Chapter 9 Java and XML

JAVACOSE@QQ.COM

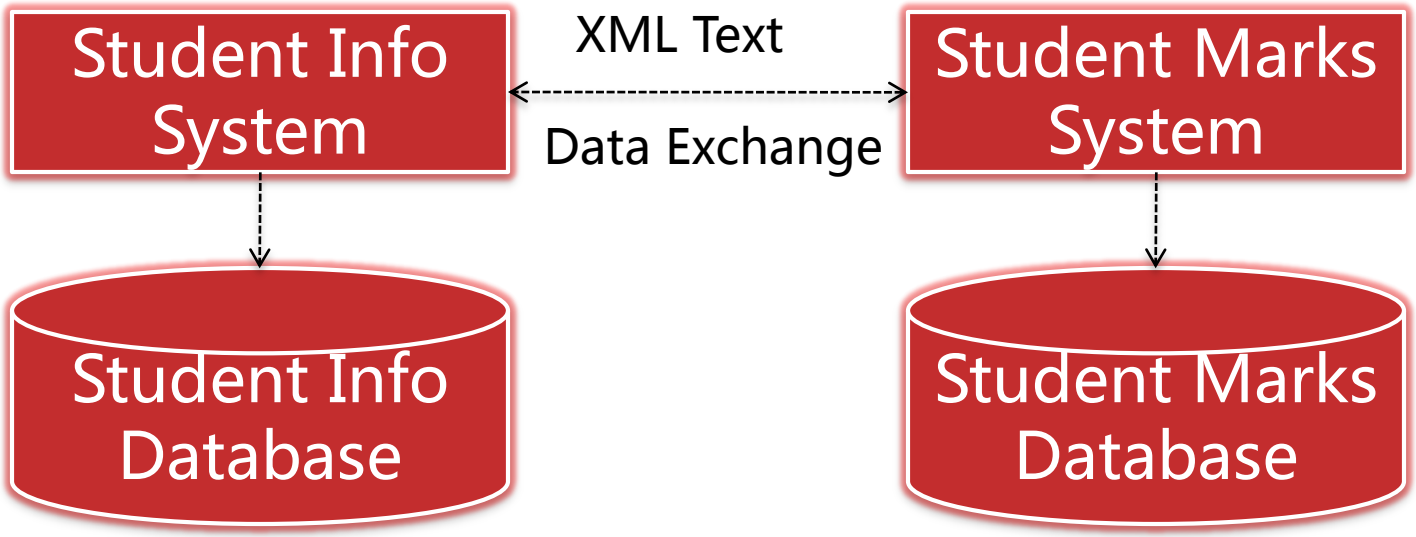
XIANG ZHANG

Content

- XML Pilot http://www.w3school.com.cn/
- Parsing XML Document
- DOM
- Objects in DOM
- Java Programming using DOM

XML Pilot

- XML 可扩展标记语言
  - eXtensible Markup Language
- Motivate: Data Exchange



XML Pilot

- Origin
  - SGML 标准通用化标记语言
    - Powerful
    - Extensible
    - Expensive

```
<QUOTE TYPE="example">
typically something like <ITALICS>this</ITALICS>
</QUOTE>
```

  - HTML 超文本标记语言
    - Limited function
    - Not extensible
    - Free

```
<html>
<head>
    <title>My Homepage</title>
</head>
<body>
    <p>I am a student in <b>CoSE@SEU</b></p>
</body>
</html>
```

