

# **Enterprise Computing: Java API for XML Processing (JAXP)**

**Stephen Gilmore**

**The University of Edinburgh**

**Lecture content copyright © 2003**

**Sun Microsystems, Inc.,**

**4150 Network Circle, Santa Clara,**

**California 95054, U.S.A.**

**All rights reserved.**

# Introduction

The **Java API for XML Processing (JAXP)** is for processing XML data using applications written in the Java programming language.

# Introduction

The **Java API for XML Processing (JAXP)** is for processing XML data using applications written in the Java programming language.

JAXP leverages the parser standards **SAX** (Simple API for XML Parsing) and **DOM** (Document Object Model) so that you can choose to parse your data as a stream of events or to build an object representation of it.

JAXP also supports the XSLT (XML Stylesheet Language Transformations) standard, giving you control over the presentation of the data and enabling you to convert the data to other XML documents or to other formats, such as HTML.

JAXP also supports the **XSLT (XML Stylesheet Language Transformations)** standard, giving you control over the presentation of the data and enabling you to convert the data to other XML documents or to other formats, such as HTML.

JAXP also provides namespace support, allowing you to work with DTDs that might otherwise have naming conflicts.

## **Pluggability in JAXP**

Designed to be flexible, JAXP allows you to use any XML-compliant parser from within your application.

## **Pluggability in JAXP**

Designed to be flexible, JAXP allows you to use any XML-compliant parser from within your application.

It does this with what is called a pluggability layer, which allows you to plug in an implementation of the SAX or DOM APIs.

## Pluggability in JAXP

Designed to be flexible, JAXP allows you to use any XML-compliant parser from within your application.

It does this with what is called a pluggability layer, which allows you to plug in an implementation of the SAX or DOM APIs.

The pluggability layer also allows you to plug in an **XSL processor**, letting you control how your XML data is displayed.



# The JAXP APIs

The main JAXP APIs are defined in the `javax.xml.parsers` package.

## The JAXP APIs

The main JAXP APIs are defined in the `javax.xml.parsers` package.

That package contains two vendor-neutral factory classes: `SAXParserFactory` and `DocumentBuilderFactory` that give you a `SAXParser` and a `DocumentBuilder`, respectively. The `DocumentBuilder`, in turn, creates a DOM-compliant `Document` object.

## **Plugging in XML implementations**

The factory APIs give you the ability to plug in an XML implementation offered by another vendor without changing your source code.

## Plugging in XML implementations

The factory APIs give you the ability to plug in an XML implementation offered by another vendor without changing your source code.

The implementation you get depends on the setting of the system properties named `javax.xml.parsers.SAXParserFactory` and `javax.xml.parsers.DocumentBuilderFactory`. The default values (unless overridden at runtime) point to Sun's implementation.

## **An overview of the JAXP API Packages**

**javax.xml.parsers:** The JAXP APIs, which provide a common interface for different vendors' SAX and DOM parsers.

## **An overview of the JAXP API Packages**

**javax.xml.parsers:** The JAXP APIs, which provide a common interface for different vendors' SAX and DOM parsers.

**org.w3c.dom:** Defines the **Document** class (a DOM), as well as classes for all of the components of a DOM.

## **An overview of the JAXP API Packages**

**javax.xml.parsers:** The JAXP APIs, which provide a common interface for different vendors' SAX and DOM parsers.

**org.w3c.dom:** Defines the **Document** class (a DOM), as well as classes for all of the components of a DOM.

**org.xml.sax:** Defines the basic SAX APIs.

## **An overview of the JAXP API Packages**

**javax.xml.parsers:** The JAXP APIs, which provide a common interface for different vendors' SAX and DOM parsers.

**org.w3c.dom:** Defines the **Document** class (a DOM), as well as classes for all of the components of a DOM.

**org.xml.sax:** Defines the basic SAX APIs.

**javax.xml.transform:** Defines the XSLT APIs that let you transform XML into other forms.



## **An overview of the Simple API for XML**

The "Simple API" for XML (SAX) is the event-driven, serial-access mechanism that does element-by-element processing.

## **An overview of the Simple API for XML**

The "Simple API" for XML (SAX) is the event-driven, serial-access mechanism that does element-by-element processing.

The API for this level reads and writes XML to a data repository or the Web.

## **An overview of the Simple API for XML**

The "Simple API" for XML (SAX) is the event-driven, serial-access mechanism that does element-by-element processing.

The API for this level reads and writes XML to a data repository or the Web.

For server-side and high-performance applications, you will want to fully understand this level. But for many applications, a minimal understanding will suffice.

# **An overview of the Document Object Model**

The DOM API is generally an easier API to use. It provides a relatively familiar tree structure of objects.

# **An overview of the Document Object Model**

The DOM API is generally an easier API to use. It provides a relatively familiar tree structure of objects.

You can use the DOM API to manipulate the hierarchy of application objects it encapsulates.

# **An overview of the Document Object Model**

The DOM API is generally an easier API to use. It provides a relatively familiar tree structure of objects.

You can use the DOM API to manipulate the hierarchy of application objects it encapsulates.

The DOM API is ideal for interactive applications because the entire object model is present in memory, where it can be accessed and manipulated by the user.

On the other hand, constructing the DOM requires reading the entire XML structure and holding the object tree in memory, so it is much more CPU and memory intensive.

On the other hand, constructing the DOM requires reading the entire XML structure and holding the object tree in memory, so it is much more CPU and memory intensive.

For that reason, the SAX API will tend to be preferred for server-side applications and data filters that do not require an in-memory representation of the data.



# An overview of XML Stylesheet Language Transformations

The XSLT APIs defined in `javax.xml.transform` let you write XML data to a file or convert it into other forms.

# An overview of XML Stylesheet Language Transformations

The XSLT APIs defined in `javax.xml.transform` let you write XML data to a file or convert it into other forms.

It can be used in conjunction with the SAX APIs to convert legacy (non-XML) data to XML.

## The Simple API for XML (SAX) APIs

To start the process, an instance of the `SAXParserFactory` class is used to generate an instance of the parser.

## The Simple API for XML (SAX) APIs

To start the process, an instance of the `SAXParserFactory` class is used to generate an instance of the parser.

The parser wraps a `SAXReader` object. When the parser's `parse()` method is invoked, the reader invokes one of several callback methods implemented in the application.

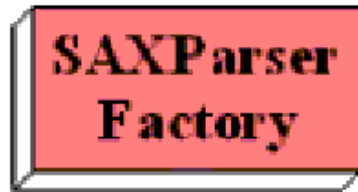
## The Simple API for XML (SAX) APIs

To start the process, an instance of the `SAXParserFactory` class is used to generate an instance of the parser.

