## DOM (Document Object Model) Parsing:

DOM (Document Object Model) Parsing হলো XML ডেটা পার্সিংয়ের একটি পদ্ধতি যা XML ডকুমেন্টটিকে একটি গাছ (Tree) স্ট্রাকচারে রূপান্তরিত করে। এই পদ্ধতি সম্পূর্ণ XML ডকুমেন্টকে মেমোরিতে লোড করে এবং এরপর ডেটার উপর কাজ করা হয়। DOM পদ্ধতি ছোট থেকে মাঝারি আকারের XML ডকুমেন্টের জন্য ভালো।

## DOM Parsing এর মূল বৈশিষ্ট্য

1. **পূর্ণ ডকুমেন্ট লোডিং:** DOM পদ্ধতিতে XML ডকুমেন্ট সম্পূর্ণভাবে মেমোরিতে লোড করা হয়। এর ফলে আপনি XML ডকুমেন্টের যেকোনো অংশে সরাসরি অ্যাক্সেস পেতে পারেন এবং সম্পূর্ণ ডকুমেন্ট নিয়ে কাজ করতে পারেন।
2. **ট্রি স্ট্রাকচার:** XML ডকুমেন্টটি একটি ট্রি স্ট্রাকচারে রূপান্তরিত হয়। ট্রির মূল (Root) এলিমেন্ট থাকে এবং এর অধীনে অন্যান্য এলিমেন্ট এবং টেক্সট নোডস থাকে। প্রতিটি এলিমেন্টের সাথে এর অ্যাট্রিবিউট এবং চাইল্ড এলিমেন্টস থাকে।
3. **নেভিগেশন ও মডিফিকেশন:** DOM API ব্যবহার করে আপনি XML ডকুমেন্টের মধ্যে নেভিগেট করতে পারেন এবং XML ডেটা পরিবর্তন করতে পারেন। নতুন এলিমেন্ট যোগ করা, বিদ্যমান এলিমেন্ট আপডেট করা বা মুছে ফেলা সম্ভব।
4. **মেমরি ব্যবহারে গুরুতর:** কারণ সম্পূর্ণ XML ডকুমেন্ট মেমোরিতে লোড হয়, বড় XML ফাইলের ক্ষেত্রে মেমরি ব্যবহারে সমস্যা হতে পারে।

## DOM Parsing এর উদাহরণ

নিচে একটি উদাহরণ দেওয়া হলো যেটি XML ডকুমেন্টের ডেটা পড়তে এবং প্রিন্ট করতে DOM API ব্যবহার করে।

**XML ডকুমেন্ট (example.xml):**

```
<?xml version="1.0"?>
<employees>
    <employee id="1">
        <name>John Doe</name>
        <age>30</age>
        <position>Developer</position>
    </employee>
    <employee id="2">
        <name>Jane Smith</name>
        <age>25</age>
        <position>Designer</position>
    </employee>
</employees>
```

**DOM Parsing কোড:**

```
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

public class DOMParserExample {
    public static void main(String[] args) {
        try {
            // DocumentBuilderFactory অবজেক্ট তৈরি করা
            DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
```

```
            // DocumentBuilder অবজেক্ট তৈরি করা
            DocumentBuilder builder = factory.newDocumentBuilder();

            // XML ফাইল পার্স করা
            Document doc = builder.parse("example.xml");

            // ডকুমেন্টের রুট এলিমেন্ট পাওয়া
            Element root = doc.getDocumentElement();
            System.out.println("Root Element: " + root.getNodeName());

            // 'employee' এলিমেন্টের সমস্ত উদাহরণ সংগ্রহ করা
            NodeList employees = doc.getElementsByTagName("employee");
            for (int i = 0; i < employees.getLength(); i++) {
                Element employee = (Element) employees.item(i);

                // 'employee' এলিমেন্টের অ্যাট্রিবিউট এবং চাইল্ড এলিমেন্ট পড়া
                String id = employee.getAttribute("id");
                String name =
employee.getElementsByTagName("name").item(0).getTextContent();
                String age =
employee.getElementsByTagName("age").item(0).getTextContent();
                String position =
employee.getElementsByTagName("position").item(0).getTextContent();

                System.out.println("Employee ID: " + id);
                System.out.println("Name: " + name);
                System.out.println("Age: " + age);
                System.out.println("Position: " + position);
                System.out.println();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## কোডের ব্যাখ্যা:

1. **Import Statements:**

   ```
   import javax.xml.parsers.DocumentBuilder;
   import javax.xml.parsers.DocumentBuilderFactory;
   import org.w3c.dom.Document;
   import org.w3c.dom.Element;
   import org.w3c.dom.NodeList;
   ```

   - এই লাইনের মধ্যে বিভিন্ন XML পার্সিং এবং DOM API ক্লাসগুলো আমদানি করা হয়েছে। `DocumentBuilderFactory`, `DocumentBuilder`, `Document`, `Element`, এবং `NodeList` হল XML ডকুমেন্ট পার্সিং এর জন্য প্রয়োজনীয় ক্লাস।

2. **DocumentBuilderFactory এবং DocumentBuilder** তৈরি:

   ```
   DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
   DocumentBuilder builder = factory.newDocumentBuilder();
   ```

   - `DocumentBuilderFactory` একটি ফ্যাক্টরি ক্লাস যা `DocumentBuilder` অবজেক্ট তৈরির জন্য ব্যবহৃত হয়। `DocumentBuilder` অবজেক্ট XML ডকুমেন্ট পার্স করার জন্য ব্যবহৃত হয়।

3. **XML** ফাইল পার্স করা:

   ```
   Document doc = builder.parse("example.xml");
   ```

   - `builder.parse("example.xml")` পদ্ধতি XML ফাইলটি পার্স করে এবং `Document` অবজেক্টে রূপান্তরিত করে।

4. **রুট এলিমেন্ট পাওয়া:**