XML Pilot – An Example

- John wrote to George in XML

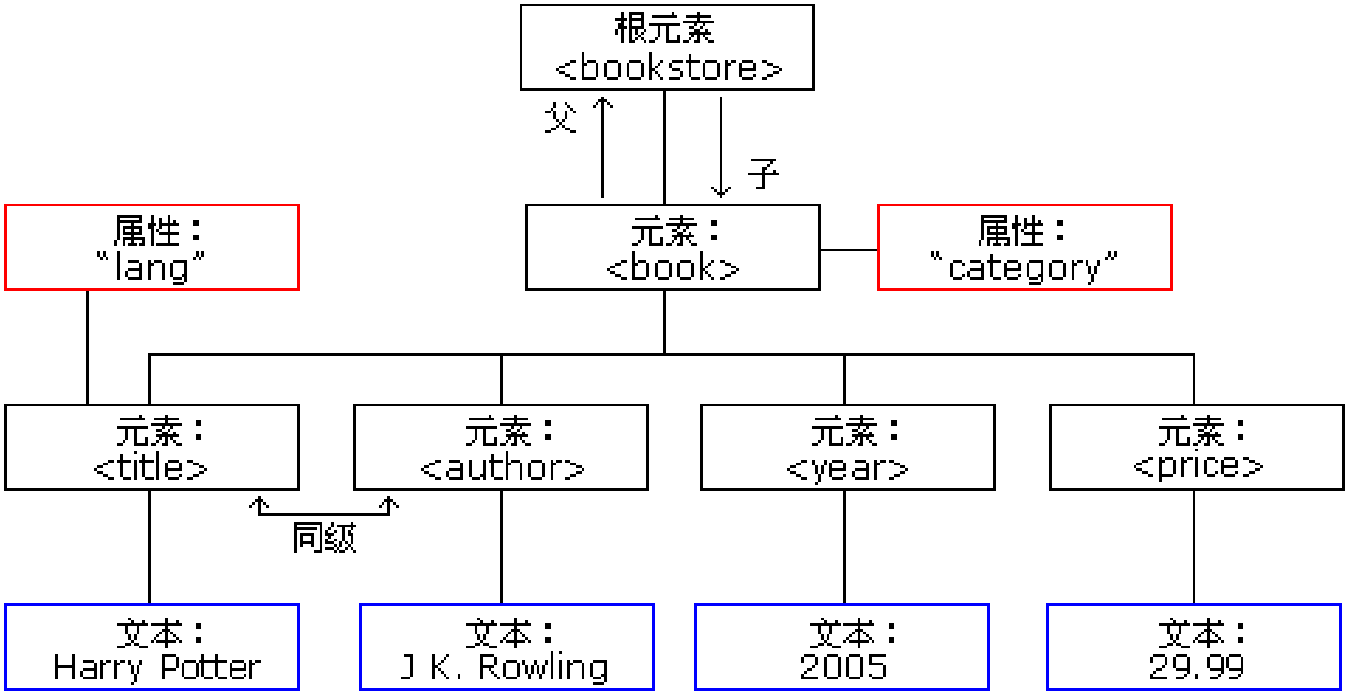
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
    <to>George</to>
    <from>John</from>
    <heading>Reminder</heading>
    <body>Don't forget the meeting!</body>
</note>
```

- An XML documents includes at least:
  - An XML declaration
  - An XML root element

XML Pilot – An Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
<book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
</book>
<book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
</book>
</bookstore>
```

XML Pilot – XML Tree Structure



XML vs. HTML

- All XML element must have a close tag

```
<p>This is a paragraph
<p>This is another paragraph
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

- XML tag is case-sensitive

```
<Message>这是错误的。</message>
<message>这是正确的。</message>
```

- XML element must be properly nested

```
<b><i>This text is bold and italic</b></i>
<b><i>This text is bold and italic</i></b>
```

XML vs. HTML

- XML document must have an root element

```
<root>
  <child>
    <subchild>.....</subchild>
  </child>
</root>
```

- XML attribute must be quoted

```
<note date=08/08/2008>
    <to>George</to>
    <from>John</from>
</note>
```

```
<note date="08/08/2008">
    <to>George</to>
    <from>John</from>
</note>
```

Entity Reference and Comments

- In XML, some characters have special meaning

