GXMLProposal

Apache gXML Proposal

Contents

Abstract

gXML will provide Java™ APIs, based on the XQuery Data Model, for tree model-independent processing of untyped and typed XML, with concrete implementations of bridges to noted tree models such as DOM and Axiom, and processors for noted tasks such as XPath evaluation.

Proposal

The gXML API is an implementation of the Handle/Body (or Bridge) design pattern, leveraging and unifying existing XML tree model APIs as implementations of the XQuery Data Model (XDM). XML tree model processing in Java faces several problems related to the primacy of DOM in Java's history. The design of gXML is intended to address these problems and to enable the state of the art for XML processing in Java to advance.

gXML can operate over any tree model for which a bridge exists, and promotes runtime injection of the model, enabling XML processors to select the model most suitable for the task without loss of interoperability. The API is specified by the XDM, removing the variability in properties and behaviors characteristic of the underlying tree models.

gXML introduces several new processing paradigms:

- in which XML is regarded as immutable: enables a degree of performance enhancement for existing tree models, enables more significant enhancement via custom models, and facilitates multi-thread/multi-core processing
- in which XML can be traversed with a "cursor": enables more flexibility in what is kept in memory and how, for example facilitating large document handling
- in which XML is "typed" with XML Schema validation and the XDM: facilitating the implementation of newer specifications like XQuery 1.0, XSLT 2.0, and XPath 2.0.
- in which underlying tree models can be substituted at runtime: enables assessment of design trade-offs appropriate to a particular task

This proposal hopes to establish an Apache project around these new XML APIs. Specifically we aim for this project to improve with exposure to a broader community. We expect that the project will refine the existing APIs developed so far, enhance the documentation, enhance existing implementations of these APIs, and explore some of the key opportunities that these APIs open up.

Background

The eXtensible Markup Language (XML) is a structured format for the textual representation of data. Because of its utility and simplicity, XML has been widely adopted for computer-to-computer communication over the Internet. APIs for processing XML in Java™ were developed early, particularly the Document Object Model, defined in Interface Definition Language and known to present a number of challenges for developers. Many tree models have been developed to address these challenges: JDOM, DOM4J, AxiOM, XOM, and a number of less well-known and even proprietary models.

W3C XML Schema provides authors of XML documents for interchange with a language for specifying the content of those documents, both structure (elements and attribute) and datatypes (text content in elements and attributes). XPath 1.0 provides navigation and selection of nodes in XML trees; XSLT 1.0 a transformation language for these trees. XPath 2.0 and XSLT 2.0, together with XQuery 1.0, are based on the XPath 2.0 and XQuery 1.0 Data Model ("XQuery Data Model," or "XDM" for short), providing a rigorous definition of properties and behaviors for XML trees, and adding optional schema-awareness based on W3C XML Schema.

The developers behind this contribution identified an approach that leverages the value of existing tree models, while enabling new uses opened up by the specifications just identified. Specifically, we thought to apply the Bridge, or Handle/Body design pattern. As applied to XML, these design patterns may be contrasted with the Facade, or Wrapper pattern: where a wrapper has at least one object instance for each node wrapped, a bridge increases weight by only a single class instance for all trees of a given type. To ensure type-safety, gXML applies the Bridge pattern by making extensive use of Java™ generics.

Rationale

gXML was originally conceived in 2006 to address a number of fundamental issues facing the XML community:

- multiplicity of models
- leverage existing investment
- performance
- foundation for building for new specifications

Manage the multiplicity of tree model APIs

Appropriately, over the years, Java has accumulated multiple XML tree APIs to address one or more design trade-offs present in existing models. Those models include DOM, JDOM, DOM4J, AxiOM, XOM, and even Xalan's internal Document Table Model. However, all the other models have had to overcome the network effects of DOM - the standards-based implementation first on the scene. This network effect creates problems for any processing engine that chooses to work with a model other than DOM.

By way of the handle/body pattern, and by providing a common API over all supported tree models and by encouraging run-time injection of an appropriate tree model, gXML enables application developers to choose the model best suited to the application, or even best suited to a particular portion of an application.

Leverage investment in XML processor development

An XML processor can be loosely defined as any library of code that processes XML. The gXML contributed code includes five processors, partly as exemplars and partly to provide commonly-required functionality independent of tree model. New processors built over the gXML API can be independent of the underlying tree model. Existing processors can be incrementally refactored to use the gXML API, gaining tree model independence (and often additional functionality) in the process.