```
Element root = doc.getDocumentElement();
System.out.println("Root Element: " + root.getNodeName());
```

   - `getDocumentElement()` পদ্ধতি XML ডকুমেন্টের রুট এলিমেন্ট (যেমন `<employees>`) ফেরত দেয়। `getNodeName()` পদ্ধতি রুট এলিমেন্টের নাম প্রিন্ট করে।

5. '**employee**' এলিমেন্টের সমস্ত উদাহরণ সংগ্রহ করা:

```
NodeList employees = doc.getElementsByTagName("employee");
for (int i = 0; i < employees.getLength(); i++) {
    Element employee = (Element) employees.item(i);
```

   - `getElementsByTagName("employee")` পদ্ধতি XML ডকুমেন্টের সব `employee` এলিমেন্ট সংগ্রহ করে। `NodeList` ক্লাস একটি লিস্ট ধারণ করে যেখানে প্রতিটি আইটেম একটি XML এলিমেন্ট।

6. '**employee**' এলিমেন্টের অ্যাট্রিবিউট এবং চাইল্ড এলিমেন্ট পড়া:

```
String id = employee.getAttribute("id");
String name =
employee.getElementsByTagName("name").item(0).getTextContent();
String age =
employee.getElementsByTagName("age").item(0).getTextContent();
String position =
employee.getElementsByTagName("position").item(0).getTextContent();
```

   - `getAttribute("id")` পদ্ধতি `employee` এলিমেন্টের `id` অ্যাট্রিবিউটের মান ফেরত দেয়।
   - `getElementsByTagName("name")` পদ্ধতি `name` এলিমেন্টের মান সংগ্রহ করে, একইভাবে `age` এবং `position` এলিমেন্টের মানও সংগ্রহ করা হয়।

7. আউটপুট প্রিন্ট করা:

```
System.out.println("Employee ID: " + id);
System.out.println("Name: " + name);
System.out.println("Age: " + age);
System.out.println("Position: " + position);
System.out.println();
```

   - প্রতিটি `employee` এলিমেন্টের `id`, `name`, `age`, এবং `position` প্রিন্ট করা হয়।

8. **Exception Handling:**