```
<message>if salary < 1000 then</message>
<message>if salary &lt; 1000 then</message>
```

- Five reserved Entity Reference

&lt;	<	小于
&gt;	>	大于
&amp;	&	和号
&apos;	'	单引号
&quot;	"	引号

- XML comments

```
<!-- This is a comment -->
```

XML Element

- XML Element can have
  - Sub-element
  - Attribute
  - Text
- Naming of elements
  - Letters, numbers and other char
  - Cannot begin with figure, or punctuation, and "XML"
  - Cannot contain spaces

```
<bookstore>
<book category="CHILDREN">
    <title>Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
</book>
<book category="WEB">
    <title>Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
</book>
</bookstore>
```

CDATA

- Long text in XML can be represented in CDATA
- All content in CDATA will be interpreted as text
- No entity reference in CDATA
- CDATA grammar:
  - Begin with "<!CDATA["
  - End with "]]>"

```
<script>
<![CDATA[
function matchwo(a,b)
{
if (a < b && a < 0)
  {
  return 1
  }
else
  {
  return 0
  }
}
]]>
</script>
```

Well-formed XML

- A "Well-formed" XML should be:
  - XML document having root element
  - All elements being properly closed
  - Case-sensitive
  - All elements being properly nested
  - All attributes being properly quoted

Validated XML

- Well-formed XML only indicates that the format of XML document is correct
- We still need a way to check whether the data structure in XML document is right too
  - Each book price in bookstore should be positive float
  - Each book contains at least one title
- The way to validate XML
  - DTD（Document Type Definition）
  - XML Schema

namespace

- Name confliction may happen when two XML contains elements with same name:

```
<table>
    <tr>
    <td>Apples</td>
    <td>Bananas</td>
    </tr>
</table>
```

