Android

Desenvolvimento de Software e Sistemas Móveis (DSSMV)
Licenciatura em Engenharia de Telecomunicações e Informática
LETI/ISEP

2025/26

Paulo Baltarejo Sousa
pbs@isep.ipp.pt

Disclaimer

Material and Slides

Some of the material/slides are adapted from various:

- Presentations found on the internet;
- Books;
- Web sites;
- ...

Outline

Permissions

Overview (I)

- To maintain security for the system and users, **Android requires apps to request permission** before the apps can use certain system data and features.
  - Because each Android app operates in a process sandbox, apps must explicitly request access to resources and data outside their sandbox.
  - They **request this access by declaring the permissions they need for additional capabilities** not provided by the basic sandbox.
- Requested permissions must be in the app **manifest** file.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.android.app.myapp" >
<uses-permission android:name="android.permission.RECEIVE_SMS" />
...
</manifest>
```

Overview (II)

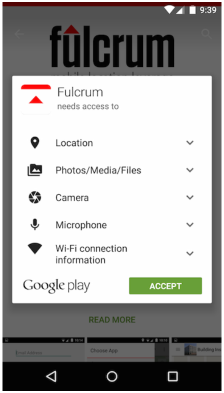
- Depending on **how sensitive the area is**, the system **may grant the permission automatically**, or it **may ask the user to approve the request**.
    - If your app **lists normal permissions in its manifest** (that is, permissions that don't pose much risk to the user's privacy or the device operation), **the system automatically grants** those permissions.
    - If your app **lists dangerous permissions in its manifest** (that is, permissions that could potentially affect the user's privacy or the device's normal operation), **the system asks the user** to explicitly grant those permissions.

Overview (III)

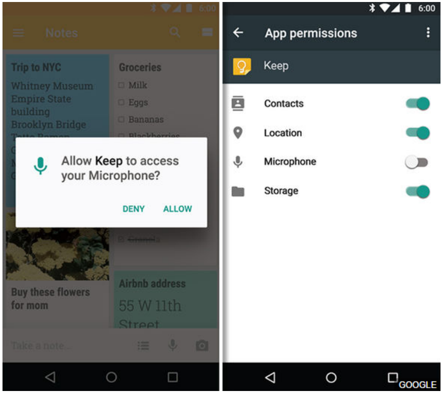
- The way Android **makes the requests depends on the system version**, and the system version targeted by your app:
  - If the device is running **Android 6.0 (API level 23) or higher**, and the app's `targetSdkVersion` is 23 or higher, the app requests permissions from the user at run-time.
    - The user can revoke the permissions at any time, so the app needs to check whether it has the permissions every time it accesses permission-protected APIs.
  - If the device is running **Android 5.1.1 (API level 22) or lower**, or the app's `targetSdkVersion` is 22 or lower, the system asks the user to grant the permissions when the user installs the app.
    - Once the user installs the app, the only way they can revoke the permission is by uninstalling the app.

Overview (IV)

- API level 22 or lower



- API level 23 or higher



Check For Permissions

- If your app needs a **dangerous permission**, you **must check whether you have that permission every time you perform an operation that requires that permission**.
- To check if you have a permission, call the `ContextCompat.checkSelfPermission()` method.

```
if(ActivityCompat.checkSelfPermission(MainActivity.this,Manifest.permission.
      READ_CONTACTS)!=PackageManager.PERMISSION_GRANTED){
//Permission not allowed
}else{
//Permission allowed
}
```

Request Runtime Permissions

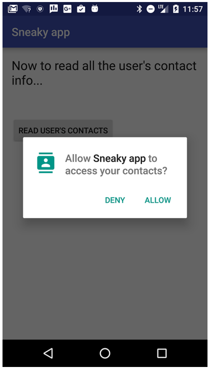
- If your app does not already have the permission it needs, the app must call the `registerForActivityResult` methods along with `ActivityResultContracts.RequestMultiple Permissions` to request the appropriate permissions.

```
ActivityResultLauncher<String[]> launcher = (ActivityResultLauncher<String[]>)
      registerForActivityResult(
new ActivityResultContracts.RequestMultiplePermissions(),
new ActivityResultCallback<Map<String, Boolean>>() {
  @Override
  public void onActivityResult(Map<String, Boolean> result) {
    ...
  }
});
```

Handle the Permissions Request Response

- When your app requests permissions, **the system presents a dialog box to the user**.
- When the user responds, **the system invokes your app's `onActivityResult` method**, passing it the user response as a `Map<String, Boolean>` collection.