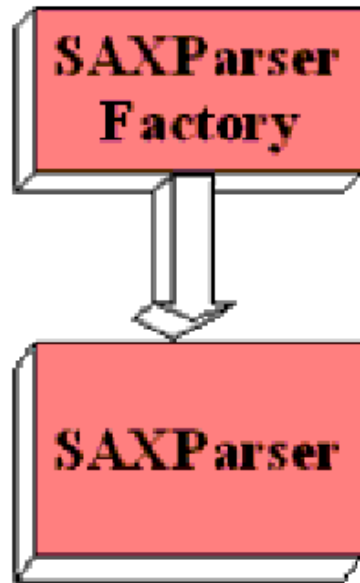
The parser wraps a `SAXReader` object. When the parser's `parse()` method is invoked, the reader invokes one of several callback methods implemented in the application.

Those methods are defined by the interfaces `ContentHandler`, `ErrorHandler`, `DTDHandler`, and `EntityResolver`.

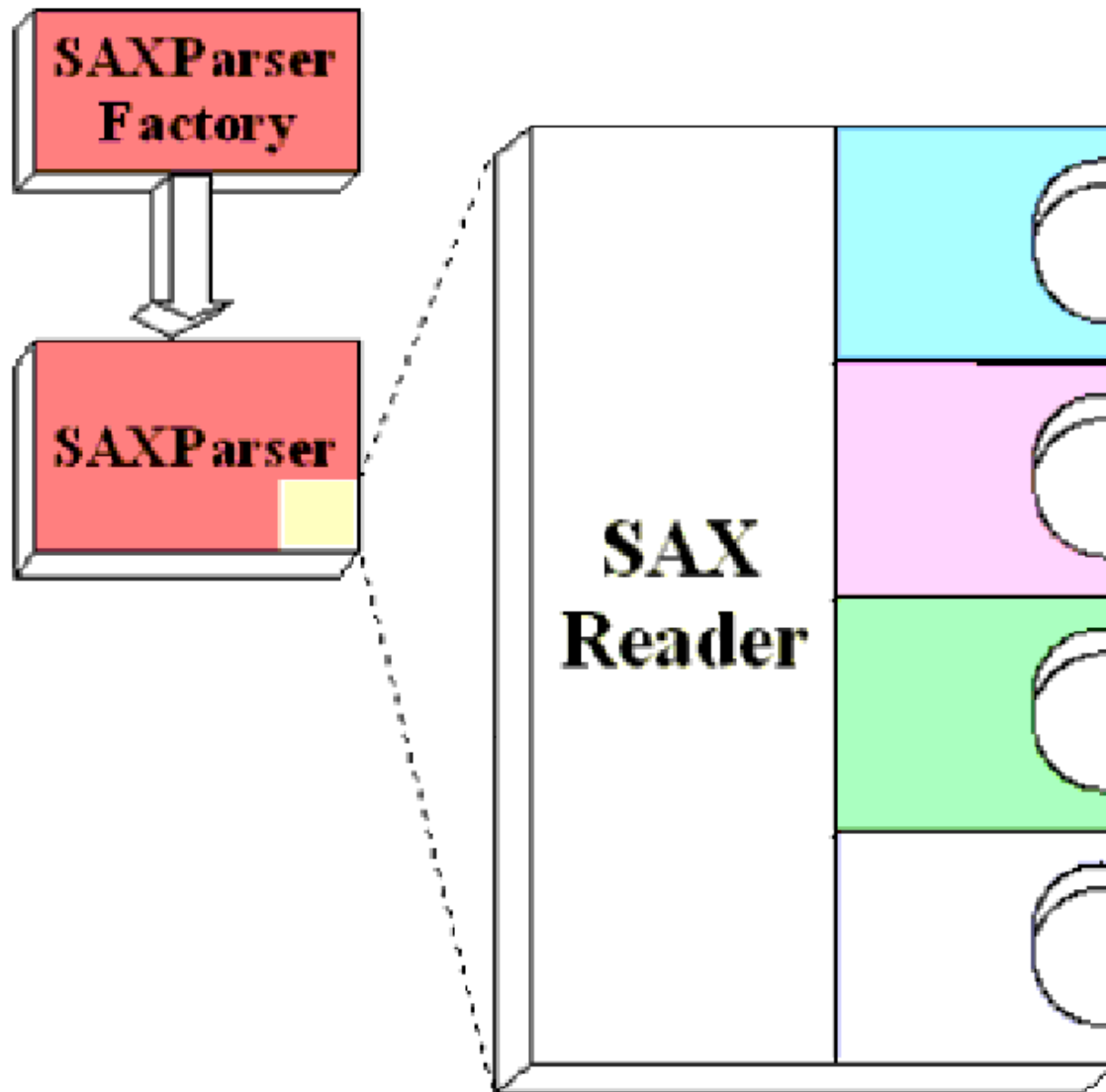
# The Simple API for XML (SAX) APIs



# The Simple API for XML (SAX) APIs

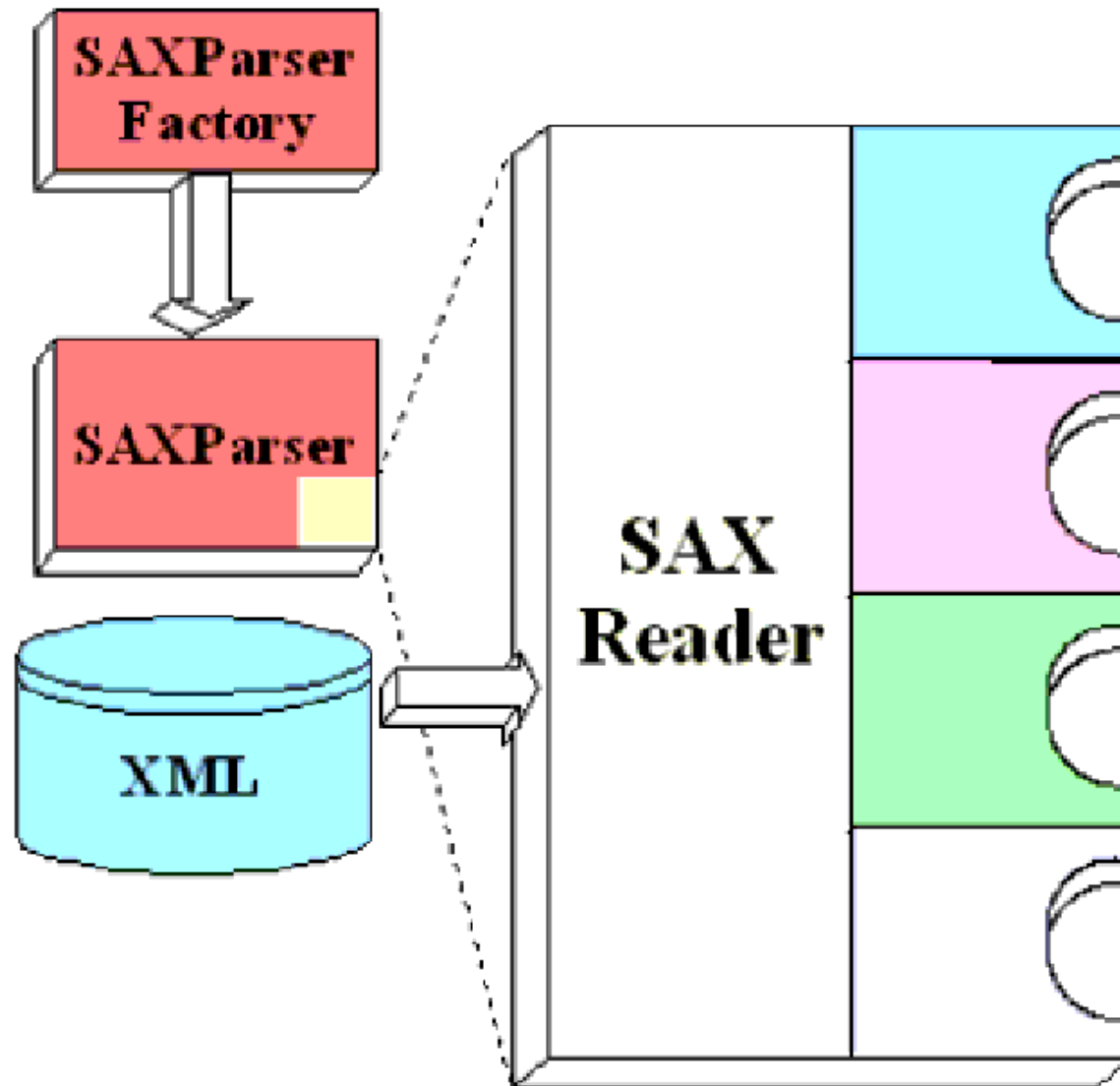


# The Simple API for XML (SAX) APIs

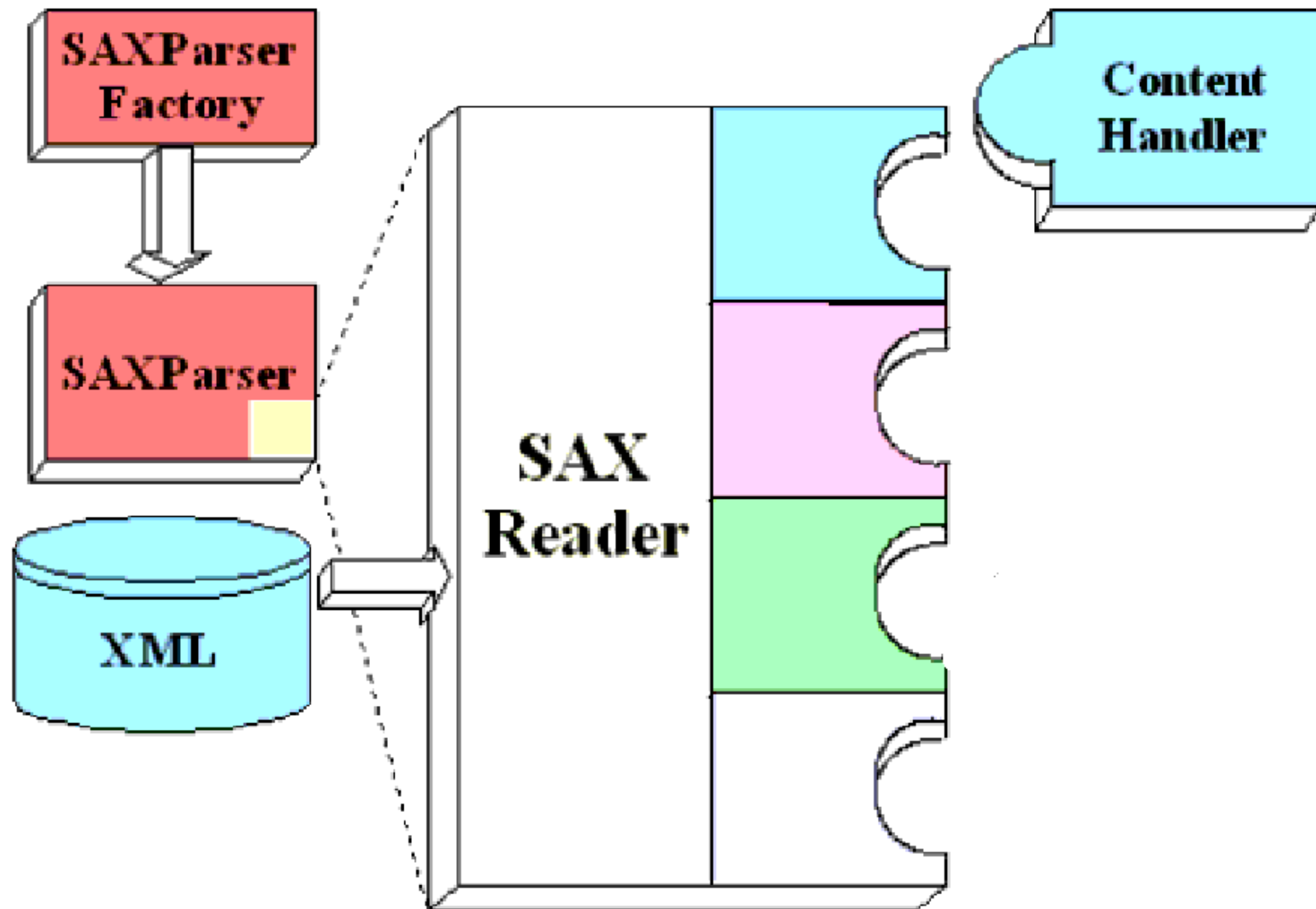




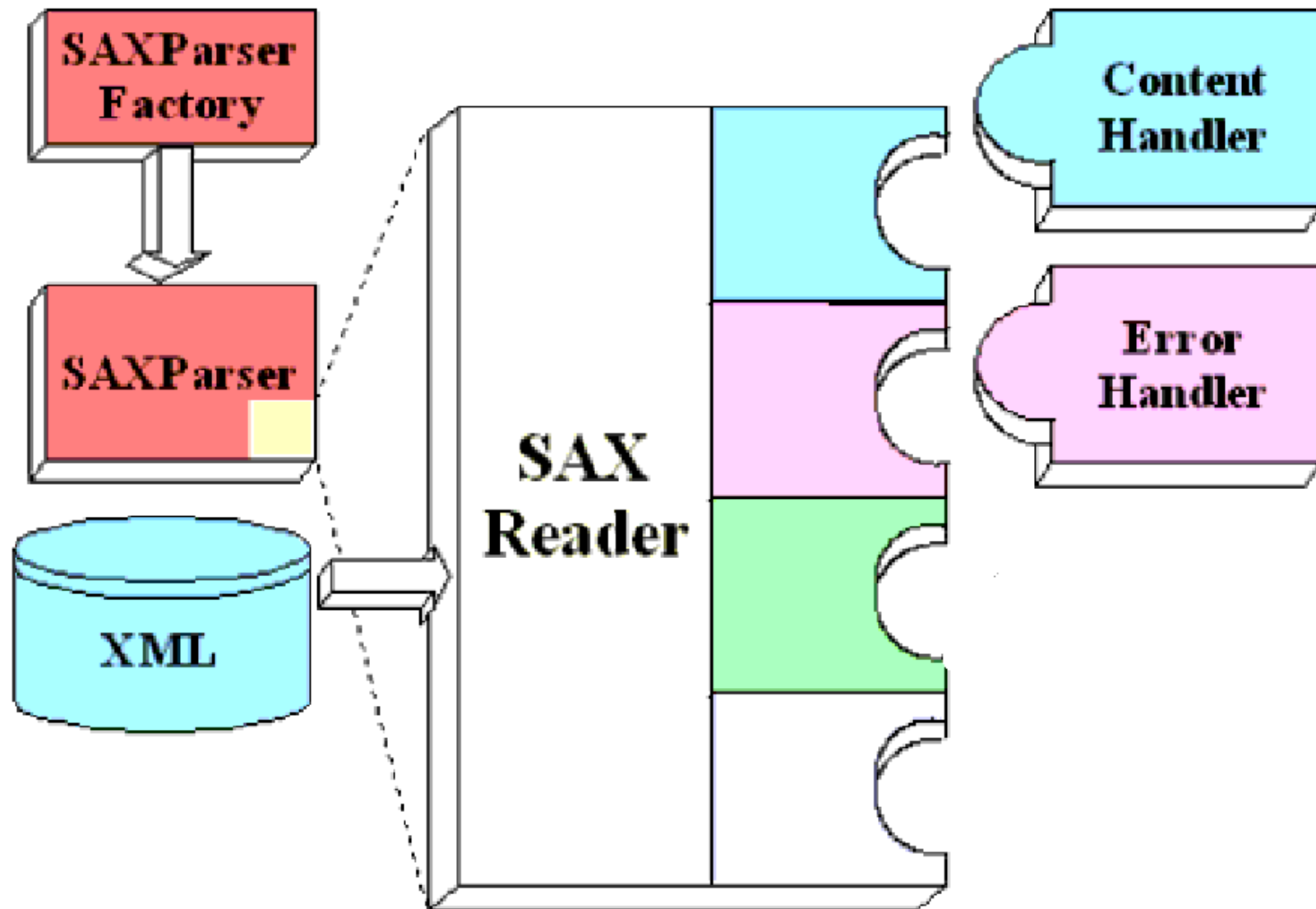
