xerces.apache.org

Xerces Native Interface

6-7 minutes

The Xerces Native Interface (XNI) is a framework for communicating a "streaming" document information set and constructing generic parser configurations. XNI is part of the Xerces2 development but the Xerces2 parser is just a standards compliant reference implementation of the Xerces Native Interface. Other parsers can be written that conform to XNI without conforming to any particular standards or using any code from the reference implementation.

The Xerces Native Interface is used to implement the Xerces2 parser from a set of modular components in a standard configuration. This configuration is then used to drive the DOM and SAX parser implementations provided with Xerces2. However, XNI is merely an *internal* set of interfaces. There is no need for an XML application programmer to learn XNI if they only intend to interface to the Xerces2 parser using standard interfaces like JAXP, DOM, and SAX. Xerces developers and application developers that need more power and flexibility than that provided by the standard interfaces should read and understand XNI.

Overview information:

- "Streaming" Information Set
- Generic Parser Configurations

Design and implementation information:

- Design Details
- Core Interfaces
- Parser Configuration
- Xerces2 Parser Components

What is meant by a "streaming" information set? Quite simply, the streaming inform set is the document information that can be communicated by parsing the documer serial manner. In other words, it is the information received as-you-see-it. An XNI pa provides this streaming info set to a registered document handler. The XNI docume handler is similar to the standard SAX `ContentHandler` interface but is different i several important ways:

- XNI attempts to provide lossless communication of the streaming information set. Therefore, XNI passes the encodings of external parsed entities and other informat is lost when using SAX.
- The XNI document handler interface is also designed to build a pipeline of parser components where the streaming information set can be fully modified and augmer each stage in the pipeline. SAX, however, is primarily a read-only set of interfaces.

The Xerces Native Interface breaks the document's streaming information set into s more manageable interfaces:

Interface	Description
`XMLDocumentHandler`	Communicates document structure and conten information.
`XMLDTDHandler`	Communicates basic DTD information such as element and attribute declarations.
`XMLDTDContentModelHandler`	Breaks down each element declaration's conte model into a set of separate methods so that ha don't have to reparse the content model string g the `XMLDTDHandler#elementDecl(String,St` method. This separation also helps those applic that want to know boundaries of entities when u

part of an element's content model.

And an additional handler is provided for convenience in defining document fragme

Interface	Description
`XMLDocumentFragmentHandler`	Communicates information about a document fragment.

For complete details of the Xerces Native Interface, refer to the Core Interfaces documentation.

The Xerces Native Interface document handler interfaces define a document's streaming information set but XNI also contains a set of interfaces that define parser components and configurations. These interfaces provide a framework for a library of parser parts that can be used interchangeably or completely replaced at the programmer's option. This framework allows an unparalleled level of configuration and implementation choices to implement XML applications.

The following list details some possible examples of parsers and configurations that can be written using the XNI parser configuration framework:

- **HTML Parser**
  An HTML scanner can be written that breaks an HTML document into a series of XNI callbacks. Using a configuration that swaps the default XML scanner with the HTML scanner, you can create DOM and SAX parsers for HTML documents.
- **Optimized Parser**
  For improved XML performance, a minimal XML scanner can be written and swapped for the default, fully compliant XML scanner. In addition, the validator component can be removed from the parser pipeline to reduce the amount of work required to parse XML

documents.

- **XInclude Processor**
  An XNI parser component can be written to handle XInclude by analyzing the streaming information set and automatically inserting the contents of referenced links into the event stream. By adding this component to the parser pipeline before the validator, included content would appear transparent to the validator as if that content was in the original document.

  This is just a small sample of what is possible when using the XNI parser configuration framework. For complete details of the XNI parser configurations, refer to the Parser Configuration documentation.