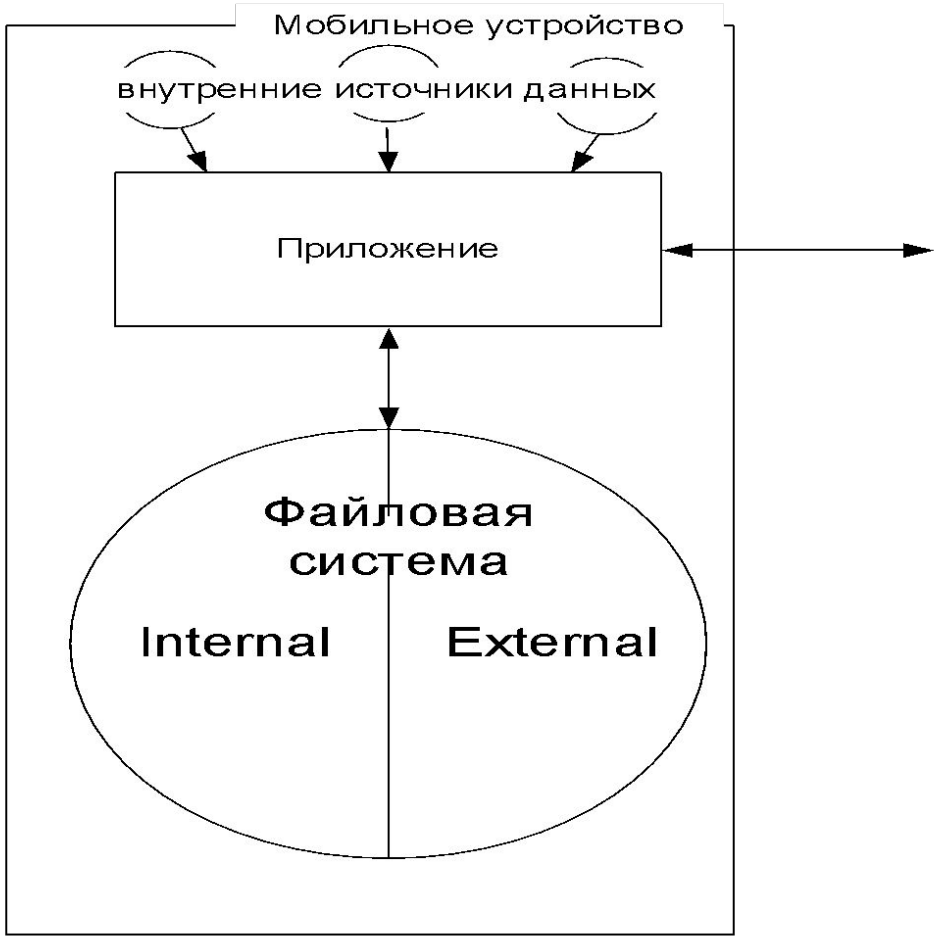
Программирование и безопасность баз данных мобильных систем

Лекция 3

Приложение – локальная файловая система



Работа с файловой системой

- JSON Java Script Object Notation
- XML Extensible Markup Language

Работа с JSON

JSON

- JSON - JavaScript Object Notation — текстовый формат обмена данными

JSON

```
{
  "orderID": 12345,
  "shopperName": "Ivan Ivanov",
  "shopperEmail": "ivanov@example.com",
  "contents": [
    {
      "productID": 34,
      "productName": "Super product",
      "quantity": 1
    },
    {
      "productID": 56,
      "productName": "Wonderful product",
      "quantity": 3
    }
  ],
  "orderCompleted": true
}
```