A key benefit of gXML is that existing processors targeted to a particular tree model for which a bridge exists can be used as-is; a processor over DOM, such as Xalan, can work seamlessly with a gXML-based processor using the DOM bridge, with no conversion cost. Similarly, SOAP or WS-* gXML-based processors can interoperate with Axis using the AxiOM bridge. This is potentially very powerful; new and refactored processors can gain mindshare without being forced to choose among tree models.

gXML also includes a conversion processor, though its use is discouraged, which uses the gXML input and output APIs to permit conversion between any supported pair of tree models, which potentially extends the reach of processors (at a cost; conversion is not cheap).

Improve XML performance

gXML defines its base API as an untyped and immutable XML tree. As the API reflects the XDM specification, it includes a standard extension to support types (including types and typed values for instance documents, and a complete W3C XML Schema model). This extension preserves the principle of the immutable tree. Treating XML trees as immutable is potentially quite valuable: it permits multiple threads to safely analyze the tree, and reduces the footprint of the objects in memory. It also allows for the treatment of non-XML such as tabular data and JSON as if it is XML format, while eliminating a conversion step for use with an XML processor.

All existing tree models are mutable; gXML also contains a standard extension that permits mutability, to ease the refactoring of processors and applications. gXML bridges over mutable tree models cannot take full advantage of the promise inherent in immutable trees, but can achieve some optimizations. The encouragement of model injection also suggests that it may become possible to design immutable tree models which can achieve further performance gains.

Similarly, the possibility exists, in the typed extension API, to pass objects optimized for type-aware processing, rather than performing schema validation at each step.

Critically, once a bridge implementation is created for a particular representation, that new implementation can be tried side-by-side with existing tree models, to assess which is most appropriate for a particular set of circumstances.

A "cursor" based processor will be able to work with XML document trees that don't fit in the available RAM.

Establish a foundation API for new and future XML Specifications

In January 2007, the W3C moved eight specification to Recommendation status (XQuery 1.0, XQueryX 1.0, XSLT 2.0, Data Model, Functions and Operators, Formal Semantics, Serialization and XPath 2.0). These specifications are built upon a new data model that sports atomic values as leaf nodes on the data model tree structure, and they depend on W3C XML Schema to provide the type system foundation. gXML provides an API for the XDM.

The XDM defines seven node types, as against eleven in the XML Infoset, or twelve in the DOM. The XDM is simpler, more consistent, and better suited for processor/application use, and it provides the foundation for more advanced use cases.

gXML opens the door to APIs and technologies both "underneath" and "above" the bridges it defines. "Underneath" gXML you can find different forms of APIs to represent data as XML. Above gXML, you can find some existing processors like XPath, but perhaps also some new and unanticipated ones which would otherwise struggle with which tree model is the least-worst compromise.

Looking Forward

Given all the opportunities that gXML opens up for exploration, we anticipate that the gXML APIs will dramatically improve the life of a developer using XML.

Initial Goals

The primary challenge facing gXML is the fostering of a community of developers to move it forward. The initial code contribution is meant to provoke interest by providing not only the API, but three bridges and five processors as well. Two bridges address high profile, widely used tree models; the third provides a history-free implementation example. The processors were chosen because they were both small enough to accomplish, and interesting enough to add useful functionality.

The contributors hope that the API will not change dramatically, although it is expected that it will be refined. The goal for the API modules is stability. Bridges are known to have missing bits of functionality, which need to be filled in; the goal for bridges is completeness, correctness, and contribution (bridges for other tree models, or even other hierarchical data models). The goal for processors is similar: complete the functionality, and encourage contributions of new functionality.

Once the API has proven stable and mature, it is hoped that development will move away from gXML focus, into independent projects that adopt the gXML API for processing.

Current Status

gXML was conceived in 2006; it has been undergoing significant refinement based on feedback from internal adopters. Up to this point, the API and all implementations have been proprietary, but opening the source has been a goal (and consequently has been factored into the development process) since 2007 at the latest.

The core APIs are based on published standards (particularly the XQuery Data Model). Implementations are for high-profile targets (both for bridges and for processors). It is hoped that the API is stable; the implementations are known to be incomplete; contributions are much desired.

