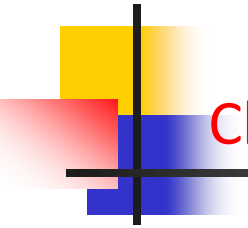
Mobile Application Development

Designing User Interface

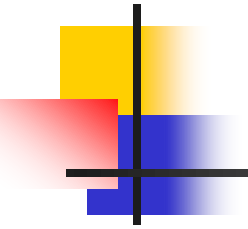
Programming Methodologies for Mobile Applications

- There are two main methods for designing user interface of a mobile application.
- 1. Declarative design method: involves writing code that specifies what should be done by the mobile device.
- When using android platform to develop mobile applications,
- **eX**tensible **M**arkup **L**anguage (XML) is used to implement declarative programming.
- 2. Procedural design method: involves writing code that specifies how user interface objects will be created.
- Android platform, Java programming language is used to implement procedural design.
- This is done by writing Java code to create and manipulate all the user interface objects such as buttons,labels and text fields



Choice of Design Method

- Android allows developers to create user interfaces in either style.
- The user interface can be designed almost entirely using either Java code or XML descriptors.
- There are Java APIs and the corresponding declarative XML attributes that do the same thing.
- Which should you use? Either way is valid, but XML is recommended due to the following reasons:
- 1. XML code is require few lines of code (shorter) than corresponding java code
- 2. XML easier to understand than the corresponding Java code



Choice of Design Method

- **eX**tensible **M**arkup **L**anguage (XML)

eXtensible Markup Language (XML)

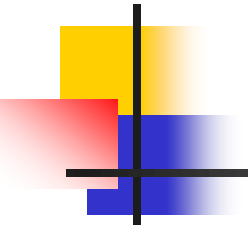
- In android platform eXtensible **M**arkup **L**anguage(XML) is used to describes what will be viewed by the user when the application is launched.
- For example, push buttons, labels and text field.
- It is also describes what information to be stored or transferred.

- Data stored on an external XML file can be available to all kinds of mobile devices.

The Difference Between XML and HTML

- XML was created to define, store, and transfer information while HTML was designed to display data, with focus on how data looks

- HTML documents can only use tags defined in the HTML standard (like <p>, <h1>, etc.) while XML allows the author to define his/her own tags and his/her own document structure



Terminologies used in XML

There are several terminologies that are used when programming using XML. These are:

1. XML tag
2. XML element
3. XML attributes
4. XML Namespace

XML tags

- A Tag is a construct for describing elements of a document.
- A tag begins with opening angle bracket (‘<‘) and ends with closing angle bracket (‘ >’)
- There are three types of tags

- *start-tags* indicate the beginning of an element.
- For example: <Text View>
- *end-tags* indicates the end of an element
- For example: </Text View>
- *empty-element tags*; do not have end-tags and must be terminated by a ‘/’. For example: <Text View />

- XML tags are not predefined. You must define your own tags

XML tags

XML tags Must be Properly Nested within each other

That is, the order of opening tags should be viceversa when closing the same element

Example:

<Linear Layout> <TextView > .....</TextView></LinearLayout>

- In the example above, "Properly nested" means that since the <TextView > element is opened inside the <Linear Layout> element, it must be closed inside the <Linear Layout> element.
- Example2
  - **<name><email>…</name></email>** is not allowed.
  - **<name><email>…</email></name>** is allowed

What is an XML Element?

- An XML element is A logical document component which either begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag .
- The characters between the start- and end-tags, if any, are the element's *content*, and may contain markup, including other elements, which are called *child elements*

An element can contain:

1. other elements text
2. attributes
3. Both elements and attributes.

XML elements: Example

```
<resources>
    <color name="orange">#ff5500</color>
    <color name="blue">#0066cc</color>
    <color name="gold">#e6b121</color>
    <color name="gray">#999999</color>
</resources>
```

In the example above,
<resources> element contains <color> element

XML elements: Example

- <color> element also has one a**ttribute** (name).

- There are four values assigned to name :

“ orange”, “blue”, “ gold”, “gray

- There are four text content:

- #ff5500, #0066cc, #e6b121, #999999.