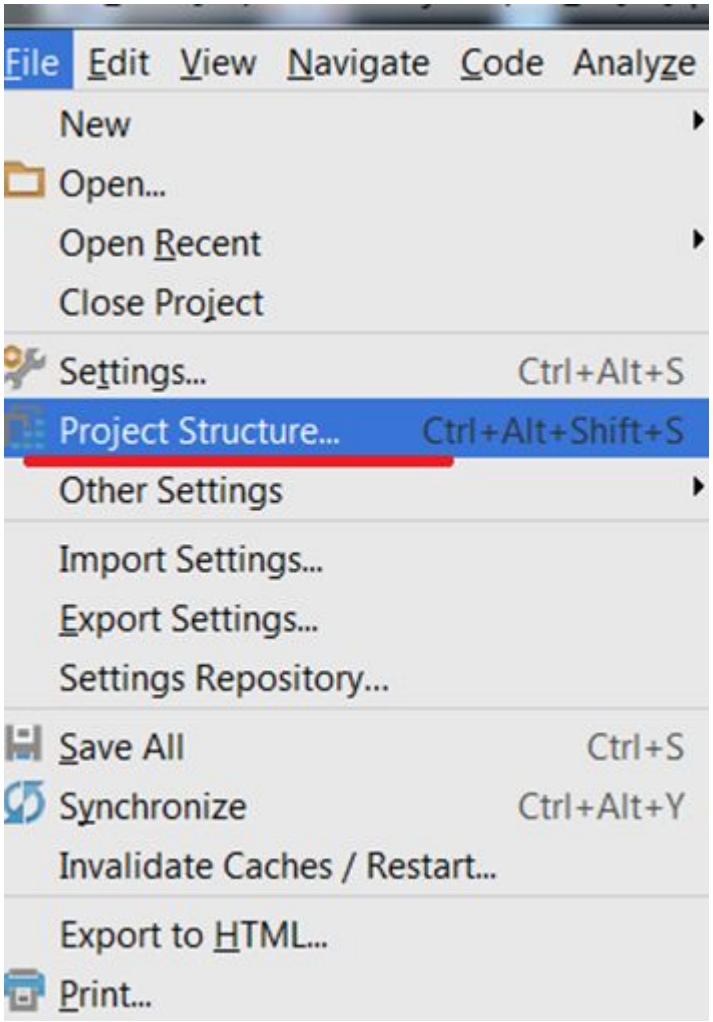
JSON

- Объект
- Массив
- Литералы
- Ключ: значение
- Строка
- Число

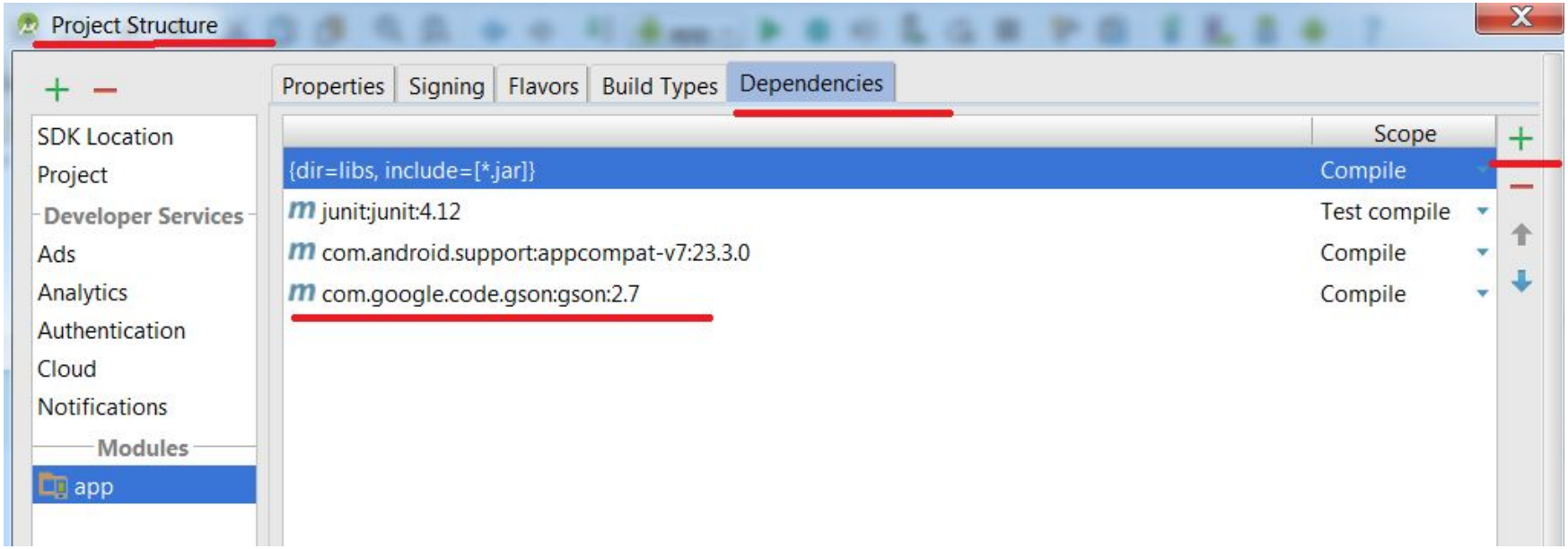
JSON

- Объект - неупорядоченное множество пар {ключ: значение}
- Ключ - строка
- Строка — это упорядоченное множество из нуля или более символов юникода, заключенное в двойные кавычки
- Число в десятичном формате
- Пары ключ-значение отделяются запятыми
- Массив - упорядоченное множество значений
- Массив заключается в [квадратные скобки]
- Значения в массиве разделяются запятыми
- Литералы true, false и null

Библиотека Google GSON



Библиотека Google GSON



GSON

- Основной класс – Gson
- Основные методы – toJson и fromJson

Работа с GSON

```
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
```

Работа с GSON – примитивные типы

```
Gson gson = new Gson();
String js1 = gson.toJson(123);
String js2 = gson.toJson("Hello, World");
String js3 = gson.toJson(true);

int     s1 = gson.fromJson(js1, int.class);
String  s2 = gson.fromJson(js2, String.class);
Boolean s3 = gson.fromJson(js3, Boolean.class);
```

Работа с GSON - массивы

```
int[]   m = new int[]{1,2,3,4};
float[] fm  = new float[] {1.0f, 2.0f, 3.0f, 4.0f};

String jm  = gson.toJson(m);
String jfm = gson.toJson(fm);

int[] rm = gson.fromJson(jm, int[].class);
float[] rfm = gson.fromJson(jfm, float[].class);
```

Работа с GSON - объекты

```
public class A {
    // примитивы
    int n;
    float f;
    char  c;
    double d;
    public A(int n, float f, char c, double d){
        this.n = n;
        this.f = f;
        this.c = c;
        this.d = d;
    }

}
```

```
A a = new A(1, 2.0f,'a', 3.0);
String ja = gson.toJson(a);
A ra = gson.fromJson(ja, A.class);
```

Работа с GSON

```
public class A {
    // примитивы
    int n;
    float f;
    char  c;
    double d;
    public A(int n, float f, char c, double d){
        this.n = n;
        this.f = f;
        this.c = c;
        this.d = d;
    }

}
```

```
public class B {
    int k = 1;
    String b = "String";
    A a = new A(1, 2.0f, 'w', 333.333);
}
```