```
catch (Exception e) {
    e.printStackTrace();
}
```

   - যদি কোন সমস্যা হয় (যেমন XML ফাইল পাওয়া না গেলে), তেমন একটি এক্সেপশন ক্যাচ করা হয় এবং এর স্ট্যাক ট্রেস প্রিন্ট করা হয়।

## সারাংশ:

এই কোডটি XML ডকুমেন্ট থেকে `employee` এলিমেন্টের তথ্য পড়ে এবং এটি কনসোলে প্রিন্ট করে। DOM পদ্ধতি ব্যবহারের মাধ্যমে XML ডকুমেন্ট সম্পূর্ণভাবে মেমোরিতে লোড হয়, যার ফলে XML ডেটার বিভিন্ন অংশে সহজেই অ্যাক্সেস পাওয়া যায় এবং এটি পরিবর্তন করা যায়।

**SAX (Simple API for XML)** পার্সিং হলো একটি ইভেন্ট-বেসড পদ্ধতি যা XML ডকুমেন্ট পার্স করতে ব্যবহৃত হয়। SAX পার্সিং মডেল একটি স্ট্রিমিং পদ্ধতি ব্যবহার করে, যেখানে XML ডকুমেন্টটি একে একে পড়া হয় এবং প্রতি ইভেন্টের জন্য একটি কলব্যাক মেথড চালানো হয়। এটি মেমোরি-এফিশিয়েন্ট এবং বড় XML ডকুমেন্ট পার্স করার জন্য কার্যকর।

## SAX Parsing এর মূল বৈশিষ্ট্য

1. ইভেন্ট-বেসড পার্সিং: SAX পার্সিং ইভেন্ট-ভিত্তিক, অর্থাৎ XML ডকুমেন্টের প্রতি অংশ পড়ার সময় বিভিন্ন ইভেন্ট (যেমন এলিমেন্টের শুরু, এলিমেন্টের শেষ, টেক্সট কন্টেন্ট) ঘটে এবং প্রত্যেকটি ইভেন্টের জন্য নির্ধারিত কলব্যাক মেথড চালানো হয়।
2. মেমোরি-এফিশিয়েন্ট: SAX পুরো XML ডকুমেন্ট মেমোরিতে লোড করে না। বরং এটি স্ট্রিমিং পদ্ধতি ব্যবহার করে, ফলে বড় XML ডকুমেন্ট পার্স করার সময় মেমরি ব্যবহারের সমস্যা কম হয়।
3. **ডেটা মডিফিকেশন অক্ষম:** SAX শুধুমাত্র ডেটা পড়তে ব্যবহৃত হয় এবং XML ডকুমেন্টের ডেটা পরিবর্তন করা যায় না।

## SAX Parsing উদাহরণ

নিচে SAX পার্সিং ব্যবহার করে একটি XML ডকুমেন্ট পার্স করার উদাহরণ দেয়া হয়েছে।

**XML ডকুমেন্ট (example.xml):**

```
<?xml version="1.0"?>
<employees>
    <employee id="1">
        <name>John Doe</name>
        <age>30</age>
        <position>Developer</position>
    </employee>
    <employee id="2">
        <name>Jane Smith</name>
        <age>25</age>
        <position>Designer</position>
    </employee>
</employees>
```

**SAX Parsing কোড:**

```
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

public class SAXParserExample {
    public static void main(String[] args) {
        try {
            // SAXParserFactory অবজেক্ট তৈরি করা
            SAXParserFactory factory = SAXParserFactory.newInstance();

            // SAXParser অবজেক্ট তৈরি করা
            SAXParser saxParser = factory.newSAXParser();

            // DefaultHandler অবজেক্ট তৈরি করা
            DefaultHandler handler = new DefaultHandler() {
                // এলিমেন্টের শুরুতে কলব্যাক মেথড
                @Override
                public void startElement(String uri, String localName, String
qName, Attributes attributes) throws SAXException {
                    if (qName.equalsIgnoreCase("employee")) {
```

```
                        System.out.println("Start Element : employee");
                        System.out.println("Employee ID: " +
attributes.getValue("id"));
                    }
                }

                // এলিমেন্টের মধ্যে টেক্সট কন্টেন্টে কলব্যাক মেথড
                @Override
                public void characters(char[] ch, int start, int length) throws
SAXException {
                    String content = new String(ch, start, length).trim();
                    if (!content.isEmpty()) {
                        System.out.println("Content: " + content);
                    }
                }

                // এলিমেন্টের শেষ দিকে কলব্যাক মেথড
                @Override
                public void endElement(String uri, String localName, String
qName) throws SAXException {
                    if (qName.equalsIgnoreCase("employee")) {
                        System.out.println("End Element : employee");
                    }
                }
            };

            // XML ফাইল পার্স করা
            saxParser.parse("example.xml", handler);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## কোডের অংশবিশেষ ব্যাখ্যা:

1. **Import Statements:**

   ```
   import javax.xml.parsers.SAXParser;
   import javax.xml.parsers.SAXParserFactory;
   import org.xml.sax.Attributes;
   import org.xml.sax.SAXException;
   import org.xml.sax.helpers.DefaultHandler;
   ```

