Declarative Amsterdam 2020 Program

Declarative programming is a new approach to applications. Rather than describing exactly how to reach the solution, it describes what the solution should look like, and leaves more of the administrative parts of the program to the computer.

In the 50s, when the first programming languages were designed, computers cost millions, and relatively, programmers were almost free. Those programming languages therefore reflected that relationship: it didn't matter if it took a long time to program, as long as the resulting program ran as fast as possible. Now, that relationship has been reversed: compared to the cost of programmers, computers are almost free. And yet we are still programming them in direct descendants of the programming languages from the 50's: we are still telling the computers step by step how to solve the problem.

Contents

Declarative Amsterdam Conference (DA)

Stakeholder(s):

Declarative Amsterdam Organizing committee

Bieke van der Korst :
(Ambrac)

Daan Vernooij

Nico Verwer :
(Rakensi)

Pieter Lamers :
(John Benjamins Publishing Company)

Pieter Masereeuw :
(Masereeuw ICT - uitgeeftechnologie)

Steven Pemberton :
(CWI)

Declarative Amsterdam Sponsors

W3C Benelux

CWI :
Centrum Wiskunde & Informatica (CWI) is the national research institute for mathematics and computer science in the Netherlands. Founded in 1946, CWI is part of the Institutes Organisation of NWO, NWO-I. While located at Amsterdam Science Park, our institute has strong international links, and we enjoy a global reputation for our innovative research.

Sister Conferences

Balisage: The Markup Conference

Markup UK

XML Prague

Vision

A new approach to programming applications

Mission

To describe what the solution should look like

10-08-1. XForms

Learn about the structure and workings of XForms

Stakeholder(s)

Steven Pemberton :
(CWI) Instructor

BBC

Xerox

National Health Service

Declarative Applications with XForms — In the 50s, when the first programming languages were designed, computers cost millions, and relatively, programmers were almost free. Those programming languages therefore reflected that relationship: it didn't matter if it took a long time to program, as long as the resulting program ran as fast as possible. Now, that relationship has been reversed: compared to the cost of programmers, computers are almost free. And yet we are still programming them in direct descendants of the programming languages from the 50s: we are still telling the computers step by step how to solve the problem. Declarative programming is a new approach to applications: rather than describing exactly how to reach the solution, it describes what the solution should look like, and leaves more of the administrative parts of the program to the computer. One of the few declarative languages available is XForms, an XML-based language that despite its name is not only about forms. Large projects, at large companies such as the National Health Service, the BBC and Xerox, have shown that by using XForms, programming time and cost of applications can be reduced to a tenth! This hands-on tutorial allows you to learn about the structure and workings of XForms, and gives you the opportunity to work on useful working programs. It is a “bring your own device” tutorial. You will be required to install some files beforehand (details to follow), and check they are working. You will be able to work using the text editor of your choice during the tutorial.

10-08-2. Greenfox

Describe resource relationships independently of the constraints using them.

Stakeholder(s)

Hans-Jürgen Rennau :
(parsQube GmbH) Instructor

An introduction to Greenfox, a schema language describing file system contents Schema validation is like a prototype of the declarative approach – to describe, rather than to code. The tutorial introduces to Greenfox, a schema language for file system contents. Resources are described by shapes, which are sets of constraints. The goals of the tutorial are twofold. First, it should help to get started with using Greenfox. Second, it should awake an awareness that the same set of constraints can be described in a less or more declarative way. This possibility is above all opened by a new possibility, offered by the upcoming version of Greenfox, to describe resource relationships independently of the constraints using them. Participants might acquire an increased awareness that it is not sufficient to ask “if” declarative or not, but we should also ask “how” declarative, think about degrees of declarativeness.

10-08-3. Saxon-JS

Compile stylesheets for use in the browser.

Stakeholder(s)

Pieter Masereeuw :
(Pieter Masereeuw ICT B.V.) Instructor

Hands-on with Saxon-JS — XSLT in the browser has long been a promise, but nowadays, creators of web browsers have lost interest. They only support the early version of the standard, but very often support is even missing or going to be withdrawn. This is a pity. A fact is that many XSLT transformations can be done on the server, but transformations in the client side can still be useful, for instance when XML documents are retrieved from an external source and have to be formatted. Saxon-JS solves this problem. Additionally, and very interestingly, it can also come in the place where people normally apply Javascript. Saxon-JS is able to respond to user events such as clicks, keystrokes, focus events, finger events and much more. It can handle such events by applying XSLT to the HTML document that lives inside the browser, for instance by changing attributes or by adding or removing content. My tutorial will give you a hands-on experience by compiling stylesheets for use in the browser and putting it to work for some common use-cases. Bringing your own laptop is required if you want to fully participate.