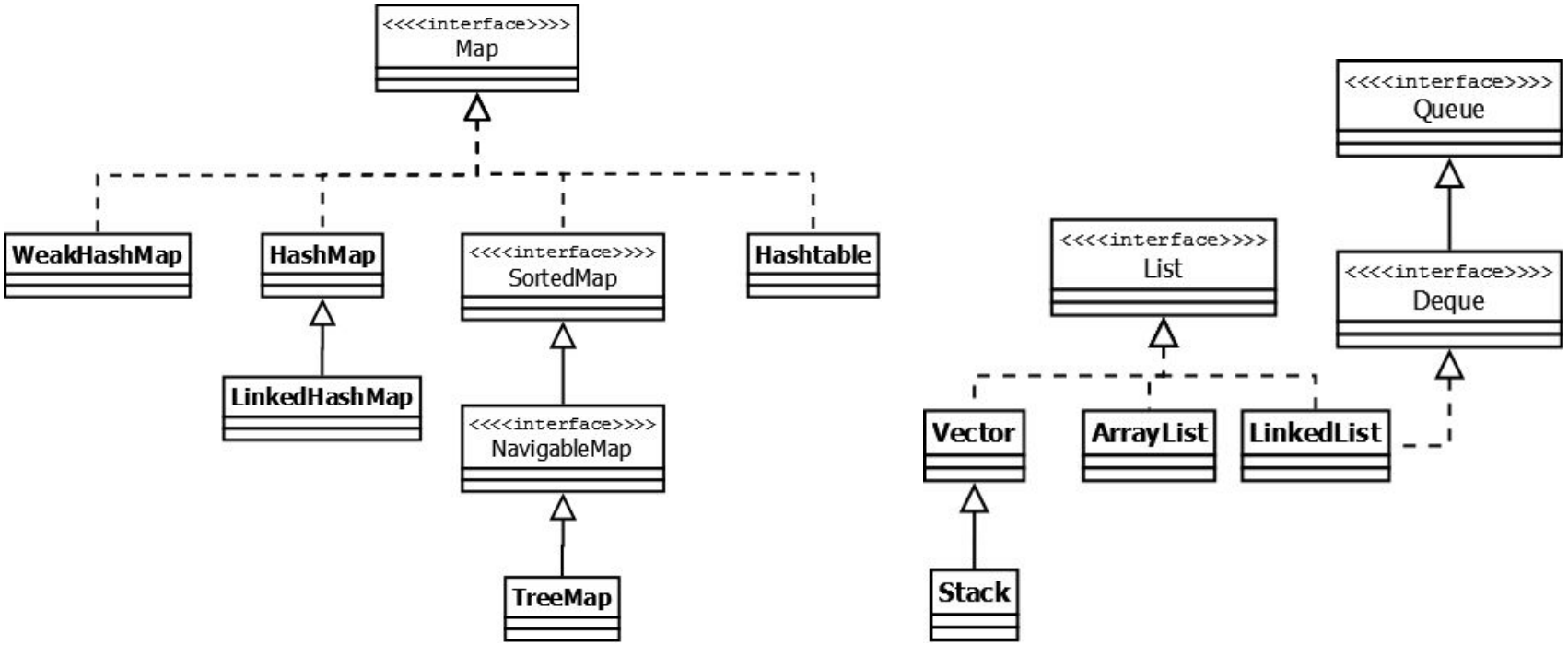
Работа с GSON

```
B  b = new B();
String jb = gson.toJson(b);
File f = new File(getFilesDir(), "jb.json");
try {
    FileWriter w = new FileWriter(f);
    w.write(jb);
    w.close();
}
catch (IOException e){
    Log.d("Lab04", e.getMessage());
}

 B rb = null;
 try {
      FileReader r = new FileReader(f);
      char[] buf = new char[(int)f.length()];
      r.read(buf);
      r.close();
      String s = new String(buf);
      Log.d("Lab04",s);
      rb = gson.fromJson(s,B.class);
 }
 catch (IOException e){
       Log.d("Lab04", e.getMessage());
 }
```

```
{"a":{"c":"w","d":333.333,"f":2.0,"n":1},"b":"String","k":1}
```

Работа с GSON - коллекции

```
Map<String, Integer> map = new LinkedHashMap<>();
map.put("a", 123);
map.put("b", 321);
String jmap = gson.toJson(map);
Map<String, Integer> rmap = gson.fromJson(jmap,new TypeToken<Map<String, Integer>>(){}.getType());
```



GsonBuilder

- GsonBuilder – разработка собственного сериализатора

GsonBuilder

```
public class  AConv implements JsonSerializer<A>, JsonDeserializer<A> {

    public JsonElement serialize(A src, Type t, JsonSerializationContext ctx){

        JsonObject o = new JsonObject();
        o.addProperty("n", src.n);
        o.addProperty("f", src.f);
        return o;
    }
    public A deserialize(JsonElement je, Type t,JsonDeserializationContext ctx){
        JsonObject o = je.getAsJsonObject();
        int n  = o.get("n").getAsInt();
        float f  = o.get("f").getAsFloat();
        return new A(n,f,' ',0.0);
    }

}
```

GsonBuilder

```
GsonBuilder gb = new GsonBuilder();
gb.registerTypeAdapter(A.class,new AConv());
Gson gson1 = gb.create();

A a1 = new A(1,1.0f,'x',1.0);
String ja1 = gson1.toJson(a1);
Log.d("Lab", ja1);
```

```
{"n":1,"f":1.0}
```

```
A ra1 = gson1.fromJson(ja1,A.class);
```

Работа с XML

XML

- Является подмножеством языка SGML – Standard Generalized Markup Language – метаязыка для определения языков разметки

W3C – стандартизация

- Консорциум Всемирной паутины - World Wide Web Consortium – организация, разрабатывающая и внедряющая технологические стандарты для web
- Глава – Тимоти Джон Бернерс-Ли
- Ок. 15 стандартов утверждены для XML:
  - XML Schema
  - XPath
  - XSLT
  - XQuery