Empty XML Elements

- An alternative syntax can be used for XML elements with no content:

- Example
- <resources> </resources>

XML Attributes

Attributes often provide information that is not a part of the data.

In the example below, the file type is irrelevant to the data, but can be important to the software that wants to manipulate the element:

Example:

```
<file type="gif">computer.gif</file>
```

XML Elements vs. Attributes

- In the example below:
- Button element has three attributes (id, layout_width, layout_height and text).
- Each attribute is initialized with a value

- Example:

```
<Button
 android:id="@+id/btnlogin"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Login" >
</Button >
```

XML Attributes for Metadata

- ID references are assigned to elements for identification purposes.

- In the following example Button element is assigned an ID btnlogin

```
<Button
 android:id="@+id/btnlogin"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="Login" >
</Button >
```

XML Namespaces (xmlns) Attribute

- A Namespace is a set of unique names assigned to a group that is identified with one name.
- It is a mechanisms by which element and attribute name can be assigned to group.
- XML Namespaces provide a method to avoid element name conflicts.
- The Namespace is identified by Uniform Resource Identifiers (URI).

- Example:

```
xmlns:android="http://schemas.android.com/apk/res/android"
```

XML Namespaces (xmlns) Attribute

A Namespace is declared using reserved attributes.
Such an attribute name must either be **xmlns** or begin with **xmlns:** as follows:

```
<element xmlns: name="URL">
```

The Namespace starts with the keyword **xmlns**.
The word **name** is the Namespace prefix.
The **URL** is the Namespace identifier.