10-09-01. Declarative Approaches

Consider success factors and pitfalls of declarative approaches

Stakeholder(s)

Nico Verwer :
(Rakensi) Presenter

Success factors and pitfalls of declarative approaches — Declarative programming may not seem to be widespread, but it has been used for several decades in some areas of information technology. Examples are SQL for querying databases, and regular expressions and grammars for text analysis. More recently, domain-specific languages have been used to take advantage of declarative methods, with varying degrees of success. What can we learn from the successful applications of declarative programming? And perhaps more importantly, is there something that failed applications have in common? In this presentation we will look at declarative techniques that are so ubiquitous that nobody notices them anymore. We will also look at two pitfalls that the author has encountered many times: genericity and reification.

10-09-02. XML Parsing

Release an open-source fully compliant Invisible XML parser

Stakeholder(s)

Tom Hillman :
(eXpertML Ltd) Presenter

JayParser: an Invisible XML implementation in XSLT — In XML Prague this year I presented on a proof of concept grammar parser in XSLT. In this presentation I plan to release an open-source fully compliant Invisible XML parser based on this work. The presentation will demonstrate iXML parsing, some technical details of the implementation, and finish with a short overview of the advantages of this declarative approach of parsing text. I also hope to show how it is possible for the parser to extend itself to other grammar languages.

10-09-03. XQuery

Compile XQuery to native machine code

Stakeholder(s)

Adam Retter :
(Evolved Binary) Presenter

Compiling XQuery to Native Machine Code — It has become a trend amongst modern high-performance databases to optimise queries by avoiding interpretation of the query language. They instead opt to compile queries to native machine code which can subsequently be executed directly by one or more CPUs and/or GPUs. Both Saxon and XSLTC have previously demonstrated, albeit by different means, compilation of XSLT and XQuery to Java bytecode which can then be executed by the Java Virtual Machine. To the best of our knowledge, we are the first group to demonstrate the compilation of XQuery directly to native machine code. As part of a research effort to develop a new performant XQuery processor for our poly-store database (FusionDB), we are constructing an XQuery parser which emits LLVM IR (Intermediate Representation), and a JIT (just-in-time) compiler to produce native machine code which is then executed. In this paper we review the current approaches to native query compilation, detail our challenges and progress in building a modern XQuery parser, and our use of LLVM for compiling XQuery to native machine code. We also consider how we might exploit LLVM's low-level IR optimizations to improve query performance.

10-09-04. LwDITA

Discuss the construction of and demonstrate LwDITA editor for technical documentation.

Stakeholder(s)

Charaf Eddine Cheraa :
(Evolved Binary) Presenter

Adam Retter :
Co-Presenter

Petal - An in-browser editor for LwDITA — With the meteoric rise of HTML5 and the slow demise of XHTML, simple in-browser WYSIWYG editors for producing XML documents have all but disappeared. Today, there are numerous Open Source JavaScript components for simple in-browser editors (e.g. CKEeditor, Editor.js, Quill, TinyMCE, etc.) that produce documents in a subset of HTML5. Likewise, there are free and commercial SaaS offerings that edit and publish reStructuredText or Markdown documents from within the browser (e.g. readthedocs.org, gitbook.com, and mkdocs.org). The lack of simple in-browser XML editors, is likely driven by several factors: 1. The complexity of offering a full XML editor. One of the advantages of XML is its flexibility, an author can arbitrarily decide on their document structure, and element and attribute names, however this adds complexity to any such editor. Conversely schema languages for XML that limit that flexibility by enforcing a certain document grammar, then require extra parsing and validation steps by the editor. 2. A decrease in XML processing support within the browser itself. 3. A perceived reduction in what constitutes an acceptable level of re-use and presentation for technical documentation. When compared to HTML5, Markdown, and reStructuredText, we believe that there are still key advantages that can be exploited by using XML for technical documentation. For the purposes of authoring and publishing the documentation for FusionDB Server, we examined several markup formats but ultimately settled on LwDITA. We intend to show that LwDITA occupies an optimum position, it allows us to reap the benefits of XML, whilst remaining simple enough to allow us to develop a simple and compliant in-browser editor. Whilst we acknowledge that there are a handful of existing commercial and/or enterprise offerings which include in-browser XML editors (e.g. Oxygen XML Web Author, easyDITA, Xopus, etc.), they are in themselves very comprehensive and complex products with accompanying costs. Rather, we intend to both discuss the construction of, and demonstrate, our in-browser LwDITA editor as an intentionally simple solution for editing technical documentation.

10-09-05. URLs

Consider the design of the URL.

Stakeholder(s)

Steven Pemberton :
(CWI) Presenter

On the design of the URL — Notations can affect the way we think, and how we operate; consider as a simple example the difference between Roman Numerals and Arabic Numerals, where Arabic Numerals allow us not only to more easily represent numbers, but they also ease manipulations of numbers and calculations with them. One of the innovations of the World Wide Web was the URL. In the last 30 years, URLs have become an ever-present element of everyday life, so present that we scarcely even grant them a second thought. And yet they are a designed artefact: there is nothing natural about their structure — each part is there as part of a design. This talk will look at the design issues behind the URL, what a URL is meant to represent, and how it relates to the resources it identifies, and its relationship with representational state transfer (REST) and the protocols that REST is predicated on. The talk will consider what mistakes, if any, were made, and with hindsight how if at all the design could have been improved. While it is too late now to change the design of URLs, we will consider what the lessons are that we can draw from their design to direct the future designs of notations.