   - SAX পার্সিং এবং ইভেন্ট হ্যান্ডলিংয়ের জন্য প্রয়োজনীয় ক্লাসগুলো আমদানি করা হয়েছে। `SAXParser`, `SAXParserFactory`, `Attributes`, `SAXException`, এবং `DefaultHandler` এর মধ্যে উল্লেখযোগ্য।

2. **SAXParserFactory এবং SAXParser তৈরি**

   ```
   SAXParserFactory factory = SAXParserFactory.newInstance();
   SAXParser saxParser = factory.newSAXParser();
   ```

   - `SAXParserFactory` একটি ফ্যাক্টরি ক্লাস যা `SAXParser` অবজেক্ট তৈরির জন্য ব্যবহৃত হয়। `SAXParser` অবজেক্ট XML ডকুমেন্ট পার্স করার জন্য ব্যবহৃত হয়।

3. **DefaultHandler তৈরি**

   ```
   DefaultHandler handler = new DefaultHandler() {
       // ইভেন্ট হ্যান্ডলিং মেথডগুলি এখানে সংজ্ঞায়িত করা হয়
   };
   ```

   - `DefaultHandler` একটি ক্লাস যা SAX ইভেন্টগুলির জন্য ডিফল্ট হ্যান্ডলার সরবরাহ করে। আমরা এটিকে ইভেন্ট-ভিত্তিক কলব্যাক মেথডগুলো লেখার জন্য ব্যবহার করি।

4. **startElement, characters, endElement মেথড:**

```
@Override
public void startElement(String uri, String localName, String qName,
Attributes attributes) throws SAXException {
    // এলিমেন্টের শুরুতে কলব্যাক
}

@Override
public void characters(char[] ch, int start, int length) throws
SAXException {
    // টেক্সট কন্টেন্টে কলব্যাক
}

@Override
public void endElement(String uri, String localName, String qName) throws
SAXException {
    // এলিমেন্টের শেষে কলব্যাক
}
```

- `startElement` মেথড XML এলিমেন্টের শুরুতে কল করা হয়, এখানে এলিমেন্টের নাম এবং অ্যাট্রিবিউটগুলি পড়া হয়।
- `characters` মেথড XML এলিমেন্টের মধ্যে টেক্সট কন্টেন্ট পড়ে।
- `endElement` মেথড XML এলিমেন্টের শেষে কল করা হয়।

5. **XML** ফাইল পার্স করা:

```
saxParser.parse("example.xml", handler);
```

- `saxParser.parse("example.xml", handler)` পদ্ধতি XML ফাইলটি পার্স করে এবং `handler` অবজেক্টে ডিফাইন্ড কলব্যাক মেথডগুলি চালায়।

6. **Exception Handling:**

```
catch (Exception e) {
    e.printStackTrace();
}
```

- যদি কোনো সমস্যা ঘটে (যেমন XML ফাইল পাওয়া না গেলে), এক্সেপশন ক্যাচ করা হয় এবং এর স্ট্যাক ট্রেস প্রিন্ট করা হয়।

### সারাংশ:

SAX পার্সিং একটি ইভেন্ট-ভিত্তিকপদ্ধতিযাXML ডকুমেন্ট স্ট্রিমিংভাবে পড়ে। এটি মেমোরি-এফিশিয়েন্ট এবং বড় XML ডকুমেন্ট পার্স করার জন্য উপযোগী, তবে XML ডেটা পরিবর্তন করা সম্ভব নয়। SAX পার্সিংয়ের মাধ্যমে আপনি XML ডকুমেন্টের বিভিন্ন অংশের উপর নিয়ন্ত্রণ রাখতে পারেন এবং প্রতি ইভেন্টে নির্ধারিত মেথডগুলো ব্যবহার করে ডেটা প্রক্রিয়া করতে পারেন।

## StAX (Streaming API for XML) পার্সিং একটি XML পার্সিং পদ্ধতি যা স্ট্রিমিং ভিত্তিক এবং মেমরি-এফিশিয়েন্ট। StAX ব্যবহার করে আপনি XML ডকুমেন্টটি একটি স্ট্রিমের মতো একে একে পড়তে পারেন এবং এতে প্রয়োজনীয় তথ্য সংগ্রহ করতে পারেন। এটি SAX এর মতো ইভেন্ট-বেসড, কিন্তু এর প্রক্রিয়া অনেক বেশি নিয়ন্ত্রণযোগ্য এবং ডেটার উপর আরও বেশি সরাসরি কাজ করতে দেয়।