# The Simple API for XML (SAX) APIs



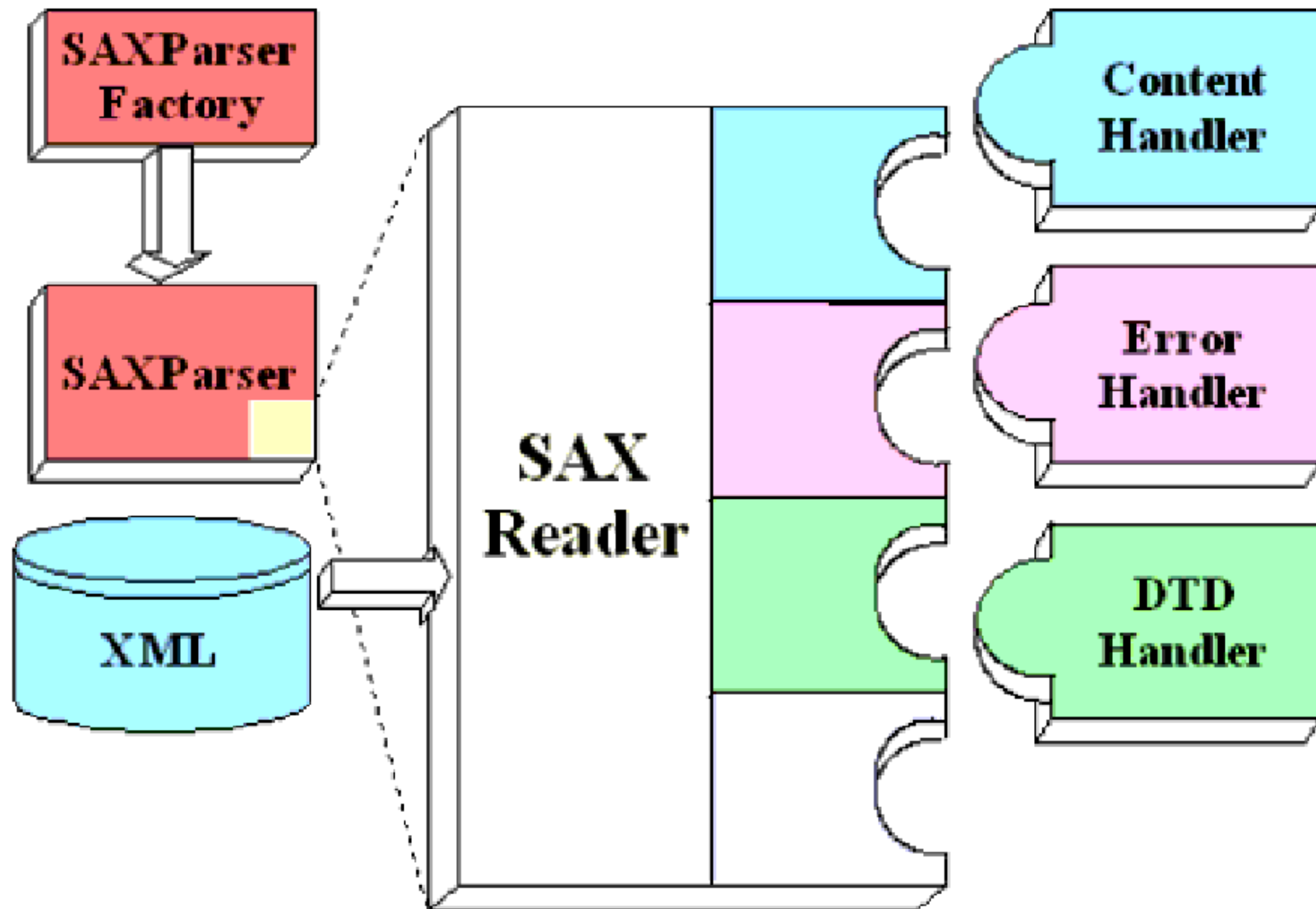
# The Simple API for XML (SAX) APIs



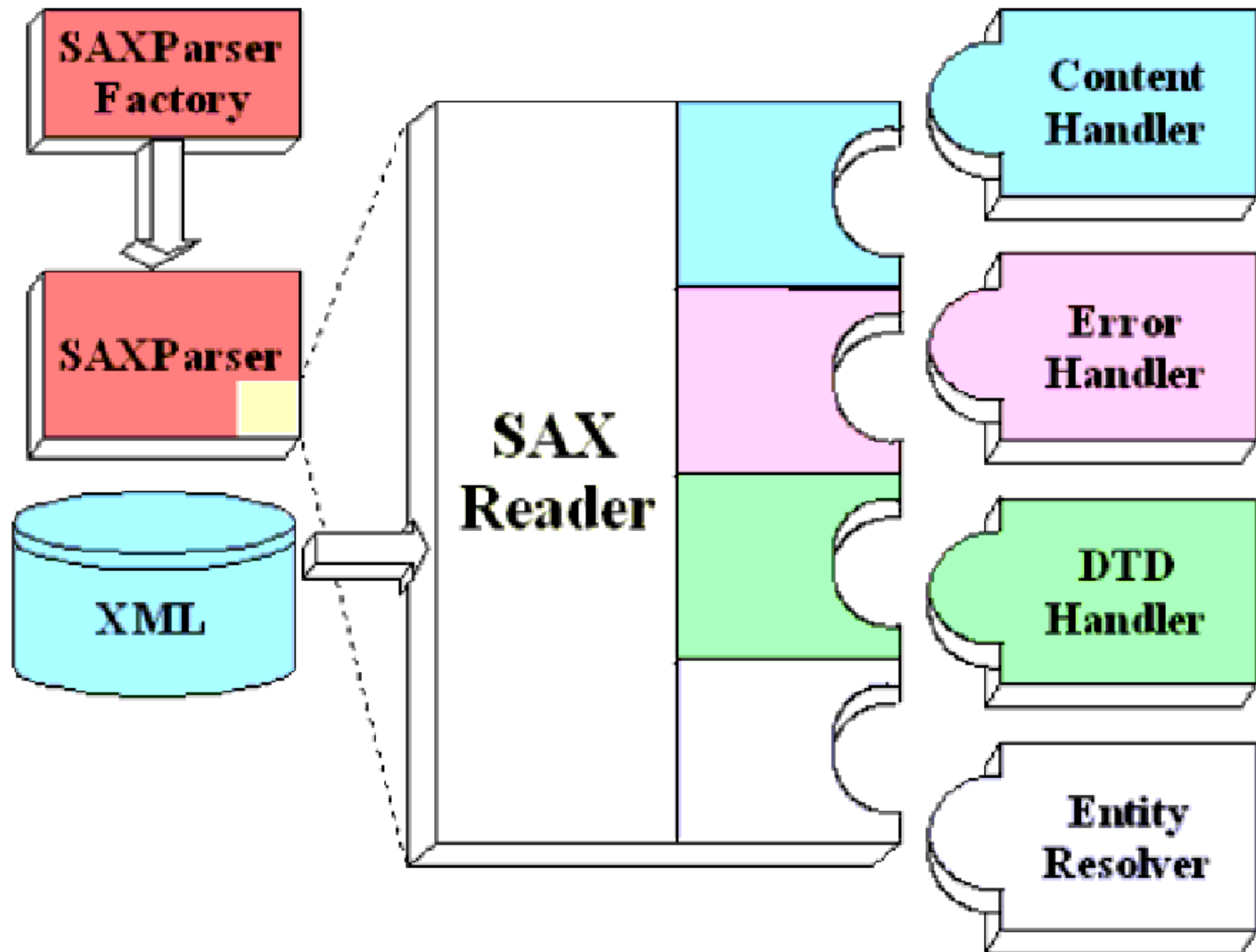
# The Simple API for XML (SAX) APIs



# The Simple API for XML (SAX) APIs



# The Simple API for XML (SAX) APIs



## Summary of the key SAX APIs

**SAXParserFactory:** A `SAXParserFactory` object creates an instance of the parser determined by the system property,  
`javax.xml.parsers.SAXParserFactory`.

## Summary of the key SAX APIs

**SAXParserFactory:** A `SAXParserFactory` object creates an instance of the parser determined by the system property,  
`javax.xml.parsers.SAXParserFactory`.

**SAXParser:** The `SAXParser` interface defines several kinds of `parse()` methods. In general, you pass an XML data source and a `DefaultHandler` object to the parser, which processes the XML and invokes the appropriate methods in the handler object.

**SAXReader:** The `SAXParser` wraps a `SAXReader`.  
It is the `SAXReader` which carries on the conversation with the SAX event handlers you define.



**SAXReader:** The `SAXParser` wraps a `SAXReader`. It is the `SAXReader` which carries on the conversation with the SAX event handlers you define.

**DefaultHandler:** A `DefaultHandler` implements the `ContentHandler`, `ErrorHandler`, `DTDHandler`, and `EntityResolver` interfaces (with null methods), so you can override only the ones you're interested in.

**ContentHandler:** Methods like `startDocument`, `endDocument`, `startElement`, and `endElement` are invoked when an XML tag is recognized. This interface also defines methods `characters` and `processingInstruction`, which are invoked when the parser encounters the text in an XML element or an inline processing instruction, respectively.

