



NTI / PAA PROGRAMOVÁNÍ MOBILNÍCH APLIKACÍ

7. Výměna dat – XML, JSON

Ing. Igor Kopetschke – TUL, NTI

<http://www.nti.tul.cz>



Android – výměna dat

- Výměna dat a práce s nimi v rámci aplikace není omezena pouze na SQLite aj.
- Mohou nastat následující scénáře:
 - Práce s daty uvnitř jedné aplikace
 - Výměna dat mezi aplikacemi
 - Výměna dat s externími systémy
- V této kapitole se zaměříme na tzv. strojově čitelná strukturovaná data
- Data v textově čitelné podobě
- Nejčastější formáty:
 - XML
 - JSON



Android – XML

- Oblíbený standardizovaný formát pro výměnu dat
- Jeho nevýhoda – značná redundance
- Výhoda – podpora snad na všech myslitelných platformách, programovacích jazycích a v mnoha aplikacích
- Android podporuje pro parsování XML dat standardní mechanismy převzaté z Javy
 - SAX parser API
 - DOM parser API
- Navíc nabízí vlastní doporučené řešení
- Rozhraní **XmlPullParser**
 - ve srovnání s DOM a SAX velice jednoduché
 - optimalizovaná implementace pro Android
 - rychlá, šetrná k paměti




Android – XmlPullParser

- Není nutno vytvářet vlastní implementaci rozhraní
`XmlPullParserFactory factory;`
`factory = XmlPullParserFactory.newInstance();`
`XmlPullParser xpp = factory.newPullParser();`
`... nebo ...`
`xpp = Xml.newPullParser(); //android.util.Xml`
- Následně určit zdroj XML dat
`xpp.setInput(input);`
- `input` může být jakýkoli
 - `FileReader`, `InputStream`, `StringReader`, `BufferedReader`...
- Tímto je zahájeno načítání a parsování XML dat
- Celý proces parsování je řízen událostmi (**events**)



Android – XmlPullParser

- Jednotlivé typy událostí (**int eventType**) jsou
 - **START_DOCUMENT**
 - **START_TAG** - `String getName()`
 - **TEXT** - `String getText()`
 - **END_TAG** - `String getName()`
 - **END_DOCUMENT**
- Jednotlivé **eventType** se získávají postupným procházením XML dat
 - `eventType = xpp.next()`
- Přístup k atributům elementů
 - `int getAttributeCount()`
 - `String getAttributeName(int index)`
 - `String getAttributeValue(int index)`
 - `String getAttributeType(int index)`



```
while (eventType != XmlPullParser.END_DOCUMENT) {  
    if(eventType == XmlPullParser.START_DOCUMENT) {  
        System.out.println("Start document");  
    } else if(eventType == XmlPullParser.START_TAG) {  
        System.out.println("Start tag " + xpp.getName());  
    } else if(eventType == XmlPullParser.END_TAG) {  
        System.out.println("End tag " + xpp.getName());  
    } else if(eventType == XmlPullParser.TEXT) {  
        System.out.println("Text " + xpp.getText());  
    }  
    eventType = xpp.next();  
}  
System.out.println("End document");
```



