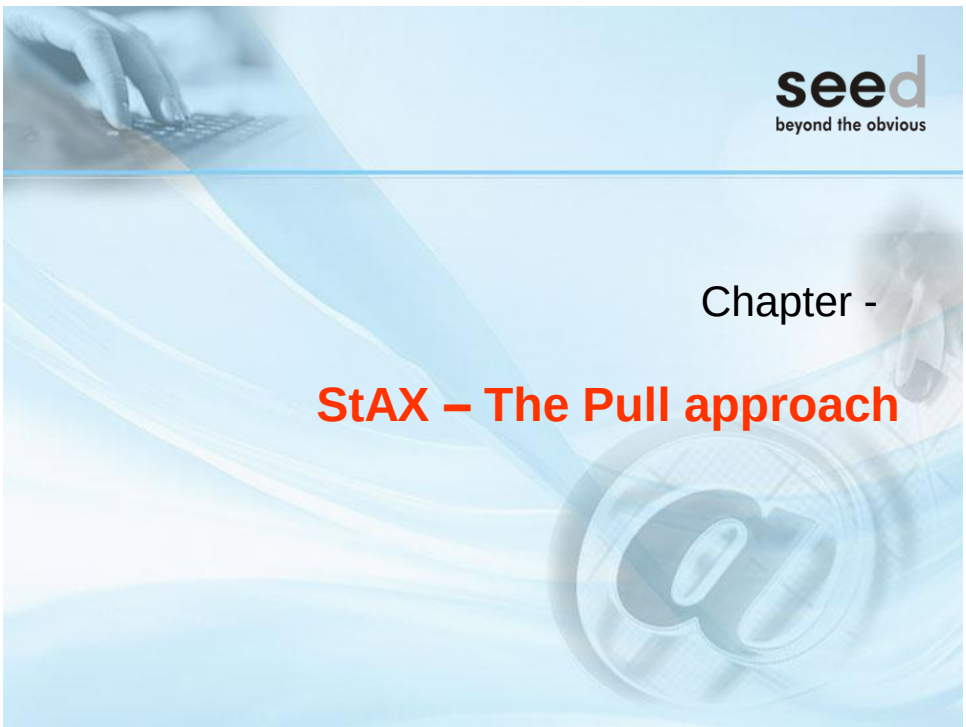

Objectives

- At the end of this chapter you will be able to understand
  - What STAX is
  - StAX Cursor API and Iterator API

Introduction

- Under the pull approach, events are pulled from an XML document under the control of the application using the parser.
- StAX (Simple transformation Api for Xml)is similar to the SAX API in that both offer event-based APIs. However, StAX differs from the SAX API in the following respects:
  - Unlike in the SAX API, in the StAX API, it is the application rather than the parser that controls the delivery of the parsing events. StAX offers two event-based APIs: a *cursor-based API* and an *iterator-based API*, both of which are under the application's control.
  - The cursor API allows a walk-through of the document in document order and provides the lowest level of access to all the structural and content information within the document.

Introduction

  - The iterator API is similar to the cursor API but instead of providing low-level access, it provides access to the structural and content information in the form of event objects.
  - Unlike the SAX API, the StAX API can be used both for reading and for writing XML documents.

Cursor API

- Key points about the StAX cursor API are as follows:
  - The *XMLStreamReader* interface is the main interface for parsing an XML document. We can use this interface to scan an XML document's structure and contents using the next() and hasNext() methods.
  - The next() method returns an integer token for the next parse event
  - Depending on the next event type, we can call specific allowed methods on the XMLStreamReader interface
  - Following Table lists various event types and the corresponding allowed methods.

Cursor API

Event Type	Allowed Methods
Any event type	getProperty(), hasNext(), require(), close(), getNamespaceURI(), isStartElement(), isEndElement(), isCharacters(), isWhiteSpace(), getNamespaceContext(), getEventType(), getLocation(), hasText(), hasName()
START_ELEMENT	next(), getName(), getLocalName(), hasName(), getPrefix(), getAttributeXXX(), isAttributeSpecified(), getNamespaceXXX(), getElementText(), nextTag()
ATTRIBUTE	next(), nextTag(), getAttributeXXX(), isAttributeSpecified()
NAMESPACE	next(), nextTag(), getNamespaceXXX()
END_ELEMENT	next(), getName(), getLocalName(), hasName(), getPrefix(), getNamespaceXXX(), nextTag()
CHARACTERS	next(), getTextXXX(), nextTag()
CDATA	next(), getTextXXX(), nextTag()
COMMENT	next(), getTextXXX(), nextTag()
SPACE	next(), getTextXXX(), nextTag()
START_DOCUMENT	next(), getEncoding(), getVersion(), isStandalone(), standaloneSet(), getCharacterEncodingScheme(), nextTag()
END_DOCUMENT	close()
PROCESSING_INSTRUCTION	next(), getPITarget(), getPIData(), nextTag()
ENTITY_REFERENCE	next(), getLocalName(), getText(), nextTag()
DTD	next(), getText(), nextTag()

Iterator API

- Key points about the StAX Iterator API are as follows:
  - The *XMLEventReader* interface is the main interface for parsing an XML document. We can use this interface to iterate over an XML document's structure and contents using the nextEvent() and hasNext() methods.
  - The nextEvent() method returns an XMLEvent object.
  - The XMLEvent interface provides utility methods for determining the next event type and for processing it appropriately.