```
@Override
public void onActivityResult(Map<String, Boolean> result) {
  boolean allGranted = true;
  for( Map.Entry<String,Boolean> entry : result.entrySet()){
    if(entry.getValue() == false){
      allGranted = false;
    }
  }
  if( ! allGranted ){
    finish();
  }
}
```

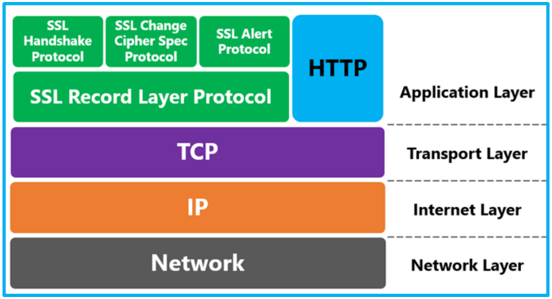

11:57
Sneaky app
Now to read all the user's contact info...
READ USER'S CONTACTS
Allow Sneaky app to access your contacts?
DENY
ALLOW

Hypertext Transfer Protocol (HTTP)

Hypertext Transfer Protocol (HTTP)(I)

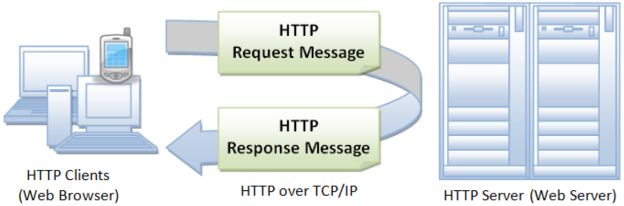
- HTTP (Hypertext Transfer Protocol) is perhaps the most popular application protocol used in the Internet.

- It is an **application layer** protocol.



HTTP (II)

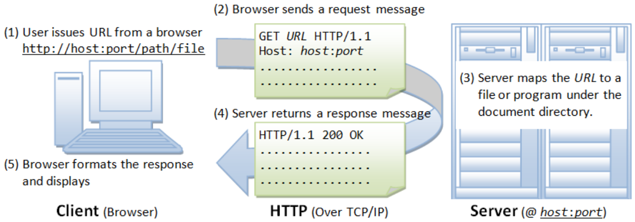
- It is an **asymmetric request-response client-server** protocol.



  - A web client (web browser) sends a request message to a web server to view a web page.
  - The web server receives that request and sends a response containing the web page information back to the web client.

HTTP (III)

- Whenever you issue a **Uniform Resource Locator (URL)** from your browser to get a web resource using HTTP, e.g. `http://www.aaaa.com/index.html`, the **browser turns the URL into a request message and sends it to the HTTP server**.
- The HTTP **server interprets the request message, and returns you an appropriate response message**, which is either the resource you requested or an error message.


(1) User issues URL from a browser
http://host:port/path/file
(2) Browser sends a request message
GET URL HTTP/1.1
Host: host:port
(3) Server maps the URL to a file or program under the document directory.
(4) Server returns a response message
HTTP/1.1 200 OK
(5) Browser formats the response and displays
Client (Browser)
HTTP (Over TCP/IP)
Server (@ host:port)

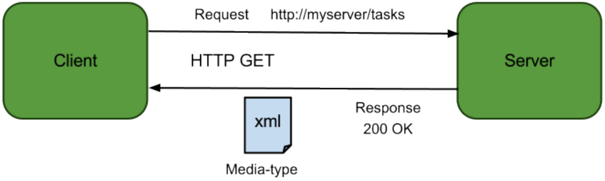
HTTP (IV)

- HTTP protocol defines a set of **request methods**.
  - **GET**: A client can use the GET request to get a web resource from the server.
  - **POST**: Used to post data up to the web server (store data on the server).
  - **PUT**: Ask the server to update data stored on the server.
  - **DELETE**: Ask the server to delete the data stored on the server.
  - ...
- A web client can use one of these request methods to send a request message to an HTTP server.

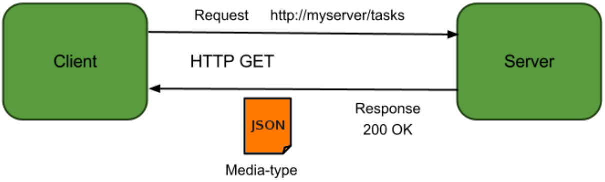
Data Interchange

Data Format Exchange

- **eXtensible Markup Language (XML)**



- **JavaScript Object Notation (JSON)**



Formats

- The XML and JSON.
    - Both are the two most common formats for data interchange in the Web today.

- XML

```
<employees>
  <employee>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName>
    <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName>
    <lastName>Jones</lastName>
  </employee>
</employees>
```

- JSON