XML Schema

- XML Schema — язык описания структуры XML-документа – предназначен для определения правил, которым должен подчиняться документ
- Создается модель данных документа, которая включает:
  - словарь (названия элементов и атрибутов);
  - модель содержания (отношения между элементами и атрибутами и их структура);
  - типы данных.
- Файл, содержащий XML Schema, обычно имеет расширение .xsd

XPath

- XPath - XML Path Language — язык запросов к элементам XML-документа.
- Разработан для организации доступа к частям документа XML в файлах трансформации XSLT

XSLT

- XSLT — eXtensible Stylesheet Language Transformations — язык преобразования XML-документов
- Правила выбора и преобразования данных пишутся на языке запросов XPath

XQuery

- XQuery — язык запросов, разработанный для обработки данных в формате XML

XML-языки – OASIS

- OASIS — Organization for the Advancement of Structured Information Standards — глобальный консорциум
- Управляет разработкой и принятием промышленных стандартов электронной коммерции

XML

- Правильно построенный документ – well-formed – соответствует синтаксическим правилам XML
- Валидный документ – valid – соответствует правилам описания типа документа

Синтаксические правила XML

- Наличие корневого элемента;
- Каждый открывающий тег имеет соответствующий закрывающий тег;
- Правильное вложение элементов документа;
- Атрибут должен иметь значение, которое берется в кавычки
- Древовидная структура документа

XML-файл

```
<?xml version="1.0" encoding="utf-8" ?>
<Group faculty ="ИТ" spec="ПОБМС"  course="3"  number="1" >

    <Student   surname="Иванов"  name="Иван" bday="1998.01.01"  />
    <Student   surname="Петров"  name="Петр" bday="1999.07.07"  />
</Group>
```

DocumentBuilder

- DocumentBuilderFactory - DocumentBuilder
- TransformerFactory – Transformer - TransformerException

Формирование и чтение XML-файла

```
try {
    DocumentBuilder xmlbld = xmlfry.newDocumentBuilder();
    Document doc = xmlbld.newDocument();

    Element group= doc.createElement("Group");
    group.setAttribute("faculty","ИТ");
    group.setAttribute("spec","ПОиБМС");
    group.setAttribute("course","3");
    group.setAttribute("number","1");

    Element student1 = doc.createElement("Student");
    student1.setAttribute("surname","I");
    student1.setAttribute("name","I");
    student1.setAttribute("bday","1998");
    group.appendChild(student1);

    Element student2 = doc.createElement("Student");
    student2.setAttribute("surname","P");
    student2.setAttribute("name","P");
    student2.setAttribute("bday","1997");
    group.appendChild(student2);
    doc.appendChild(group);
```

Формирование и чтение XML-файла

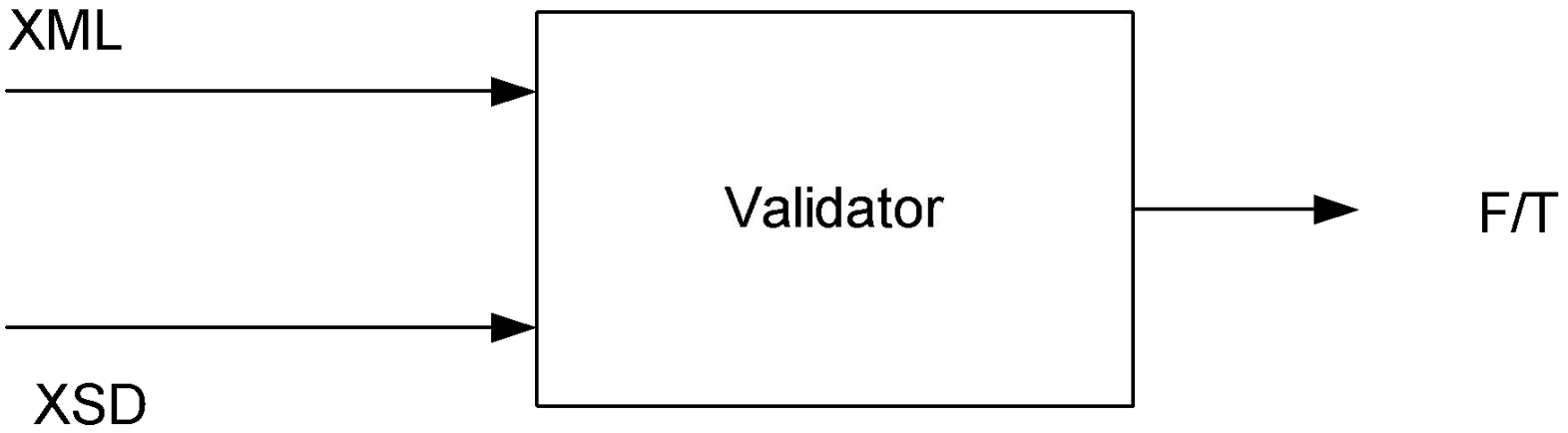
```
    TransformerFactory tfry = TransformerFactory.newInstance();
    Transformer trans = tfry.newTransformer();
    DOMSource src = new DOMSource(doc);

    FileOutputStream os = openFileOutput("XMLTest.xml",MODE_PRIVATE);
    StreamResult result = new StreamResult(os);
    trans.transform(src, result);
    os.close();
}
```

```
catch (ParserConfigurationException e){
    Log.d("Lab03XML:", e.getMessage());
}
catch(TransformerException e){
    Log.d("Lab03XML:", e.getMessage());
}
catch(IOException e){
    Log.d("Lab03XML:", e.getMessage());
}
```