```
<table>
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
</table>
```

Name	Amount
Apple	20
Banana	15

namespace

- Using namespace to avoid naming confliction

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
  <h:td>Apples</h:td>
  <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table xmlns:f="http://www.w3school.com.cn/furniture">
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

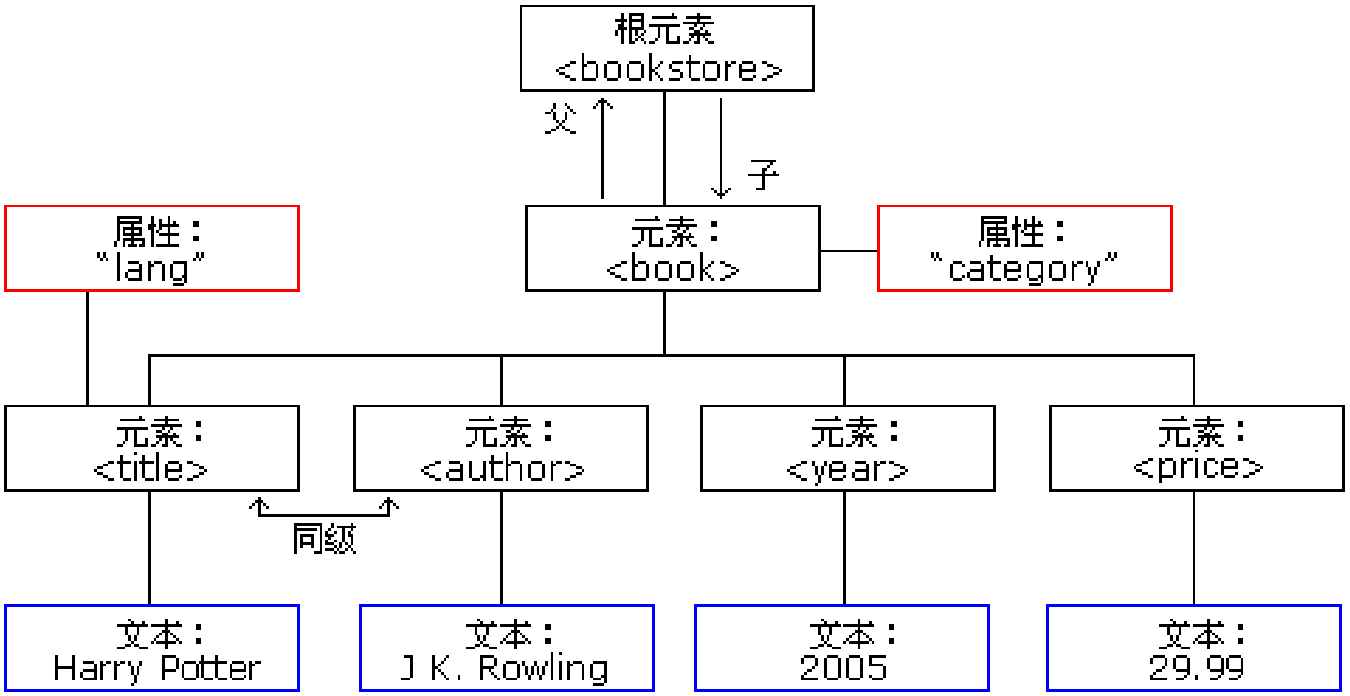
DOM

- Document Object Model 文档对象模型
- DOM is
  - Programming interface for handling XML document
  - Define how to visit and manipulate XML document
  - Represent XML document in a tree structure
- Feature
  - Independent with platform
  - Independent with language

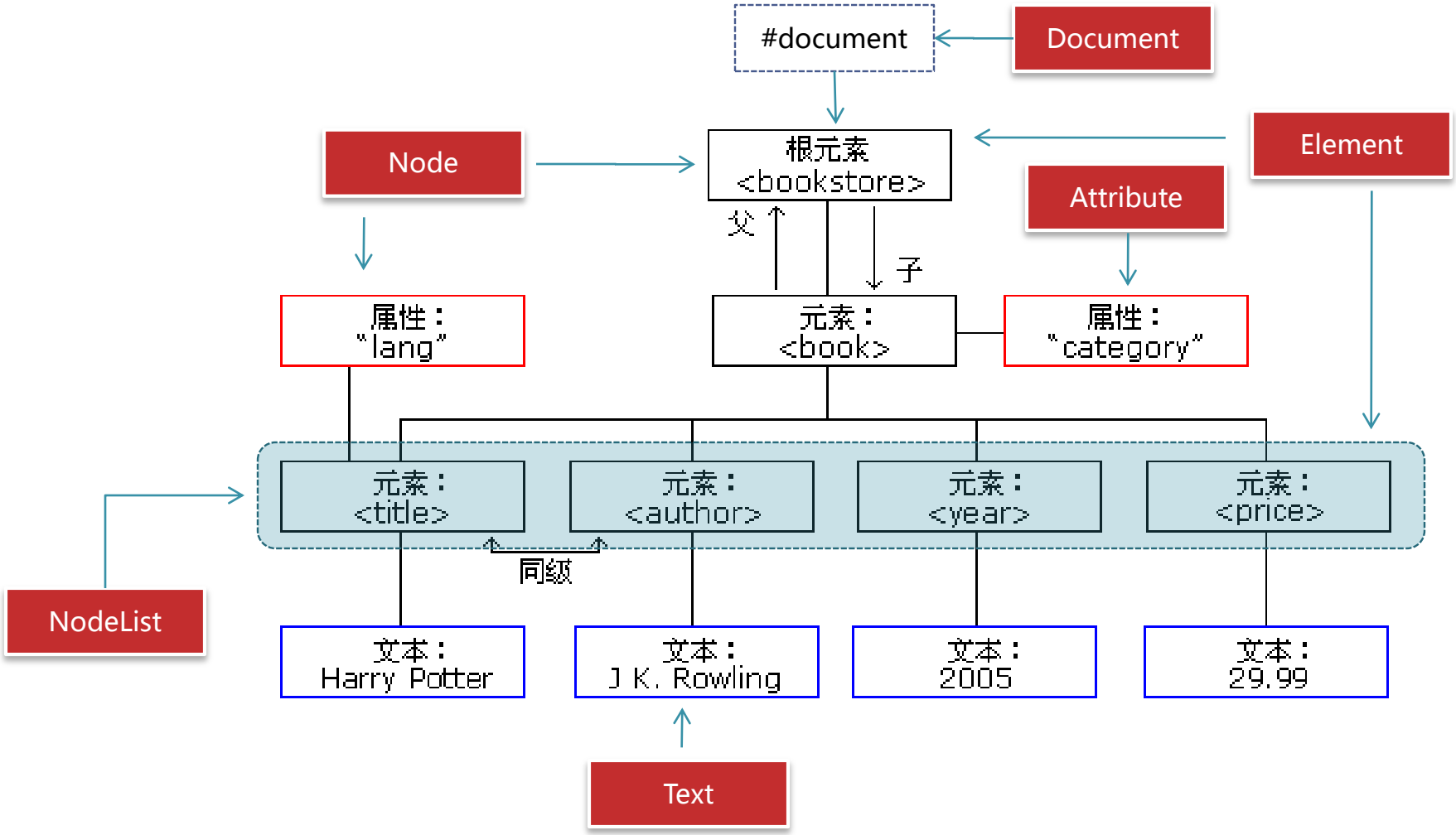
DOM Object

- org.w3c.dom.Document // XML doc entrance
- org.w3c.dom.Node // nodes in XML doc
- org.w3c.dom.Element // element node in XML doc
- org.w3c.dom.Attr // attribute node in XML doc
- org.w3c.dom.Text // text node in XML doc
- org.w3c.dom.NodeList // ordered nodelist in XML doc
- org.w3c.dom.NamedNodeMap // a mapping between node and node name

DOM Tree


根元素
<bookstore>
父
子
属性：
"lang"
元素：
<book>
属性：
"category"
元素：
<title>
元素：
<author>
元素：
<year>
元素：
<price>
同级
文本：
Harry Potter
文本：
J K. Rowling
文本：
2005
文本：
29.99

DOM树中的对象


#document
Document
Node
根元素
<bookstore>
Element
Attribute
父
子
属性：
"lang"
元素：
<book>
属性：
"category"
元素：
<title>
元素：
<author>
元素：
<year>
元素：
<price>
同级
NodeList
文本：