Stax API

- The StAX API is recommended for data-binding applications, specifically for the marshaling and unmarshaling of an XML document during the bidirectional XML-to-Java mapping process.
- A StAX API implementation is included in J2SE 6.0.

Comparing the Parsing Approaches

Parsing Approach	Advantages	Disadvantages	Suitable Application
DOM	Ease of use, navigation, random access, and XPath support	Must parse entire document, memory intensive	Applications that modify structure and content of an XML document, such as visual XML editors*
SAX	Low memory consumption, efficient	No navigation, no random access, no modification	Read-only XML applications, such as document validation
StAX	Ease of use, low memory consumption, application regulates parsing, filtering	No random access, no modification	Data binding, SOAP message processing

Parsing with StAX

- In the push model, the parser generates events as the XML document is parsed.
- With the pull parsing in StAX, the application generates the parse events; thus, we can generate parse events as required
- The StAX API (JSR-173) is implemented in J2SE 6.0

Parsing with StAX

- Key points about StAX API are as follows:
  - The StAX API classes are in the *javax.xml.stream* and *javax.xml.stream.events* packages
  - The StAX API offers two different APIs for parsing an XML document: a cursor-based API and an iterator-based API.
  - The *XMLStreamReader* interface parses an XML document using the cursor API.
  - *XMLEventReader* parses an XML document using the iterator API.
  - We can use the *XMLStreamWriter* interface to generate an XML document.

Cursor API

- We can use the *XMLStreamReader* object to parse an XML document using the cursor approach.
- The next() method generates the next parse event. We can obtain the event type from the *getEventType()* method
- We can create an XMLStreamReader object from an *XMLInputFactory* object, and we can create an *XMLInputFactory* object using the static method *newInstance()*

Cursor API

- **Creating an XMLStreamReader Object**

```
XMLInputFactory
inputFactory=XMLInputFactory.newInstance();
InputStream input=new FileInputStream(new
File("catalog.xml"));
XMLStreamReader xmlStreamReader =
inputFactory.createXMLStreamReader(input);
```


Cursor API

- **Obtaining a Parsing Event**

```
while (xmlStreamReader.hasNext()) {
int event = xmlStreamReader.next();
}
```

- The next() method returns an int, which corresponds to a parsing event, as specified by an *XMLStreamConstants* constant
- Following Table lists the event types returned by the XMLStreamReader object

Cursor API

Event Type	Description
START_DOCUMENT	Start of a document
START_ELEMENT	Start of an element
ATTRIBUTE	An element attribute
NAMESPACE	A namespace declaration
CHARACTERS	Characters may be text or whitespace
COMMENT	A comment
SPACE	Ignorable whitespace
PROCESSING_INSTRUCTION	Processing instruction
DTD	A DTD
ENTITY_REFERENCE	An entity reference
CDATA	CDATA section
END_ELEMENT	End element
END_DOCUMENT	End document
ENTITY_DECLARATION	An entity declaration
NOTATION_DECLARATION	A notation declaration

Cursor API

- Demo:- .\eclipse-workspace-stax

Iterator API

- The *XMLEventReader* object parses an XML document with an object event iterator and generates an *XMLEvent* object for each parse event.
- To create an XMLEventReader object, we need to first create an *XMLInputFactory* object with the static method *newInstance()* and then obtain an XMLEventReader object from the XMLInputFactory object with the *createXMLEventReader* method

```
XMLInputFactory inputFactory=XMLInputFactory.newInstance();
InputStream input=new FileInputStream(new
File("catalog.xml"));
XMLEventReader xmlEventReader =
inputFactory.createXMLEventReader(input);
```


seed
beyond the obvious

Iterator API

- An *XMLEvent* object represents an XML document event in StAX.
- We obtain the next event with the *nextEvent()* method of an *XMLEventReader* object.
- The *getEventType()* method of an *XMLEventReader* object returns the event type

```
XMLEvent event=xmlEventReader.nextEvent();
int eventType=event.getEventType();
```

- The event types listed in previous Table for an *XMLStreamReader* object are also the event types generated with an *XMLEventReader* object.
- The *isXXX()* methods in the *XMLEventReader* interface return a boolean if the event is of the type corresponding to the *isXXX()* method.

seed
beyond the obvious

Quick Recap . . .

- The pull approach provides complete control to an application over how the document parse events are processed, and provides a cursor-based approach and an iterator-based approach to control the flow of parse events.
- This approach is best suited for processing XML content that is being streamed over a network connection. Also, this API is useful for marshaling and unmarshaling XML documents from and to Java types.
- Major areas of applications for this API include web services–related message processing and XML-to-Java binding.
- The StAX API included in J2SE 6.0.