to a document which does not meet the target constraint. It comprises the name and namespace of the root element, as well as optional assertions. <namespace> elements define prefix bindings used by the RDF output. And <resource> elements represent resource models – a semantic map is essentially a collection of resource models.

Slide#12 [Resource model (for <painter>)]

A semantic map is a collection of resource models, and each resource model is a recipe for mapping a resource node to a resource description. A resource model has a target node constraint, which tells the processor to which XML nodes the model may be applied. The resource model contains a list of RDF classes to which the resource belongs, an IRI expression and a set of property models. The IRI expression maps the resource node to a resource IRI. A property model describes how to construct triples with a particular property IRI. The property values are obtained from the value expression found in the @value attribute.

Slide#13 [Resource model (for <painting>)]

And here's another resource model. The value expression can be accompanied by a @type attribute specifying the type of the property value. When the type is set to the token "#resource", the nodes returned by the value expression are interpreted as resource nodes. The RDFe processor finds for each one an appropriate resource model, applies it to the node, obtaining a resource description and uses the subject IRI from that description as value of the property under construction.

Slide#14 [XPath – the moving part betwiXt XML and RDF]

This slide summarizes how a given RDF model – viewed as a set of resource types and their properties – can be populated by different types of XML documents. Each XML document type is translated into the RDF model by a specific set of XPath expressions. A semantic map connects an RDF graph model with an XML tree model, using XPath expressions as adapters.

Slide#15 [XPath – the moving part betwiXt XML and RDF]

The XPath expressions appear like moving parts transmitting the energy of XML data engines to the abstract carriage of an RDF graph model.

Slide#16 [RDFe processing model]

We have talked about how translation is controlled – but we have not yet considered the *scope* of translation – what to include and what to omit. To understand the problem, imagine this scenario. Data about painters include a list of museums exhibiting their work. You have a huge XML document with data about museums – locations, opening hours, etc. You want your graph about painters to include resource descriptions of museums, but please only of those museums as are referenced in your data. The problem is solved by the distinction between *asserted resource descriptions* and *required resource descriptions*. Asserted descriptions are the descriptions obtained for asserted target nodes. What's that? Every resource model has an optional assertedTargetNodes expression, which is applied to all input documents and yields the asserted target nodes of the model. Each resource model is applied to each of its asserted target nodes, producing the asserted resource descriptions. The asserted descriptions may include properties whose values are resources without an asserted description. The RDFe processor takes care that those resources will also be described – as these additional descriptions are considered required. As additional descriptions may in turn reference resources not yet described, the discovery of required resource descriptions is a recursive process. If the resource model of museums has no assertedTargetNode expression, only those museums will be described which are used by other resources as property values.

Slide#17 [RDFe – advanced features]

By now you understand the basic mechanics of RDFe – how to translate, and what to translate. Let us take a glance at a few advanced features. Semantic maps may import other maps. Each semantic map can define an evaluation context, consisting of named variables and functions. The context can enable you to simplify XPath expressions. Navigation to property values can be across document boundaries, and new input documents may be discovered on the way, by resolving document URIs found in the XML data. Using conditional properties, you can let the settings of a property model depend on XML contents, e.g. traversing links if they are present and staying with local nodes otherwise. RDF lists, reversed and inverse triples are supported. Cardinality constraints help you assure quality – warnings will be raised when an RDF property is supplied with less or more value items than you expect.

Slide#18 [Voilà, a catalog of paintings]

To get an impression of some of these features, imagine we have a catalog of paintings at our disposal, and we want to use it for augmenting the information about painters.

Slide#19 [… with a semantic map]

The catalog of paintings comes along with an own semantic map.

Slide#20 [Let's pull it in!]

Let's pull it in - easily done! We modify the semantic map for <painters> documents as follows. We add an import of the semantic map for our paintings catalog, and we modify the value expression of the „created" property. Using a complex XPath expression, we navigate into the catalog document and select the <painting> element with a title and date matching the title and date of the local <painting> element. As the <painting> element is found in the catalog document which is targeted by the imported semantic map, the processor will use the imported map, giving us the extended description which we want.

Slide#21 [Result, old]

Here‘s the result before the changes.

Slide#22 [Result, new]

Here‘s the new result.

Slide#23 [Better use a context]

The XPath expression was fairly complex. To enable simplified expressions, you may define an evaluation context, consisting of named variables and functions. The context is available to all expressions from the containing semantic map. We use the context for introducing a map-defined function – „getPainting“ - which returns the <painting> element from the catalog, given a title and date. With a call of this function, the value expression of the „created“ property becomes simple. Important detail about context: for each input document, a distinct instance of the context is constructed, using the input document root as the context item. This means that the context may reflect document-specific contents.

Slide #24 [Conditional settings]

Sometimes you may want to modify the settings of a property model in response to what the value expression returns. Imagine, for instance, that <painting> elements have an optional @wikidataID attribute, which allows lookup of comprehensive information in a catalog. You want to take advantage of the catalog element when possible, and stay with the local element otherwise. It can be implemented like this…

Slide #25 [RDFa - RDFe]

Why RDFe if we have RDFa? Simply because they are complementary. RDFa is for text oriented, human authored documents, where markup is about layout and gives no clue to the semantic content. Here, the addition of semantic attributes is appropriate. RDFe is for data-oriented XML, machine generated and using markup which is semantically rich. In this scenario, RDFa is problematic, because it modifies the original XML, violating schemas and cluttering content. RDFe, on the other hand, is non-invasive, and it lets you exploit the semantic information pent up in the markup to the fullest.

Slide #26 [RDFe – main goals]

RDFe is meant to lower the barrier separating RDF from XML. Given any data-oriented XML, an RDF representation should be only a short and simple step away. No matter how complex your XML is, and how distant from the RDF vocabulary. The perspective is emphatically not a single document one. RDFe is designed to cope with Linked Data scenarios: linking the contents of multiple documents of multiple document types, discovered iteratively.

Slide #27 [RDFe - issues]

At this moment, the main issue is performance. The processing of larger amounts of input data can be very slow. But these problems are not of a fundamental kind – they are caused by a rather naive implementation. The ideal would be RDFe support built into XQuery processors.

Slide #28 []

XML and RDF are complementary technologies, more than it is generally thought. To take advantage of this, you need tools that let you move between them easily. RDFe is a tool and a model. It is my hope that some people will find it useful. It would be my ideal if some people found it useful and inspiring, inspiring them to see RDF in XML, XML in RDF, more clearly.

Slide #29 [Merci!]

And that's the end of this presentation, isn't it.