### StAX Parsing এর মূল বৈশিষ্ট্য

1. **স্ট্রিমিং পদ্ধতি:** StAX একটি স্ট্রিমিং পদ্ধতি ব্যবহার করে XML ডকুমেন্ট পড়ে। এটি একে একে XML ডকুমেন্টের বিভিন্ন অংশ পড়ে, এবং মেমরিতে শুধুমাত্র বর্তমান স্ট্রিমের অংশটাই থাকে।
2. **পুলিং এবং পুশিং মোড:** StAX দুটি মোডে কাজ করে:

- **পুলিং (Pull) মোড:** আপনি XML স্ট্রিম থেকে তথ্যগুলি নিজে "পুল" করেন, অর্থাৎ আপনি প্রোগ্রাম কন্ট্রোলের অধীনে থাকেন।
- **পুশিং (Push) মোড:** XML স্ট্রিম থেকে তথ্যগুলি স্বয়ংক্রিয়ভাবে "পুশ" করা হয়, এটি সাধারণত SAX স্টাইলের মতো।

3. **মেমরি-এফিশিয়েন্ট:** StAX কেবলমাত্র বর্তমান XML অংশটি মেমরিতে রাখে, ফলে বড় XML ডকুমেন্ট পার্স করার সময় এটি কম মেমরি ব্যবহার করে।
4. **ডেটা মডিফিকেশন:** StAX সাধারণত শুধুমাত্র ডেটা পড়ার জন্য ব্যবহৃত হয়। তবে আপনি কাস্টম কোডের মাধ্যমে XML ডকুমেন্টের তথ্য প্রক্রিয়া করতে পারেন।

## StAX Parsing উদাহরণ

নিচে একটি StAX পার্সিং উদাহরণ দেওয়া হয়েছে যা XML ডকুমেন্ট থেকে তথ্য পড়ে এবং কনসোলে প্রিন্ট করে।

**XML ডকুমেন্ট (example.xml):**

```
<?xml version="1.0"?>
<employees>
    <employee id="1">
        <name>John Doe</name>
        <age>30</age>
        <position>Developer</position>
    </employee>
    <employee id="2">
        <name>Jane Smith</name>
        <age>25</age>
        <position>Designer</position>
    </employee>
</employees>
```

**StAX Parsing কোড:**

```
import javax.xml.stream.XMLOutputFactory;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamReader;
import javax.xml.stream.XMLStreamWriter;
import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamConstants;
import java.io.FileInputStream;

public class StAXParserExample {
    public static void main(String[] args) {
        try {
            // XMLInputFactory অবজেক্ট তৈরি করা
            XMLInputFactory factory = XMLInputFactory.newInstance();

            // XMLStreamReader অবজেক্ট তৈরি করা
            XMLStreamReader reader = factory.createXMLStreamReader(new
FileInputStream("example.xml"));

            // XML ডকুমেন্ট পার্সিং
            while (reader.hasNext()) {
                int event = reader.next();

                switch (event) {
                    case XMLStreamConstants.START_ELEMENT:
                        if (reader.getLocalName().equals("employee")) {
                            System.out.println("Start Element: employee");
                            System.out.println("Employee ID: " +
reader.getAttributeValue(null, "id"));
                        }
                        break;
```

```
                    case XMLStreamConstants.CHARACTERS:
                        String text = reader.getText().trim();
                        if (!text.isEmpty()) {
                            System.out.println("Content: " + text);
                        }
                        break;

                    case XMLStreamConstants.END_ELEMENT:
                        if (reader.getLocalName().equals("employee")) {
                            System.out.println("End Element: employee");
                        }
                        break;
                }
            }
            reader.close();
        } catch (XMLStreamException | java.io.IOException e) {
            e.printStackTrace();
        }
    }
}
```

## কোডের অংশবিশেষ ব্যাখ্যা:

1. **Import Statements:**

   ```
   import javax.xml.stream.XMLInputFactory;
   import javax.xml.stream.XMLStreamException;
   import javax.xml.stream.XMLStreamReader;
   import javax.xml.stream.XMLStreamConstants;
   import java.io.FileInputStream;
   ```

   - StAX পার্সিং এবং XML স্ট্রিম রিডার এর জন্য প্রয়োজনীয় ক্লাসগুলো আমদানি করা হয়েছে।