Harry Potter
文本：
J K. Rowling
文本：
2005
文本：
29.99
Text

Node Interface

- An interface for all nodes in XML doc tree
- Derive all other interfaces in DOM
- The type of a node is marked in nodetype
- Method
  - getNodeType() /NodeName() / NodeValue()
  - getParentNode() / getChildNodes()
  - getFirstChild () /getLastChild() / getChild
  - getNextSibling()

Document Interface

- Inherited from Node interface
- A virtual root element – the entrance of XML doc
- Method
  - getElementsByTagName(String tagname)
  - getDocumentElement()
  - createElement(String tagName)
  - createAttribute(String name)
  - createTextNode(String data)

Element Interface

- Inherited from Node Interface
- Representing XML elements
- Method
  - getAttributes() / setAttribute()
  - getTextContent() / setTextContent()
  - getOwnerDocument()

Attr Interface

- Inherited from Node Interface
- Representing XML attributes
- Methods
  - getOwnerElement()
  - getSpecified()
  - getValue() / setValue()

NodeList Interface

- Ordered list of nodes
- For example: an ordered list of all nodes derived by getChildNodes() of book element
- Methods
  - getLength()
  - item(int index)

NamedNodeMap Interface

- A Mapping between nodes and node names
- For example: a collection of all attributes derived by getAttributes() of an element
- Methods
  - getNamedItem(String name)
  - setNamedItem(Node node)

JAXP Implementing DOM

- JAXP : Java API for XML Parsing
- Using JAXP to parse XML