**ErrorHandler:** Methods `error`, `fatalError`, and `warning` are invoked in response to various parsing errors. The default error handler throws an exception for fatal errors and ignores other errors (including validation errors).

**ErrorHandler:** Methods `error`, `fatalError`, and `warning` are invoked in response to various parsing errors. The default error handler throws an exception for fatal errors and ignores other errors (including validation errors).

**EntityResolver:** The `resolveEntity`<sup>a</sup> method is invoked when the parser must identify data identified by a URI. In most cases, a URI is simply a URL.

---

<sup>a</sup>Entities are external unparsed character data, not XML.

## Using the SAX APIs

A typical application implements most of the `ContentHandler` methods, at a minimum.

## Using the SAX APIs

A typical application implements most of the `ContentHandler` methods, at a minimum.

Since the default implementations of the interfaces ignore all inputs except for fatal errors, a robust implementation may want to implement the `ErrorHandler` methods, as well.

# The SAX Packages

| Package                          | Description                                                        |
|----------------------------------|--------------------------------------------------------------------|
| <code>org.xml.sax</code>         | Defines the SAX interfaces.                                        |
| <code>org.xml.sax.ext</code>     | Defines SAX extensions used e.g., to process a DTD.                |
| <code>org.xml.sax.helpers</code> | Contains helper classes, e.g., default handlers with null methods. |
| <code>javax.xml.parsers</code>   | Defines the <code>SAXParserFactory</code> class.                   |

# The Document Object Model (DOM) APIs

The `javax.xml.parsers.DocumentBuilderFactory` class is used to get a `DocumentBuilder` instance which is used to produce a `Document` (a DOM) which conforms to the DOM specification.



# The Document Object Model (DOM) APIs

The `javax.xml.parsers.DocumentBuilderFactory` class is used to get a `DocumentBuilder` instance which is used to produce a `Document` (a DOM) which conforms to the DOM specification.

The builder which you get is determined by the `javax.xml.parsers.DocumentBuilderFactory` system property.

# The Document Object Model (DOM) APIs

The `javax.xml.parsers.DocumentBuilderFactory` class is used to get a `DocumentBuilder` instance which is used to produce a `Document` (a DOM) which conforms to the DOM specification.

The builder which you get is determined by the `javax.xml.parsers.DocumentBuilderFactory` system property.


This selects the factory implementation that is used to produce the builder.

You can also use the `DocumentBuilder`  
`newDocument()` method to create an empty  
`Document` that implements the  
`org.w3c.dom.Document` interface.

You can also use the `DocumentBuilder` `newDocument()` method to create an empty `Document` that implements the `org.w3c.dom.Document` interface.

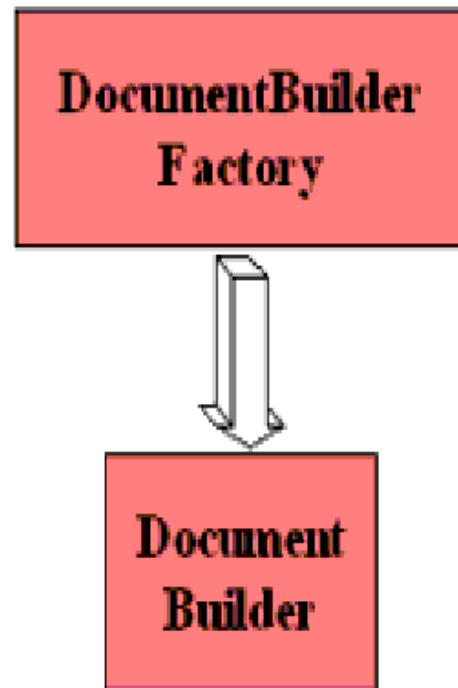
Alternatively, you can use one of the builder's `parse` methods to create a `Document` from existing XML data. The result is a DOM tree.