2. **XMLInputFactory** এবং XMLStreamReader তৈরি:

   ```
   XMLInputFactory factory = XMLInputFactory.newInstance();
   XMLStreamReader reader = factory.createXMLStreamReader(new
   FileInputStream("example.xml"));
   ```

   - `XMLInputFactory` একটি ফ্যাক্টরি ক্লাস যা `XMLStreamReader` তৈরি করার জন্য ব্যবহৃত হয়। `XMLStreamReader` অবজেক্ট XML ডকুমেন্ট পড়তে ব্যবহৃত হয়।

3. **XML ডকুমেন্ট পার্সিং:**

   ```
   while (reader.hasNext()) {
       int event = reader.next();
       switch (event) {
           case XMLStreamConstants.START_ELEMENT:
               // এলিমেন্টের শুরুতে কলব্যাক
               break;
           case XMLStreamConstants.CHARACTERS:
               // টেক্সট কন্টেন্টে কলব্যাক
               break;
           case XMLStreamConstants.END_ELEMENT:
               // এলিমেন্টের শেষে কলব্যাক
               break;
       }
   }
   ```

   - `while (reader.hasNext())` লুপ XML ডকুমেন্টের প্রতিটি অংশে চলে। `reader.next()` পদ্ধতি পরবর্তী ইভেন্ট রিটার্ন করে (যেমন START_ELEMENT, CHARACTERS, END_ELEMENT)। এই ইভেন্টগুলির উপর ভিত্তি করে ডেটা পড়া হয় এবং প্রিন্ট করা হয়।
   -

4. **স্ট্রিম ক্লোজ করা:**

```
reader.close();
```

   - স্ট্রিম ব্যবহার শেষে ক্লোজ করা হয়।

5. **Exception Handling:**

```
catch (XMLStreamException | java.io.IOException e) {
    e.printStackTrace();
}
```

   - যদি কোনো সমস্যা ঘটে (যেমন XML ফাইল পাওয়া না গেলে), এক্সেপশন ক্যাচ করা হয় এবং এর স্ট্যাক ট্রেস প্রিন্ট করা হয়।

### সারাংশ:

StAX পার্সিং একটি স্ট্রিমিং ভিত্তিক XML পার্সিং পদ্ধতি যা মেমরি-এফিশিয়েন্ট এবং বড় XML ডকুমেন্ট পার্স করার জন্য উপযুক্ত। এটি XML ডকুমেন্টকে একে একে পড়ে এবং ইভেন্ট-বেসড পদ্ধতি ব্যবহার করে তথ্য প্রক্রিয়া করে। StAX-এর পুলিং মোড ব্যবহারে আপনি XML ডেটার উপর পূর্ণ নিয়ন্ত্রণ রাখতে পারেন এবং একটি স্ট্রিমিং পদ্ধতির মাধ্যমে ডেটা পড়তে পারেন।

## JAXB (Java Architecture for XML Binding) ব্যবহার করে XML এবং Java অবজেক্টের মধ্যে কনভার্সন অনেক সহজ হয়ে যায়। JAXB একটি API যা XML ডেটাকে Java অবজেক্টে এবং Java অবজেক্টকে XML ডেটাতে রূপান্তর করতে সাহায্য করে।

### JAXB এর মূল বৈশিষ্ট্য

1. অ্যাম্বিগুয়াস কনভার্সন: JAXB XML ডেটা এবং Java অবজেক্টের মধ্যে দ্বিদিশদ্বিত কনভার্সন সহজ করে দেয়, অর্থাৎ XML থেকে Java অবজেক্ট তৈরি করা এবং Java অবজেক্ট থেকে XML ডেটা তৈরি করা।
2. অ্যানোটেশন ব্যবহার**:** JAXB XML ডেটার সাথে Java ক্লাসের সম্পর্ক নির্ধারণ করতে বিভিন্ন অ্যানোটেশন ব্যবহার করে। যেমন, `@XmlRootElement`, `@XmlElement`, `@XmlAttribute`, ইত্যাদি।
3. **স্বয়ংক্রিয় ম্যাপিং:** JAXB ক্লাস এবং XML ডেটা স্ট্রাকচারের মধ্যে স্বয়ংক্রিয়ভাবে ম্যাপিং করে, যার ফলে ম্যানুয়াল কোড লেখার প্রয়োজন হয় না।