Формирование и чтение XML-файла

```
DocumentBuilderFactory xmlfry1 = DocumentBuilderFactory.newInstance();
// xmlfry1.setValidating(false);
try {
    DocumentBuilder xmlbld = xmlfry1.newDocumentBuilder();
     File f = new File(getFilesDir(), "XMLTest.xml");
     FileReader fr  = new FileReader(f);
     char[] buf = new char[(int)f.length()];
     fr.read(buf);
     String s = new String(buf);
     Document doc = xmlbld.parse(f);

}
catch (ParserConfigurationException e ){
    Log.d("Lab03XML:", e.getMessage());
}
catch(SAXException e){
    Log.d("Lab03XML:", e.getMessage());
}
```

XML Schema



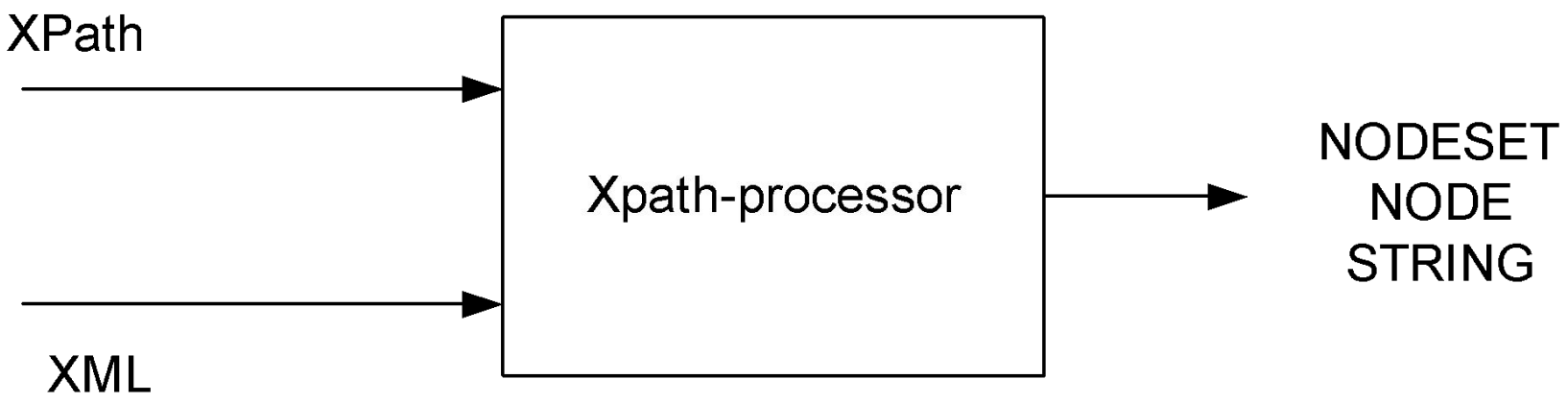
XML Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Group">
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="unbounded" name="Student">
                    <xs:complexType>
                        <xs:attribute name="surname" type="xs:string" use="required" />
                        <xs:attribute name="name" type="xs:string" use="required" />
                        <xs:attribute name="bday" type="xs:unsignedShort" use="required" />
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="faculty" type="xs:string" use="required" />
            <xs:attribute name="spec" type="xs:string" use="required" />
            <xs:attribute name="course" type="xs:integer" use="required" />
            <xs:attribute name="number" type="xs:integer" use="required" />
        </xs:complexType>
    </xs:element>
</xs:schema>
```

XML Schema