```
import javax.xml.parsers.*;
import org.w3c.dom.*;
...

File xmlFile = new File("d:/bookstore.xml");
try{
      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      DocumentBuilder xmlParser = factory.newDocumentBuilder();
      xmlDocument = xmlParser.parse(xmlFile);
}catch(Exception e){
      e.printStackTrace();
}
```

Episode – Factory Pattern

- Sample s = new Sample();
- If Sampleis an interface, then
  - Sample s1 = new MySample1();
  - Sample s2 = new MySample2();
- Using factory pattern

```
Class SampleFactory{
    public static Sample newSample(int parameter){
        if(parameter==1){
            //创建一个新的MySample1，并且返回
        }else if(parameter ==2){
            //创建一个新的MySample2，并且返回
        }
    }
}

Sample s1 = SampleFactory.newSample(1);
Sample s2 = SampleFactory.newSample(2);
```

Create a DOM Tree in Memory

```
<bookstore>
    <book category="COOKING">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
</bookstore>
```

Create a DOM Tree in Memory

```
public class BookstoreXML {
    File outputFile = new File("d:/output.xml");
    DocumentBuilder xmlBuilder;
    Document xmlDocument;
    public BookstoreXML(){
        try{
            //创建一个DocumentBuilderFactory
            DocumentBuilderFactory factory =
                DocumentBuilderFactory.newInstance();
            //创建一个DocumentBuilder
            xmlBuilder = factory.newDocumentBuilder();
            //创建一个Document
            xmlDocument = xmlBuilder.newDocument();
        }catch(Exception e){
            e.printStackTrace();
        }
    }
```

Create a DOM Tree in Memory

```
public void buildDOM(){
        //添加根节点
        Element bookstore = xmlDocument.createElement("bookstore");
        xmlDocument.appendChild(bookstore);

        //添加Book元素及其属性
        Element book = xmlDocument.createElement("book");
        book.setAttribute("category", "COOKING");
        bookstore.appendChild(book);