XML Namespaces (xmlns) Attribute : **Example**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/gold"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="User Name"
    android:gravity="center"
    />
</LinearLayout>
```

Here, the Namespace prefix is **Android**, and the Namespace identifier (URI) as http://schemas.android.com/apk/res/android.

This means, the element names and attribute names with the **android** prefix ,all belong to the http://schemas.android.com/apk/res/android namespace.

XML Namespaces (xmlns) Attribute

- In the example above, the xmlns attribute in the <linear Layout> tag give the android a qualified namespace.

- When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.
- The purpose is to give the namespace a unique name.
- However, often companies use the namespace as a pointer to a web page containing namespace information.

Uniform Resource Identifier (URI)

- A Uniform Resource Identifier (URI) is a string of characters which identifies an Internet Resource .
- Such identification enables interaction with representations of the web resource over a network, typically the World Wide Web, using specific protocols.
- An URI identifies a resource either by location, or a name, or both.
- A URI has two specializations:
- 1. Uniform Resource Locator (URL)
- 2. Universal Resource Name (URN).

Uniform Resource Locator (URL)

- Uniform Resource Locator (URL) is used to identify an Internet domain address.
- It specifies where an identified resource is located and the protocol for retrieving it.
- Examples of retrieving protocol are:
- http://, ftp:// or smb://

Uniform Resource Name (URN)

- An Uniform Resource Name (URN) is a Uniform Resource Identifier (URI) that uses he URN scheme, and defines the identity of a resource.
- It does not imply availability of the identified resource.
- A URN has to be of this form
- "urn:" <NID> ":" <NSS>
- where
- <NID> is the Namespace Identifier,
- <NSS> is the Namespace Specific String.

Uniform Resource Name (URN)

Example1: •urn:isbn:0451450523

- In this example, a book is identified by its book number.
- ISBN stands for **International Standard Book Number**.
- It is a unique numeric commercial book identifier.
- **ISBN** is assigned to each edition and variation (except re- printings) of a book.

Example 2: urn:issn:0167-6423

ISSN stands for **International Standard Serial Number**

It is an internationally accepted code which identifies the title of serial publications. e.g. Journal is identified by its serial number

Uniform Resource Name (URN):

Other Examples

URN	corresponds to
`urn:isbn:0451450523`	The 1968 book *The Last Unicorn*, identified by its book number.
`urn:isan:0000-0000-9E59-0000-O-0000-0000-2`	The 2002 film *Spider-Man*, identified by its audiovisual number.
`urn:ISSN:0167-6423`	The scientific journal *Science of Computer Programming*, identified by its serial number.
`urn:ietf:rfc:2648`	The IETF's RFC 2648.
`urn:mpeg:mpeg7:schema:2001`	The default namespace rules for MPEG-7 video metadata.
`urn:oid:2.16.840`	The OID for the United States.
`urn:uuid:6e8bc430-9c3a-11d9-9669-0800200c9a66`	A version 1 UUID.
`urn:nbn:de:bvb:19-146642`	A National Bibliography Number for a document, indicating country (`de`), regional network (`bvb` = *Bibliotheksverbund Bayern*), library number (19) and document number.

XML Document tree Structure

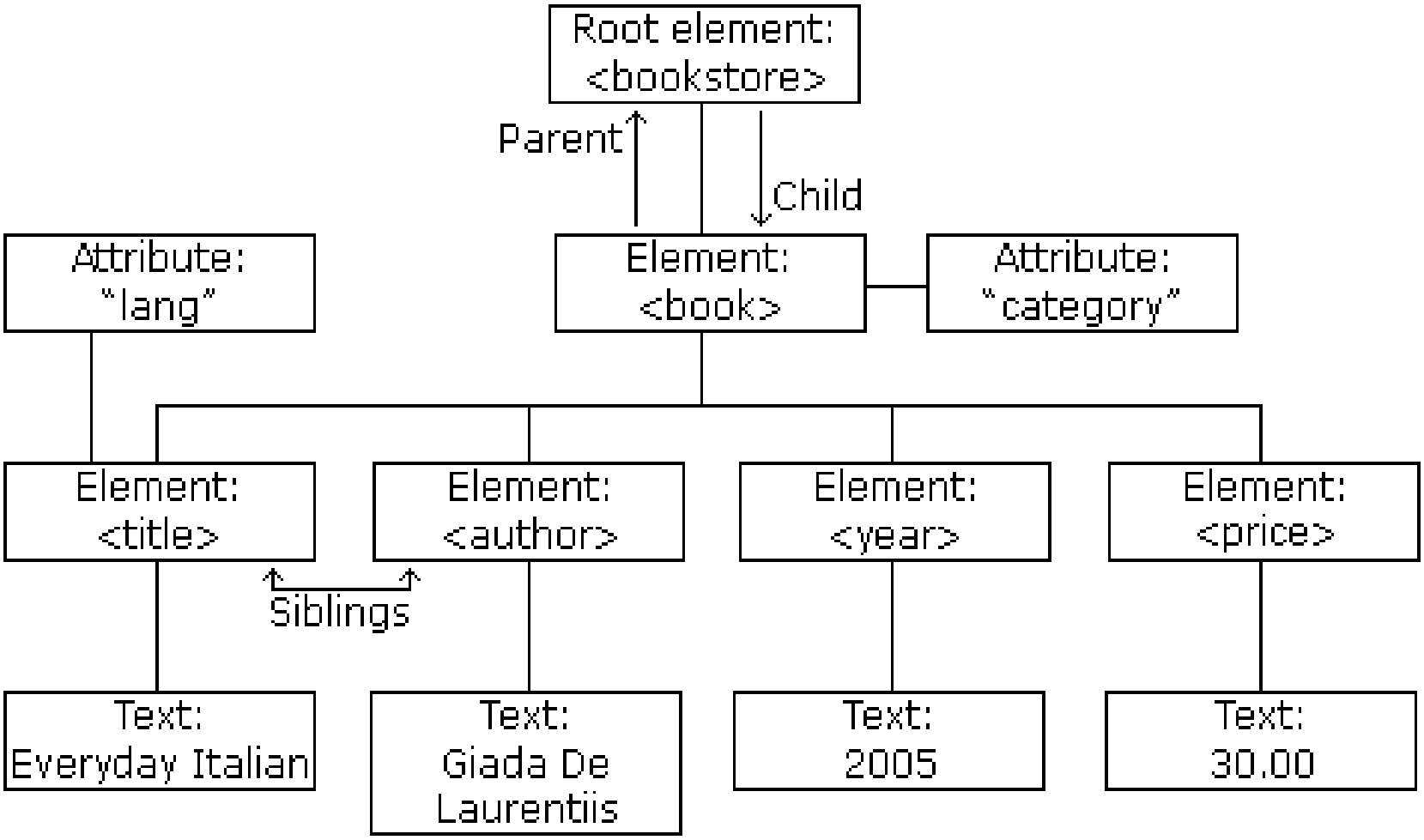
- XML documents must contain a **root element**.
- A document has a single root element

- This element is "the parent" of all other elements.
- The tree starts at the root and branches to the lowest level of the tree.
- All elements can have sub elements (child elements):
- <root>
  <child>
  <subchild>.....</subchild>
  </child>
  </root>

XML Document tree Structure

- The terms parent, child, and sibling are used to describe the relationships between.
- The terms parent, child, and sibling are used to describe the relationships between elements.
- Parent elements have children. Children on the same level are called siblings (brothers or sisters).
- All elements can have text content and attributes (just like in HTML).
-

XML Document tree Structure example



XML Document tree Structure example

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

XML Document tree Structure example

- In this example, the root element in the example is <bookstore>.
- All <book> elements in the document are contained within <bookstore>.
- The <book> element has 4 children:
- <title>,< author>, <year>, <price>.

XML Syntax Rules

1. In XML, it is illegal to omit the closing tag.

- All elements **must** have a closing tag:

Example

- <Edit Text>.....</Edit Text>

- **Note**: XML declaration did not have a closing tag since it is not a part of the XML document itself, and it has no closing tag.

XML Syntax Rules

2. **XML Tags are Case Sensitive**

- Example: The tag <TextView> is different from the tag <textview>.

- Opening and closing **tags must be written with the same case:**
- Example:

- <Button> This is correct </Button>

- **Note:** "Opening and closing tags" are often referred to as "Start and end tags".