```
{"employees":
  [
    { "firstName":"John", "lastName":"Doe" },
    { "firstName":"Anna", "lastName":"Smith" },
    { "firstName":"Peter", "lastName":"Jones" }
  ]
}
```

XML

- XML, is the functional cousin to HTML.
    - Where **HTML is responsible for displaying data in a human-readable format** in a Web browser, for example, **(machine-to-human)**
    - **XML is responsible for representing the structure of that data before it is transported from one system to another (machine-to-machine)**.
- XML is well-defined, widely supported and clearly structured.

JSON

- XML has worked and worked well in many different situations, but, in most cases, **JSON is now the preferred means of data marshalling**.
    - Marshalling is the process of transforming the memory representation of an object to a data format suitable for storage or transmission over network.
- The biggest reason that JSON is now being used over XML is that **JSON is inherently more efficient**.

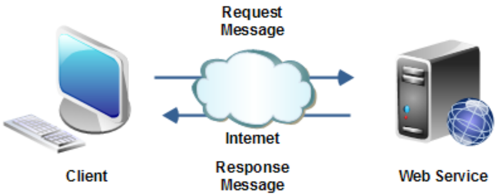
Web Services

What Are Web Services? (I)

- A **web service** is a service offered by an electronic device (such as computers) to another electronic device, communicating with each other via the World Wide Web (WWW), namely via HTTP.
    - **Web services are client and server applications that communicate over HTTP**.
- In a web service, web protocols such as HTTP, originally designed for **human-to-machine** communication, are utilized for **machine-to-machine** communication, more specifically for transferring machine readable file formats such as XML and JSON.
    - Web services **provide a standard means of interoperating** between software applications running on a variety of platforms and frameworks.

What Are Web Services? (II)

- A **web service is a way for two machines** to communicate with each other over a network.
  - A **web server** running on a computer **listens for requests** from other computers.
  - When a **request message from another computer is received**, over a network, the web service returns a **response message** with the requested resources.
    - This resource could be JSON, XML, an HTML file, Images, Audio Files, etc.

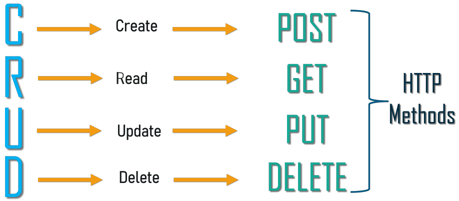


Types of Web Services

- Simple Object Access Protocol (SOAP) web services (will be no covered).
- **Representational State Transfer** (RESTful) web services.
    - REST defines a set of architectural principles:
        - Use HTTP methods explicitly.
        - Be stateless.
        - Expose directory structure-like URIs.
        - Transfer XML, JSON, or both.

RESTful Web services

- RESTful uses **HTTP methods** explicitly and in a way that's consistent with the protocol definition.
- This basic REST design principle establishes a one-to-one mapping between **C**reate, **R**ead, **U**pdate, and **D**elete (**CRUD**) operations and HTTP methods.
  - To create a resource on the server, use POST.
  - To retrieve a resource, use GET.
  - To change the state of a resource or to update it, use PUT.
  - To remove or delete a resource, use DELETE.



Networking

`HttpUrlConnection`

- `HttpUrlConnection` used for communicate between android application with outside resources data.
  - It is `URLConnection` with support for HTTP-specific features
  - Each `HttpUrlConnection` instance is used to make a single request
- `HttpUrlConnection` cannot run in UI Thread
  - It has to be executed by a worker thread or in background approach.

Work-flow `HttpUrlConnection`

1. Create an URL
2. Open an URL connection
3. Check if it is an HTTP connection
4. Configure the HTTP request
    - Method
    - Headers
    - Body (if any)
        - Writing to the connection `OutputStream`
5. Send the HTTP request
6. Receive the HTTP response
    - Get Status code
    - Get Body content (if any)
        - Reading from the connection `InputStream`
7. Close connection

Example: GET

```
public static String get(String urlStr) {
  String body = null;
  InputStream in = null;
  HttpURLConnection httpConn = null;
  int resCode = -1;
  try {
    URL url = new URL(urlStr);
    URLConnection urlConn = url.openConnection();
    if (!(urlConn instanceof HttpURLConnection)) {
      throw new IOException("URL is not an Http URL");
    }
    httpConn = (HttpURLConnection) urlConn;
    httpConn.setRequestMethod("GET");
    httpConn.connect();
    resCode = httpConn.getResponseCode();
    if (resCode == HttpURLConnection.HTTP_OK) {
      in = httpConn.getInputStream();
      body = readBody(in);
    }
  }catch (MalformedURLException e) {e.printStackTrace();
  }catch (IOException e) {e.printStackTrace();
  }finally {
    if(httpConn != null)
      httpConn.disconnect();
  }
  return body;
```

Example: POST