10-09-06. Quality Control

Demonstrate how quality control infrastructure can be generated from a requirements document.

Stakeholder(s)

Tom Hillman :
(eXpertML Ltd) Presenter

Vincent Lizzi :
(Taylor & Francis) Co-Presenter

Self-Generating Quality Control: A Case Study — This paper demonstrates how quality control infrastructure can be generated from a single requirements document. Taken from a recent project that is now being used in production at a large journal publisher, it discusses some of the challenges faced and techniques used when generating Schematron, XSpec tests, XML grammar checks, and documentation. The project set out to implement quality control requirements for journal articles using Schematron. In pursuing this objective, the project also created quality control infrastructure for Schematron itself that streamlines the process for incorporating iterative changes to requirements. The techniques used in this project and described in this paper may be generally applicable in other projects.

10-09-07. Text Processing

Process plain text in structured documents

Stakeholder(s)

Nico Verwer :
(Rakensi) Presenter

Plain text processing in structured documents — Applications that analyze and process natural language can be used for things like named entity recognition, anonymization, topic extraction, sentiment analysis. In most cases, these applications use the plain text of a document, and may add or change markup. This causes problems when the original document already contains markup that must be preserved. The text to be analyzed may run across markup boundaries, and newly generated markup may lead to unbalanced (non well-formed) structures. This presentation shows how the Separated Markup API for XML (SMAX) can be used to apply natural language processing to XML documents. It preserves the existing document structure and allows for balanced insertion of new markup. A demonstration will be given of the use of SMAX for extracting and marking references in legal documents. This Link eXtractor was built for the Dutch center for governmental publications. SMAX and Simple Pipelines of Event API Transformers (SPEAT) will be available as open source software at the time of Declarative Amsterdam.

10-09-08. Language Solutions

Develop language solutions based on TEI and ODD.

Stakeholder(s)

Eduard Drenth :
(Fryske Akademy) Presenter

Development of language solutions based on TEI and ODD — The Text Encoding Initiative (TEI) is a vast, long standing and widely used encoding standard, covering different areas in the humanities. High quality documentation with many examples and active discussions on the rationale behind the available elements and attributes and their intended use are among the many qualities of the TEI. The TEI presents itself as guidelines, trying to cover as many areas and use-cases in the humanities as possible. The TEI is also designed to be customized for use in specific situations. Customization is achieved via a "One Document Does it all" (ODD). An ODD offers a mechanism to override, restrict, eliminate and extend (parts of) the guidelines in a documented way. ODD can be seen as a powerful abstraction layer from which validation, documentation, but also processing models can be generated. A nice, but complex feature of ODD is that they can be chained, enabling you to have focused ODD's and to promote reuse. In my work on corpora and dictionaries at the Fryske Akademy ODD is the basis from which I generate XSD, configuration, SQL, Java, bind.xml etc. In my presentation I will show you how we benefit from ODD in for example editing and publishing solutions, our goal being to enhance tool development and interoperability through standardization. Outline of the presentation: 1. ODD - explanation and background - chaining - generation 2. Usages - corpora - dictionaries - lexicons - interoperability 3. editing - oXygen: validation and customizing author mode 4. The future - processing model and TEI Publisher

10-09-09. TV Applications

Develop TV applications in IPTV systems.

Stakeholder(s)

Alan Guedes :
(TeleMídia Lab, PUC-Rio, Brazil) Presenter

Sergio Colcher :
(advisor)

Declarative Programming of TV Application Using NCL — NCL is the declarative programming language used to develop TV applications in IPTV systems and Terrestrial TV standardized by ITU and Brazilian TV Forum, respectively. Its main characteristics are support: defining temporal synchronization among media assets and viewer interactions; layout reuse facilities (and); support multi-device presentation; scripts in the light-weight and embeddable language Lua; and an API for building and modifying applications on-the-fly called NCL editing command. This talk briefly introduces NCL, highlights its recent advances and discuss the future of the language. https://www.itu.int/rec/T-REC-H.761a

10-09-10. Text Parsing

Parse text with XSLT 3.

Stakeholder(s)

Liam Quin :
(Delightful Computing) Presenter

Parsing text With XSLT 3 — Although there is at least one parser generator that targets XSLT, this talk is about hand-written parsers. Some difficulties will be described, along with mitigations. Some techniques made possible by XSLT 3 will be described and some examples given. Attention to debugging and testing is also given. The techniques have much wider application than formal parsing, and it turns out to be both useful and fun.

Administrative Information

Start Date: 2020-10-08
End Date: 2020-10-09

Publication Date: 2020-09-15
Source: https://declarative.amsterdam/list?model=da-program

Submitter:

Given Name: Owen
Surname: Ambur
Email: Owen.Ambur@verizon.net
Phone:

PDF formatted using TopLeaf XML publisher
www.turnkey.com.au