Meritocracy

We look forward to growing gXML in an environment that supports a meritocracy. As a broad foundational technology, gXML can only benefit from the diverse perspectives of many users and developers, and it will only be successful if it represents the goals of large majority. The Apache meritocracy structure will lower the barriers for contribution while at the same time providing governance to ensure quality.

Community

gXML needs a community development process. It is designed as a core, infrastructural technology for processing XML in Java, unifying the fractured area of tree model development. Contributors and committers are making efforts (publications and presentations) to attract interest. It is hoped that, by contributing the project to Apache, it will become more robust, more visible, and better positioned to survive over time.

Core Developers

gXML was created by David Holmes in 2006; his vision and energy drove it until January 2010. Amelia Lewis, Eric Johnson, and Joe Baysdon were seconded to the project at one time or another; they provided reviews, implementations, and proofs of concept. As of January 2010, the project has fallen to Amy, Eric, and Joe to carry forward. All four developers have been active in the development of XML technologies in Java for approximately ten years. This is not a particularly diverse group, but these developers have shared experiences and knowledge around the problems associated with processing XML in Java (particularly in the enterprise/high performance/high volume context) that have informed the development of the project.

Alignment

Apache is the home of some of the premier XML technologies for Java, notably Xerces and Xalan. The list of projects that gXML might touch, or that might make use of gXML, is extremely extensive: Axis, AxiOM, Woden, XML Security, and Neethi have all been explored by the gXML team as possible "converts" to use of gXML. gXML includes a bridge for AxiOM. As an embodiment of the XQuery Data Model, gXML provides a sound foundation for the development of processors for XPath2 and XSLT2 (from a Xalan base?), as well as for an XQuery processor.

Apache is the right place for a vision of XML in Java that escapes from the constraints of the DOM without attempting to supplant it.

Management of Known Risks

Orphaned Products

The possibility of gXML being orphaned (or "lost in incubation") was acknowledged and addressed by the corporate contributor (TIBCO Software Inc.) prior to creation of this proposal. Specifically, we have plans to continue using this technology in our projects, meaning that our own interest will remain strong. We've also recognized the need to publicize this project to gain additional interest, and have already submitted a paper to one conference, and are planning on attending others. We are considering modern publicity techniques, such as blogging, and explicitly "announcing" the project in our traditional marketing process. In addition, with our collective years in the business, we do have contacts across a number of companies that might be interested in the technology. In short, while the initial committers are salaried developers paid by the corporate contributor, we recognize that the project needs greater participation, and plan to work to find that.

Developers believe that the code is mature enough (or will be soon) to be adopted by other projects, and has functionality that makes the risk of doing so well worth taking; this should help to drive interest and involvement.

(Some) Inexperience with Open Source

The initial contributors have some experience with open source projects, although largely from the periphery. We've filed bugs (with occasional patches), proselytized its use, and contributed to mailing lists. We look forward to the prospect of active discussions and the meritocracy as we see it will almost certainly be a huge positive influence on this gXML project.

The company we work for has moved over the past decade from minimal direct open source involvement, to having a formal process for establishing an "open source" direction for projects such as the one behind this proposal. Our process includes all the appropriate legal clearances. In addition, our company now specifically sponsors some amount of open source development.

Homogeneous Developers

Initial committers are certainly homogeneous. All are committed to encouraging participation from all sources, and are looking forward to the new directions that new contributors will likely to bring to the project.

Reliance on Salaried Developers

As with the problem of developer homogeneity, the initial contribution of code and support (in the form of developers) from TIBCO brings with it a significant danger. As with the problem of homogeneity, this one should be resolved by encouraging participation and recruiting committers. Initial committers are also sufficiently excited about the possibilities in the code that they expect to remain involved regardless of TIBCO's commitment.

Relationships with Other Apache Products

gXML builds a bridge for AxiOM. The schema parser and validator might be viewed as competing with Xerces, but aren't; they are designed as processors rather than as features within the parse engine. Likewise, there is no expectation of conflict with Xalan; gXML does not include any XSLT processors. However, the design of gXML is to enable schema-aware XSLT 2, XPath 2, and XQuery processors; we hope that the Apache implementations of those specifications build upon gXML principles, at least (but better if they build upon the API).

Proofs of concept (not included in the source tree) have used Apache Woden, Apache Neethi, and Apache Security as test beds. We hope to interest those projects, and perhaps others (even Axis) in making use of gXML.

gXML is an infrastructure project, designed in part to reconcile the XML tree model problem in Java. As a consequence, the more it is adopted (either via new bridges or via new processors), the more valuable it becomes to those who have adopted it, and the more attractive it is to those who are considering adoption.