Android – XML

- Pro vytváření XML dat Android slouží implementace rozhraní **XMLSerializer**
- Není nutno implementovat vlastní
`XmlSerializer s = Xml.newSerializer();`
- Poté stačí :
 - `setOutput(writer)` – specifikace výstupu
 - `startDocument() / endDocument()`
 - `startTag() / endTag()`
 - `attribute(...)`
 - `text(...)`
 - `writer.toString()`



```
private String writeXml(List<Message> messages){
    XmlSerializer serializer = Xml.newSerializer();
    StringWriter writer = new StringWriter();
    try {
        serializer.setOutput(writer);
        serializer.startDocument("UTF-8", true);
        serializer.startTag("", "messages");
        serializer.attribute("", "number", String.valueOf(messages.size()));
        for (Message msg: messages){
            serializer.startTag("", "message");
            serializer.attribute("", "date", msg.getDate());
            serializer.startTag("", "title");
            serializer.text(msg.getTitle());
            serializer.endTag("", "title");
            serializer.startTag("", "url");
            serializer.text(msg.getLink().toExternalForm());
            serializer.endTag("", "url");
            serializer.startTag("", "body");
            serializer.text(msg.getDescription());
            serializer.endTag("", "body");
            serializer.endTag("", "message");
        }
        serializer.endTag("", "messages");
        serializer.endDocument();
        return writer.toString();
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```




Android – JSON

- Moderní formát pro výměnu dat
- Výhodou – velmi malá redundance
- Vytváření JSON řetězců
 - JSONObject – sada mapování klíč / hodnota
 - JSONArray – indexované pole hodnot
 - JsonWriter – helper pro vytváření JSON včetně výstupního proudu
- Parsování JSON řetězců
 - JSONTokener – parsuje JSON ze zadaného řetězce, umí JSONObject
 - JsonReader – parsuje JSON ze vstupního proudu



Android – JSONObject

- Zapouzdřuje "objekt", obecně dvojice klíč / hodnota
- Klíčem je řetězec (String)
- Hodnoty
 - int, long, double, boolean, String, NULL (ne null)
 - JSONObject, JSONArray
 - pro jednotlivé datové typy metody getXXX(), putXXX()
- Další užitečné metody
 - has(String key) – test na existenci klíče key
 - keys() – vrací Iterator klíčů
 - names() – vrací JSONArray klíčů
 - length() – počet dvojic key / value
 - remove(String key) – odstranění key a jeho hodnoty
 - isNull(String key) – test je-li hodnota klíče NULL
 - toString() – vrací JSON řetězec



Android – JSONObject

```
public void writeJSON() {  
    JSONObject object = new JSONObject();  
    try {  
        object.put("name", "Jack Hack");  
        object.put("score", new Integer(200));  
        object.put("current", new Double(152.32));  
        object.put("nickname", "Hacker");  
    } catch (JSONException e) {  
        e.printStackTrace();  
    }  
    System.out.println(object);  
}
```



Android – JSONArray

- Reprezentuje indexované JSON pole
- Hodnoty
 - int, long, double, boolean, String, NULL i null
 - JSONObject, JSONArray
 - serializovatelné Object
 - pro jednotlivé datové typy metody getXXX(), put()
- Další užitečné metody
 - length() – počet hodnot v poli
 - remove(int index) – odstranění hodnoty na indexu
 - isNull(int index) – test je-li hodnota na indexu NULL nebo null
 - toString() – vrací JSON řetězec



Android – JSONArray

```
JSONArray jsonArr = new JSONArray();

for (PhoneNumber pn : person.getPhoneList() ) {
    JSONObject pnObj = new JSONObject();
    pnObj.put("num", pn.getNumber());
    pnObj.put("type", pn.getType());
    jsonArr.put(pnObj);
}

jsonObj.put("phoneNumber", jsonArr);

return jsonObj.toString();
```



Android – JsonWriter

- Helper třída pro vytváření JSON a zápisu na Writer
- V konstruktoru předat instanci požadovaného Writeru
- Každý JSON dokument musí obsahovat jeden top-level JSONObject nebo JSONArray
- Vytvoření JSONObject
 - beginObject()
 - name(key), value(hodnota)
 - endObject()
- Vytvoření JSONArray
 - beginArray()
 - value(hodnota)
 - endArray()
- Uzavřít JsonWriter – close()



Android – JsonReader

- Helper třída pro parsování JSON z Reader streamu
- V konstruktoru předat instanci požadovaného Readeru
- Rekurzivně parsuje dokument
- Parsování JSONObject
 - beginObject()
 - nextLong(), nextInt(), nextBoolean(), nextString(), nextName() ...
 - endObject()
- Parsování JSONArray
 - beginArray()
 - nextLong(), nextInt(), nextBoolean(), nextString(), nextNull() ...
 - endArray()
- Možnost přeskočit – skipValue()
- Uzavřít JsonWriter – close()



Použité a doporučené zdroje

- <http://developer.android.com/>
- <http://www.zdrojak.cz/serialy/vyvijime-pro-android/>
- <http://www.itnetwork.cz/java/android>
- <https://users.fit.cvut.cz/cermaond/dokuwiki>
- Google...



.. A to je pro dnešek vše

DĚKUJI ZA POZORNOST