```
<?xml version="1.0" encoding="utf-8"?>
```

XML Syntax Rules

3. XML Documents Must Have a Root Element

- XML documents must contain one element that is the **parent** of all other elements.
- This element is called the **root** element.
- Format:

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

XML Syntax Rules

4. XML Attribute Values Must be Quoted

- XML elements can have attributes in name/value pairs just like in HTML.
- In XML, the attribute values must always be quoted.

- Example:

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="User Name"
    android:gravity="center"
    />
</LinearLayout>
```

XML Syntax Rules

5. Comments in XML

Comments refers to statements that will be ignored by CPU during execution.

The are used to help understanding of the code

- The syntax for writing comments in XML is as follows:

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

XML Syntax Rules

6. White-space is Preserved in XML

- This means that the white-space in a document is not truncated.

- Example:
- XML : User Name

- Output: User Name

XML Naming Rules

XML elements must follow six naming rules:

1. Names can contain letters, numbers, and other characters
2. Names cannot start with a number or punctuation character
3. Names cannot start with the letters xml (or XML, or Xml, etc)
4. Names cannot contain spaces
5. Any name can be used, no words are reserved.
6. The name should be meaningful

Exercise1

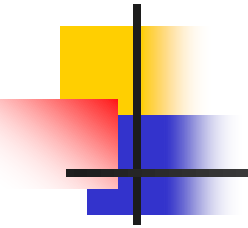
Consider the following login.xml File

login.xml | Login.java

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".Login">

    <TextView android:text="@string/hello_world"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```



Exercise1

Identify the following elements in the login.xml file:

1. Root element
2. Parent elements
3. Child elements
4. Attributes
5. XML Namespaces
6. Values

Exercise 2

Draw a tree structure that represents the following XML document

```
<?xml version="1.0"?>
<Company>
 <Employee>      <FirstName>Tanmay</FirstName>
          <LastName>Patil</LastName>
          <ContactNo>1234567890</ContactNo>
          <Email>tanmaypatil@xyz.com</Email>
          <Address>
                    <City>Bangalore</City>
                    <State>Karnataka</State>
                    <Zip>560212</Zip>
          </Address>
</Employee>
</Company>
```

End

Thank You

Questions