An Excessive Fascination with the Apache Brand

Contributors are well aware of the power of the Apache brand, which is clearly one of the reasons for making the submission here rather than elsewhere. The Apache reputation has the potential to help address some of the known risks outlined above: developer homogeneity, open source inexperience, and reliance on salaried developers, all issues which will be best addressed by increasing involvement in the project. A high profile, borrowing from Apache's reputation, is the most direct way to attract new community members.

However, the contributors/initial committers are more interested in Apache because it is a repository of experience and knowledge (which is part of the reason that it has the "brand" or reputation that it does). The foundation's members and participants know how to carry a project through to success, and how to keep it going for as long as it is useful and interesting. We believe that gXML is going to be useful and interesting for at least ten years, and want to know how to build an environment for the project that ensures its continuation. It is primarily that experience and knowledge which fascinates the contributors, and we do not think that it is possible to be excessive in one's desire for either knowledge or experience.

Documentation

A programmer's reference provides more in-depth information on the architecture of gXML. Unfortunately, it is stale and its build is broken.

Javadoc exists for most modules, but more work needs to be done.

The documentation is, in a word, rather weak. Improving it is on the list of priorities.

Initial Source

The initial source was a proprietary development, beginning in 2006, designed to enable Java™ APIs for XQuery (and XPath 2 and XSLT 2). It has been through several iterations and code reviews. From quite early in development, attention has been given to the prospect of opening the source, because the API is more valuable as open source than as proprietary. The proprietary code base has been deployed for shipping products, but has since been refactored for greater generality and utility prior to submission.

Contributors expect to see the code base change further (but without breaking the principle of specification conformance), but assert that it embodies principles and practices that have already proven themselves in code.

A large portion of the project source is currently available here.

Source and Intellectual Property Submission Plan

The current structure of the repository (which uses Maven to build):

- api
- xpath-api
- bridge
    - bridgekit
    - bridgetest
    - dom
    - axiom
    - cx
- processor
    - input-output
    - iotest
    - convert
    - xpath.impl
    - w3c.xs
    - w3c.xs.validation

There is also a documentation module, which is currently rather stale. The general pattern is: API (which should see little change); bridges (including developer tools (bridgekit) and test/conformance tools (bridgetest)); processors.

All of the above have already been cleared by our employer for the creation of an open source project - whether or not that project is hosted at Apache.

We will also be soliciting contributions, for both bridges and processors, and will be encouraging the adoption of the API in existing XML processors and applications.

A number of additional tests and suites are targeted for inclusion; for externally-developed suites (such as the XML Schema Validation suite) that are targeted, the licenses have been verified as permitting use.

External Dependencies

The gXML core has no external dependencies beyond JDK 1.5, except for an implementation of StAX. For Java 1.6, this is satisfied in the JDK; for 1.5, the dependency is upon the Woodstox implementation (offered dual-licensed, LGPL or ASL, so we would be using it via ASL).

Each bridge carries a corresponding dependency. Three bridges are included: DOM (in the JDK), and AxiOM (hosted at Apache); the third is a proof-of-concept standalone implementation with no additional external dependencies. Others may be added, depending upon developer interest (bridges for models which have incompatible licensing may be refused hosting on the ground of license incompatibility).

Cryptography

The proposal does not involve cryptographic code.

Required Resources

Mailing lists

The following (standard) mailing lists will be required:

- gxml-private
- gxml-dev
- gxml-commits

Subversion Directory

https://svn.apache.org/repos/asf/incubator/gxml

Issue Tracking

JIRA gXML (GXML)

Other Resources

There are no other special infrastructure requirements.

Initial Committers

Joe Baysdon (jbaysdon at tibco dot com)

Eric Johnson (eric at tibco dot com)

Amelia Lewis (alewis at tibco dot com)

Affiliations

Joe Baysdon: TIBCO Software Inc.

Eric Johnson: TIBCO Software Inc.

Amelia Lewis: TIBCO Software Inc.

Sponsors

Champion

Paul Fremantle (Apache Member)

Nominated Mentors

Emmanouil Batsis <manos at abiss dot gr> [not ASF member]

Jochen Wiedmann <jochen dot wiedmann at gmail dot com> [not ASF member]

Sponsoring Entity

The Incubator PMC (requested).