```
//XML-Schema
    try{
        FileInputStream xmlstream =  openFileInput("XMLTest.xml");
        FileInputStream xsdstream =  openFileInput("XMLFile.xsd");
        Source xmlsrc = new StreamSource(xmlstream);
        Source xsdsrc = new StreamSource(xsdstream);
        SchemaFactory xsdfry = SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
        Schema schema = xsdfry.newSchema(xsdsrc);
        Validator validator = schema.newValidator();
        validator.validate(xmlsrc);
    }
    catch(SAXException e){
       Log.d("Lab03XMLXSD:", e.getMessage());
    }
    catch(IOException e){
        Log.d("Lab03XMLXSD:", e.getMessage());
    }
    catch(Exception e){
        Log.d("Lab03XMLXSD:", e.getMessage());
    }
```

XPath



Запрос данных XPath

- для доступа к элементам и атрибутам XML-документа
  - Дочерние элементы узла /customer/*
  - Все атрибуты узла /customer/!?*
  - Чтобы вернуть только покупателей из региона Dallas /customer[@region = " Dallas "]

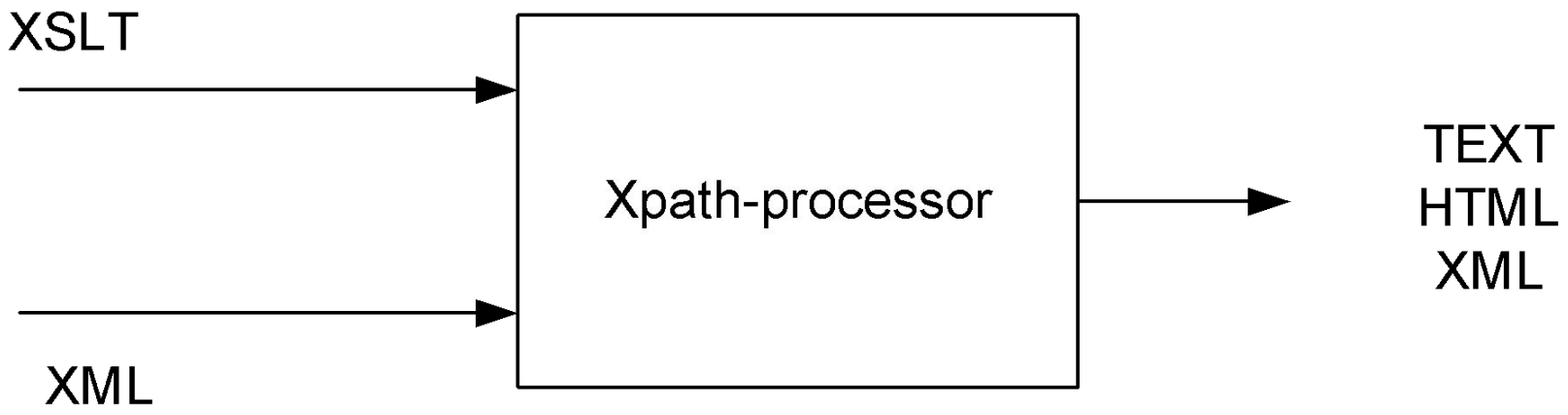
Оси данных XPath

Имя	Описание
ancestor	Содержит родительский узел и все вышестоящие родительские узлы вплоть до корневого узла.
ancestor-or-self	Содержит узлы-предки вместе с самим контекстным узлом, вплоть до корневого узла.
attribute	Содержит атрибуты контекстного узла, если контекстный узел — узел элемента.
child	Содержит дочерние узлы
descendant	Содержит дочерние и т.д. узлы
descendant-or-self	Содержит сам контекстный узел и все его дочерние и т.д. узлы.
following	Содержит все узлы того же документа, к которому принадлежит контекстный узел, которые находятся после текущего контекстного узла в порядке документа, но не включает никаких потомков, пространства имен или узлов атрибутов.
following-sibling	То же, что и following, но содержит все узлы, которые имеют того же ро дителя, что и контекстный узел.
namespace	Содержит пространство имен контекстного узла, до тех пор, пока контекстный узел является элементом.
parent	Содержит родительский узел контекстного узла. Корневой элемент не имеет родителя. Эта ось — противоположна оси child.

XPath