        //添加title author year price元素及其属性
        Element title = xmlDocument.createElement("title");
        title.setAttribute("lang", "en");
        title.setTextContent("Everyday Italian");
        Element author = xmlDocument.createElement("author");
        author.setTextContent("Giada De Laurentiis");
        Element year = xmlDocument.createElement("year");
        year.setTextContent("2005");
        Element price = xmlDocument.createElement("pirce");
        price.setTextContent("30.00");
        book.appendChild(title);book.appendChild(author);
        book.appendChild(year);book.appendChild(price);
}
```

Create a DOM Tree in Memory

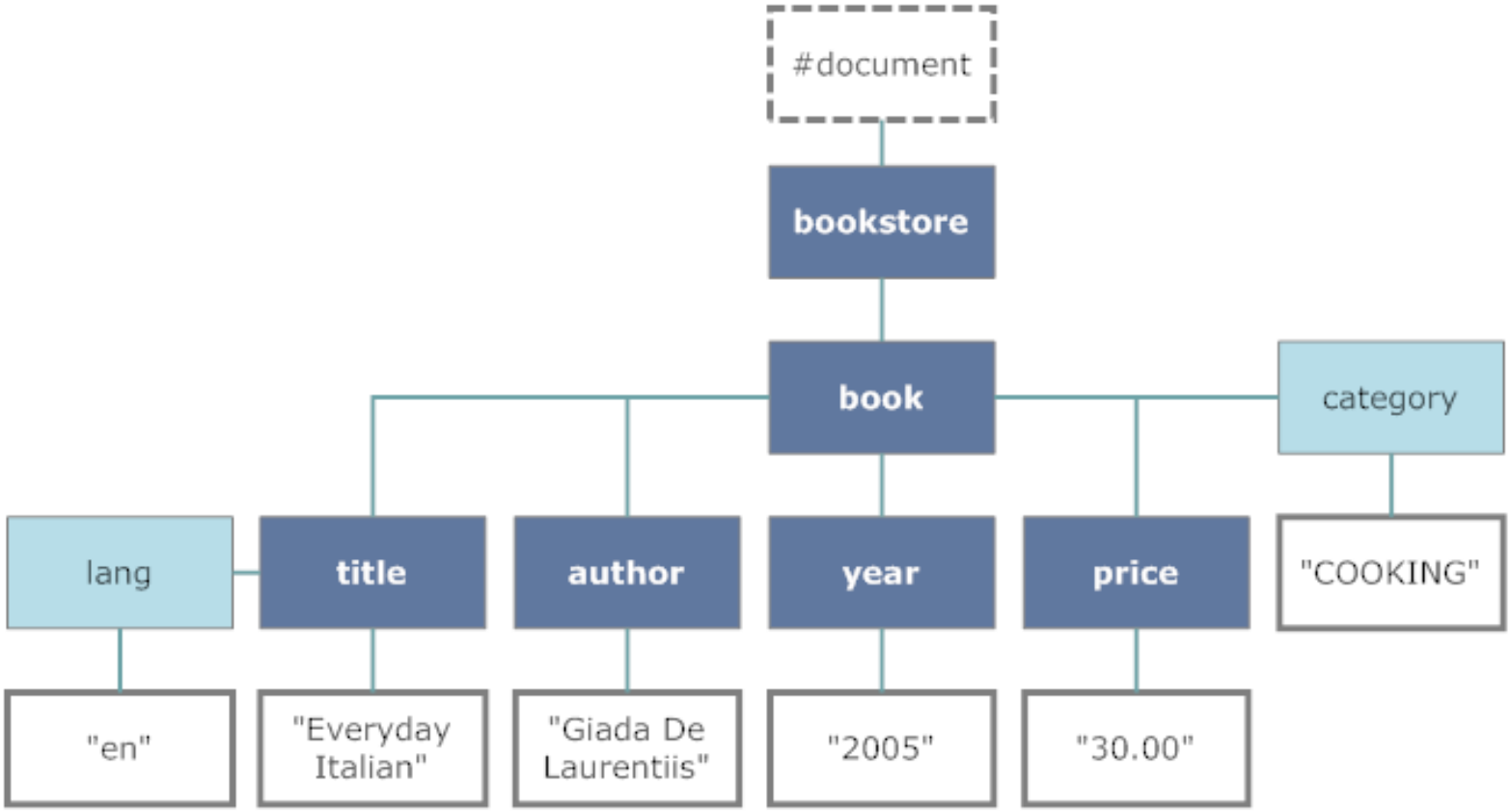
```
/*
 * 将Document对象转化为序列化的String
 */
public String toString(){
    try{
        TransformerFactory tran = TransformerFactory.newInstance();
        Transformer transformer = tran.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT,"yes");
        transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount","4");
        StringWriter xmlout = new StringWriter();
        StreamResult result = new StreamResult(xmlout);
        transformer.transform(new DOMSource(xmlDocument),result);
        return xmlout.toString();
    }catch(Exception e){
        e.printStackTrace();
        return null;
    }
}
```

Create a DOM Tree in Memory

```
    /*
    *  将Document对象的序列化String保存到文件中
    */
    public void saveDOM(){
        try{
            FileWriter xmlout = new FileWriter(outputFile);
            xmlout.write(this.toString());
            xmlout.close();
        }catch(Exception e){
            e.printStackTrace();
        }
    }

    public static void main(String[] args){
        BookstoreXML bookstore = new BookstoreXML();
        bookstore.buildDOM();
        System.out.println(bookstore);
        bookstore.saveDOM();
    }
}
```

Think: How to Traverse XML Nodes



Self-study

- XML Document Model
  - DOM
  - SAX
- XML Parser
  - JAXP
  - Xerces
  - JDOM
  - DOM4J