# The Document Object Model (DOM) APIs

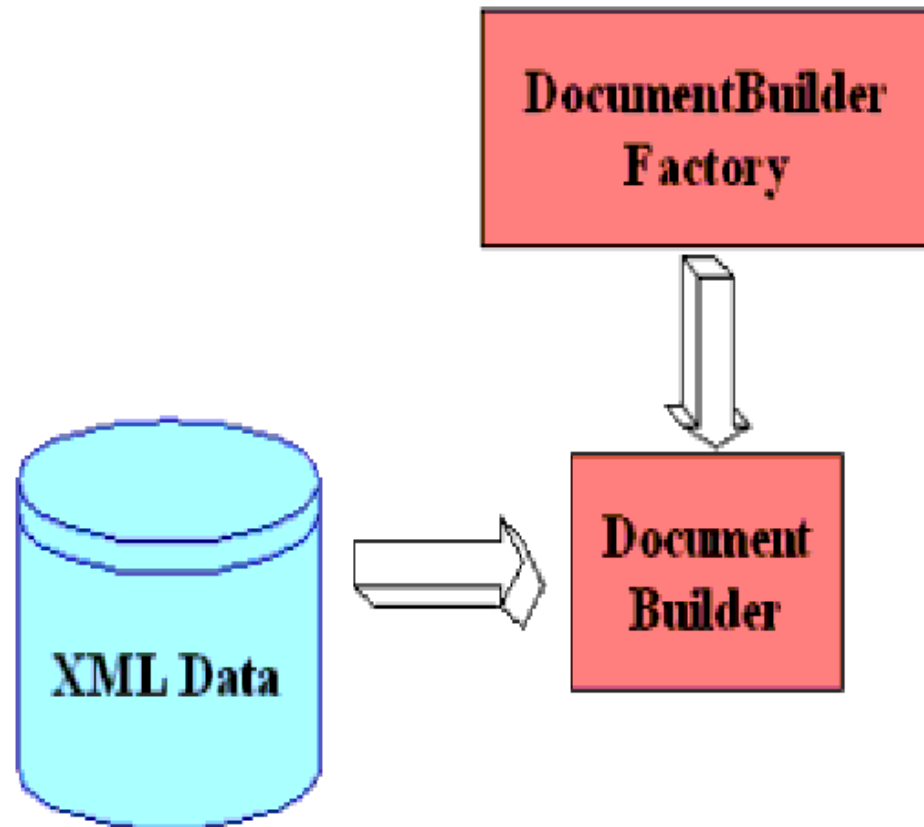


**DocumentBuilder**  
**Factory**

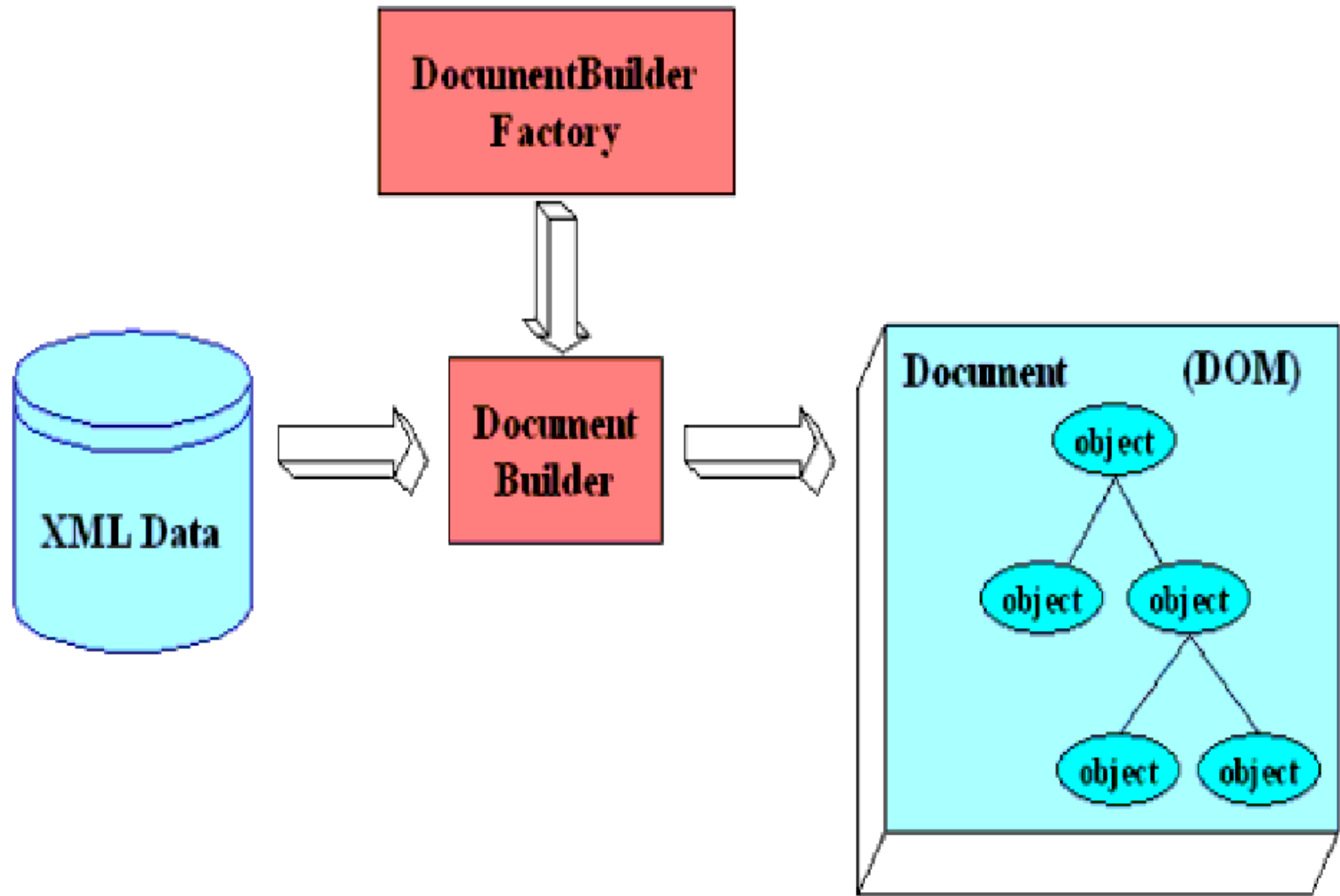
# The Document Object Model (DOM) APIs



# The Document Object Model (DOM) APIs



# The Document Object Model (DOM) APIs





# The DOM Packages

| Package                        | Description                                                                                                                                                                        |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>org.w3c.dom</code>       | Defines the DOM programming interfaces for XML.                                                                                                                                    |
| <code>javax.xml.parsers</code> | Defines the <code>DocumentBuilderFactory</code> class and the <code>DocumentBuilder</code> class, which returns an object that implements the W3C <code>Document</code> interface. |

## The XML Stylesheet Language for Transformation (XSLT) APIs

A `TransformerFactory` object is instantiated, and used to create a `Transformer`. The source object is the input to the transformation process. A source object can be created from a SAX reader, from a DOM, or from an input stream.

# The XML Stylesheet Language for Transformation (XSLT) APIs

A `TransformerFactory` object is instantiated, and used to create a `Transformer`. The source object is the input to the transformation process. A source object can be created from a SAX reader, from a DOM, or from an input stream.