```
public static int post(String urlStr,String data) {
 OutputStream out = null;
 int resCode = -1;
 HttpURLConnection httpConn=null;
 try {
  URL url = new URL(urlStr);
  URLConnection urlConn = url.openConnection();
  if (!(urlConn instanceof HttpURLConnection)) {
   throw new IOException("URL is not an Http URL");
  }
  httpConn = (HttpURLConnection) urlConn;
  httpConn.setRequestMethod("POST");
  httpConn.setRequestProperty("Content-Type", "application/xml");
  out = httpConn.getOutputStream();
  writeBody(out, data);
  httpConn.connect();
  resCode = httpConn.getResponseCode();
 }catch (MalformedURLException e) {e.printStackTrace();
 }catch (IOException e) {e.printStackTrace();
 }finally {
  if(httpConn != null)
   httpConn.disconnect();
 }
 return resCode;
}
```

HTTP packet: Read and write body

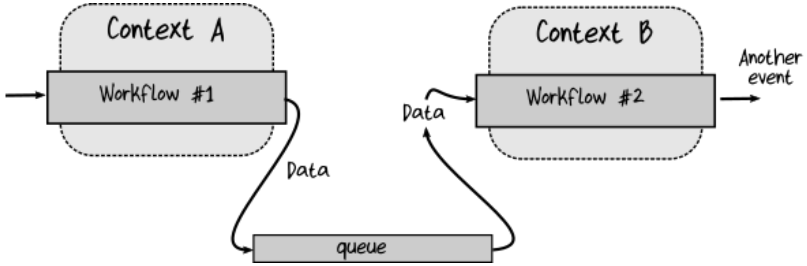
```
String readBody(InputStream in){
 StringBuilder sb = new StringBuilder();
 BufferedReader br = new BufferedReader(new InputStreamReader(in));
 try {
  String read = br.readLine();
  while(read !=null){
   sb.append(read);
   read = br.readLine();
  }
 }catch (IOException e) { e.printStackTrace(); }
 return sb.toString();
}
```

```
void writeBody(OutputStream writer, String body){
 try {
  byte[] dataBytes = body.getBytes("UTF-8");
  writer.write(dataBytes);
  writer.flush();
  writer.close();
 } catch (UnsupportedEncodingException e) {  e.printStackTrace();
 }catch (IOException e) {e.printStackTrace();}
}
```

Data Transfer Object (DTO)

Transferring Data

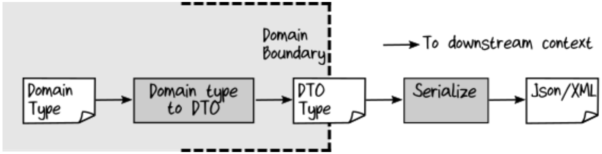
- Domain model must be known by other applications?



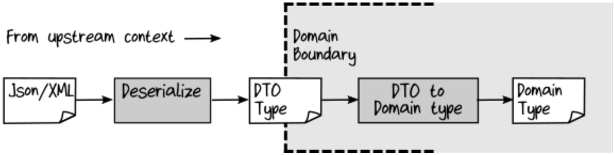
- (Answer) No. Domain model must be protected, hidden, unknown and ... from outside.
  - **Domain model is a secret**.
- But, it is required a shared (known) communication format?
  - (Answer) DTOs. **DTOs form a kind of contract** between bounded contexts.

DTOs as Contracts Between Bounded Contexts

- At the boundary of the upstream context then, the domain objects are converted into DTOs, which are in turn serialized into JSON, XML format:



- At the downstream context, the process is repeated in the other direction: the JSON or XML is deserialized into a DTO, which in turn is converted into a domain object:



What are DTOs?

- DTOs as in the **simple objects that carry data**, with no functionality at all.
- The difference between DTOs and domain (business) objects is that a **DTO does not have any behavior** except for serialization and deserialization of its own data.

Class for DTOs

- A DTO class must have:
  - Constructor with no parameter
  - Getters and setters for all attributes
- A DTO class cannot have:
  - Any business logic

```
public class PessoaDTO {
    private long nif;
    private String nome;
    private DataDTO nascimento;
    public PessoaDTO() {
    }
    public long getNif() {...}
    public void setNif(long nif) {...}
    public String getNome() {...}
    public void setNome(String nome) {...}
    public DataDTO getNascimento() {...}
    public void setNascimento(DataDTO
         nascimento) {...}
}
```

Bibliography

Resources

- "Mastering Android Application Development", by Antonio Pachon Rui, 2015
- `https://developer.android.com/index.html`
- `http://simple.sourceforge.net/home.php`
  `http://simple.sourceforge.net/download/stream/doc/tutorial/tutorial.php`