### JAXB এর ব্যবহার

**১. JAXB** অ্যানোটেশন:

- `@XmlRootElement`: XML ডকুমেন্টের রুট এলিমেন্ট চিহ্নিত করতে ব্যবহৃত হয়।
- `@XmlElement`: একটি XML এলিমেন্টের সাথে Java ফিল্ড বা মেথড ম্যাপ করতে ব্যবহৃত হয়।
- `@XmlAttribute`: XML অ্যাট্রিবিউটের সাথে Java ফিল্ড ম্যাপ করতে ব্যবহৃত হয়।

**২. JAXB কোড উদাহরণ:**

**XML ডকুমেন্ট (employee.xml):**

```
<?xml version="1.0"?>
<employee id="1">
    <name>John Doe</name>
```

```
    <age>30</age>
    <position>Developer</position>
</employee>
```

**Java ক্লাস:**

```
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import javax.xml.bind.Unmarshaller;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import java.io.StringReader;
import java.io.StringWriter;

// XML রুট এলিমেন্টের সাথে মেলানো Java ক্লাস
@XmlRootElement(name = "employee")
public class Employee {
    private int id;
    private String name;
    private int age;
    private String position;

    @XmlAttribute
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @XmlElement
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @XmlElement
    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @XmlElement
    public String getPosition() {
        return position;
    }

    public void setPosition(String position) {
        this.position = position;
    }
}


//……..
//……..

// JAXB ব্যবহার করে পার্সিং এবং মার্শালিং
public class JAXBExample {
    public static void main(String[] args) {
        try {
            // JAXBContext তৈরি
```

```
            JAXBContext context = JAXBContext.newInstance(Employee.class);

            // Unmarshaller তৈরি
            Unmarshaller unmarshaller = context.createUnmarshaller();
            String xml = "<employee id=\"1\"><name>John
Doe</name><age>30</age><position>Developer</position></employee>";

            // XML থেকে Java অবজেক্টে কনভার্সন
            Employee employee = (Employee) unmarshaller.unmarshal(new
StringReader(xml));
            System.out.println("Employee ID: " + employee.getId());
            System.out.println("Name: " + employee.getName());
            System.out.println("Age: " + employee.getAge());
            System.out.println("Position: " + employee.getPosition());

            // Marshaller তৈরি
            Marshaller marshaller = context.createMarshaller();
            marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);

            // Java অবজেক্ট থেকে XML তে কনভার্সন
            StringWriter writer = new StringWriter();
            marshaller.marshal(employee, writer);
            System.out.println("XML Output:\n" + writer.toString());
        } catch (JAXBException e) {
            e.printStackTrace();
        }
    }
}
```

**ব্যাখ্যা:**

- **Java ক্লাস:**
    - `Employee` ক্লাসের সাথে XML ডকুমেন্টের স্ট্রাকচার ম্যাচ করার জন্য JAXB অ্যানোটেশন ব্যবহার করা হয়েছে। `@XmlRootElement`, `@XmlElement`, এবং `@XmlAttribute` ব্যবহার করে XML এলিমেন্ট এবং অ্যাট্রিবিউটগুলি Java ফিল্ড এবং মেথডের সাথে যুক্ত করা হয়েছে।
- **JAXBContext, Unmarshaller এবংMarshaller:**
    - `JAXBContext` একটি কনটেক্সট অবজেক্ট যা JAXB পার্সিং পরিচালনা করে।
    - `Unmarshaller` XML ডেটা থেকে Java অবজেক্ট তৈরি করতে ব্যবহৃত হয়।
    - `Marshaller` Java অবজেক্টকে XML ডেটাতে রূপান্তর করতে ব্যবহৃত হয়।
- **পার্সিং এবং মার্শালিং:**
    - XML স্ট্রিং থেকে Java অবজেক্ট তৈরি করা হয়েছে এবং Java অবজেক্ট থেকে XML স্ট্রিং তৈরি করা হয়েছে।

## সারাংশ:

JAXB XML এবং Java অবজেক্টের মধ্যে কনভার্সন প্রক্রিয়া সহজ করে দেয়। এটি অ্যানোটেশন ব্যবহার করে XML ডকুমেন্ট এবং Java ক্লাসের মধ্যে একটি স্বয়ংক্রিয় ম্যাপিং প্রক্রিয়া প্রদান করে, যার ফলে কোড লেখা সহজ হয় এবং XML ডেটার সাথে কাজ করা আরও সুবিধাজনক হয়।