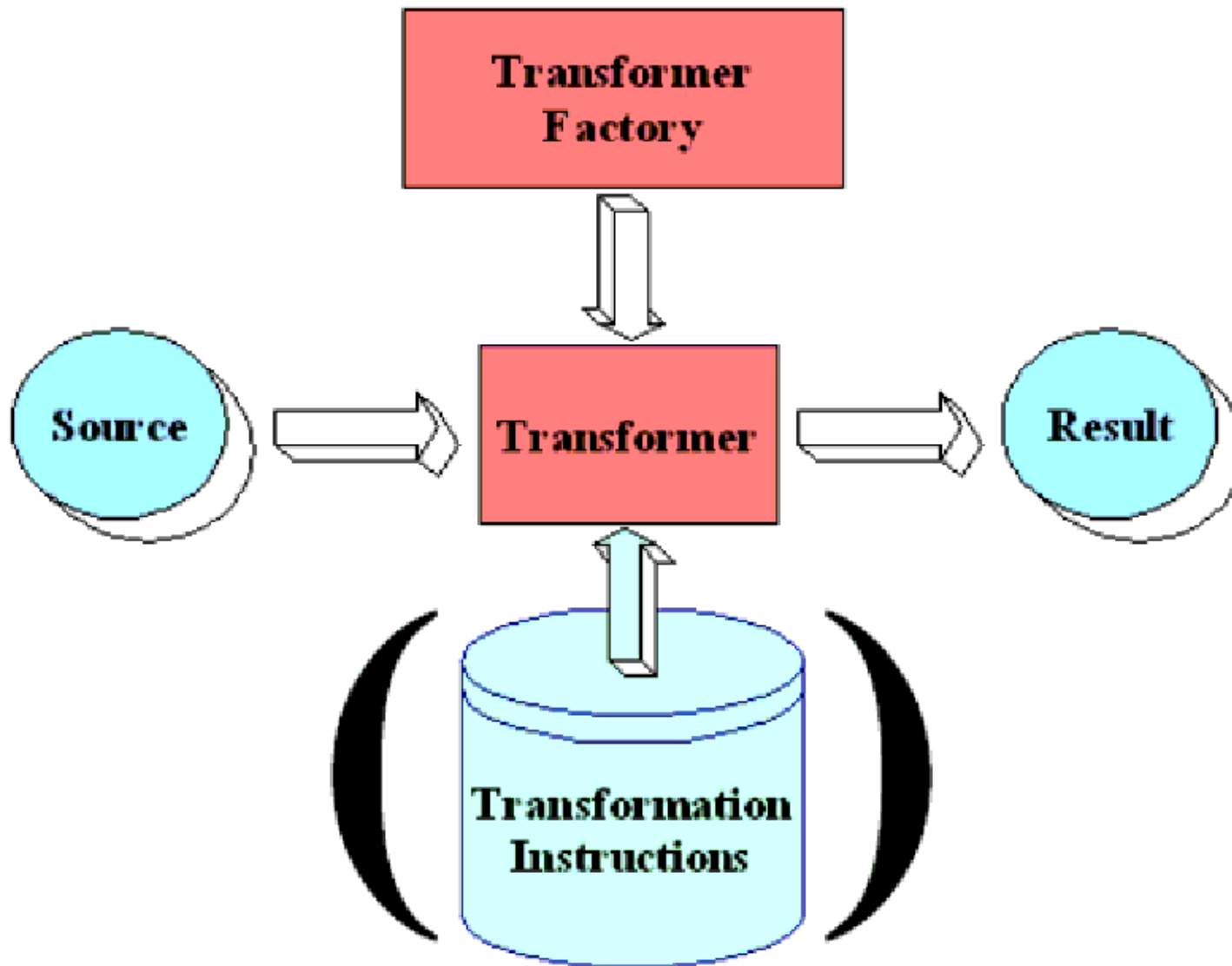
Similarly, the result object is the result of the transformation process. That object can be a SAX event handler, a DOM, or an output stream.

When the transformer is created, it may be created from a set of **transformation instructions**, in which case the specified transformations are carried out.

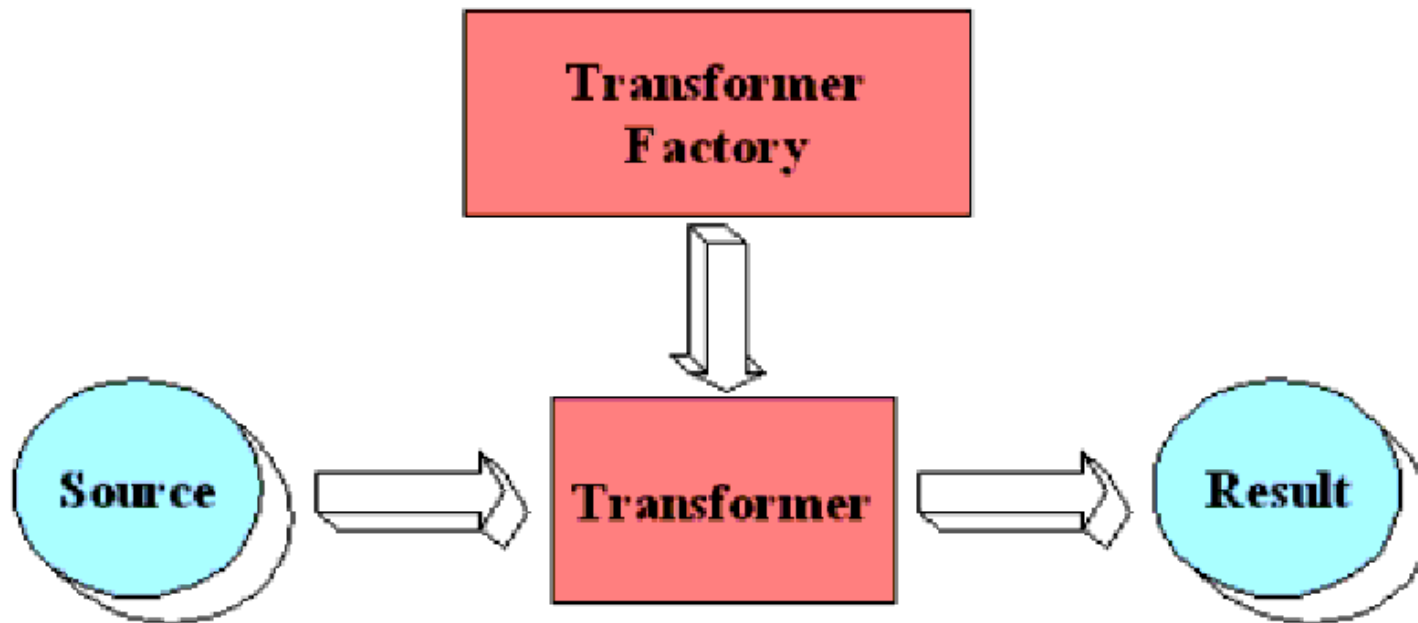
When the transformer is created, it may be created from a set of **transformation instructions**, in which case the specified transformations are carried out.

If it is created without any specific instructions, then the transformer object simply copies the source to the result.

# The XML Stylesheet Language for Transformation (XSLT) APIs



# The XML Stylesheet Language for Transformation (XSLT) APIs



# The XSLT Packages

| Package                          | Description                                                                                                                                                                                                                                               |
|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>javax.xml.transform</code> | Defines the <code>TransformerFactory</code> and <code>Transformer</code> classes, which you use to get a object capable of doing transformations and invoke its <code>transform()</code> method, providing it with an input (source) and output (result). |



## The XSLT Packages (continued)

| Package                              | Description                                                                                              |
|--------------------------------------|----------------------------------------------------------------------------------------------------------|
| <code>javax.xml.transform.dom</code> | Classes to create input (source) and output (result) objects from a DOM.                                 |
| <code>javax.xml.transform.sax</code> | Classes to create input (source) from a SAX parser and output (result) objects from a SAX event handler. |

## The XSLT Packages (continued)

| Package                                           | Description                                                                      |
|---------------------------------------------------|----------------------------------------------------------------------------------|
| <code>javax.xml.<br/>transform.<br/>stream</code> | Classes to create input (source) and output (result) objects from an I/O stream. |