```
//XPath

    try {
        DocumentBuilder xmlbld = xmlfry1.newDocumentBuilder();
        File f = new File(getFilesDir(), "XMLTest.xml");
        FileReader fr = new FileReader(f);
        char[] buf = new char[(int) f.length()];
        fr.read(buf);
        String s = new String(buf);
        Document doc = xmlbld.parse(f);

        XPathFactory xpf = XPathFactory.newInstance();
        XPath xp = xpf.newXPath();

        XPathExpression xpe = xp.compile("/Group/Student");
        NodeList nl = (NodeList)xpe.evaluate(doc, XPathConstants.NODESET);
        int k = nl.getLength();
        String surname0 = nl.item(0).getAttributes().getNamedItem("surname").getNodeValue();
        String bday0    = nl.item(0).getAttributes().getNamedItem("bday").getNodeValue();
        String surname1 = nl.item(1).getAttributes().getNamedItem("surname").getNodeValue();
        String bday1    = nl.item(1).getAttributes().getNamedItem("bday").getNodeValue();

        XPathExpression xpe1 = xp.compile("/Group/Student[@surname='Иванов']");
        String bday2 =((Node)xpe1.evaluate(doc, XPathConstants.NODE))
                                .getAttributes().getNamedItem("bday").getNodeValue();

    }
    catch (Exception e){
        Log.d("Lab03XMLXPath:", e.getMessage());
    }
```

XSLT



XSLT

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Group faculty ="ИТ" spec="ПОБМС"  course="3"  number="1" >
    <Student   surname="Иванов"  name="Иван" bday="1998"  />
    <Student   surname="Петров"  name="Петр" bday="1999"  />
</Group>
```

ГРУППА

Факультет	ИТ
Курс	3
Специальность	ПОБМС
Номер	1

СПИСОК СТУДЕНТОВ

Иванов Иван 1998г.р.
Петров Петр 1999г.р.

XSLT

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-prefixes="msxsl"
>
    <xsl:output method="text" encoding="utf-16" />


    <xsl:template match="Group">
                              ГРУППА
      Факультет       <xsl:value-of select ="@faculty"/> <xsl:text></xsl:text>
      Курс            <xsl:value-of select ="@course"/> <xsl:text></xsl:text>
      Специальность   <xsl:value-of select ="@spec"/>  <xsl:text></xsl:text>
      Номер           <xsl:value-of select ="@number"/> <xsl:text></xsl:text>

                       СПИСОК СТУДЕНТОВ

        <xsl:apply-templates></xsl:apply-templates>
    </xsl:template>

    <xsl:template match="Student">
        <xsl:value-of select ="@surname"/> <xsl:text> </xsl:text>
        <xsl:value-of select ="@name"/>    <xsl:text> </xsl:text>
        <xsl:value-of select ="@bday"/>    <xsl:text>г.р. </xsl:text>
        <xsl:apply-templates></xsl:apply-templates>
    </xsl:template>

</xsl:stylesheet>
```

XSLT

```
// TransformerFactory
try{
    FileInputStream xmlf = openFileInput("XMLTest.xml");
    FileInputStream xslf = openFileInput("XSLTFile1.xslt");
    FileOutputStream txt = openFileOutput("result.txt", MODE_PRIVATE);
    TransformerFactory fry = TransformerFactory.newInstance();
    Source xsltsrc = new StreamSource(xslf);
    Source xmlsrc =  new StreamSource(xmlf);
    Transformer t = fry.newTransformer(xsltsrc);
    t.transform(xmlsrc, new StreamResult(txt));
    int k = 1;
}

catch(IOException e){
    Log.d("Lab03XMLXPath:", e.getMessage());
}
catch (TransformerException e){

    Log.d("Lab03XMLXPath:", e.getMessage());
}
```

Вопросы?