Android e Rest

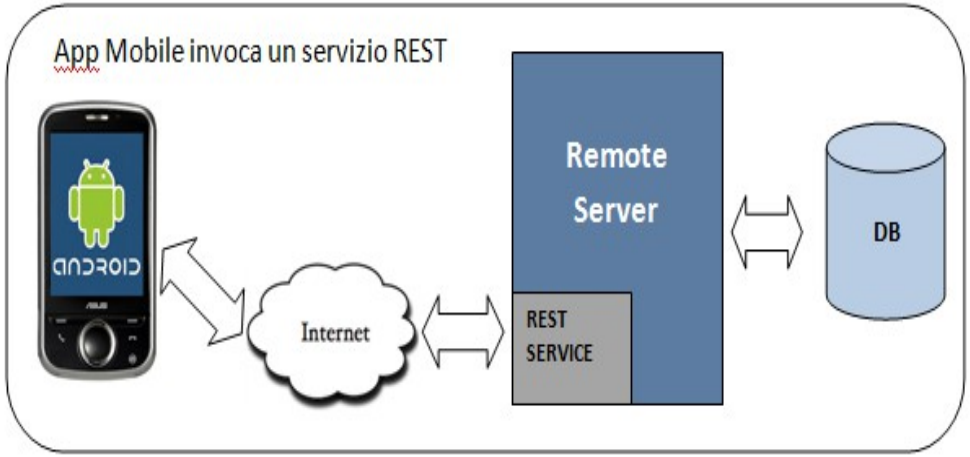
Introduzione a Rest

- SOAP è stato per tanto tempo sinonimo di WebService.
- Thomas Roy Fielding ha definito uno stile architetturale chiamato REpresentational State Transfer architecture, ovvero REST.
- Scopo: snellire l'architettura dei Web Services realizzando un modello più semplice da utilizzare
- Negli ultimi anni REST è emerso come design model predominante

I 4 principi di REST

- Utilizzo esplicito dei metodi HTTP
- Interazioni stateless
- I servizi sono strutturati con URI in forma di directory (Content Provider!!!)
- Scambio di informazioni in XML, JavaScript ObjectNotation (JSON) o entrambi

Rest



- Indipendente dalla piattaforma
- Indipendente dal linguaggio di programmazione
- Basato su un protocollo di comunicazione Standard (HTTP)
- Poichè usa la porta 80 (porta di default per traffico HTTP) non ha bisogno di particolari configurazioni del firewall

Rest VS SOAP

- Un WS Rest è incentrato sulla risorsa
- Un WS Soap è incentrato sul servizio
- Un WS Rest gestisce le risorse accessibili con tipiche chiamate http
- Un WS Soap espone dei metodi accessibile in remoto da parte di un client
- In Rest, interazioni complesse sulle risorse potrebbero richiedere maggiore attenzione e tempo rispetto ad approcci differenti.
- In Rest, si ha una riduzione della parte di configurazione e nella riduzione dei parametri di contollo

Operazioni CRUD e Http

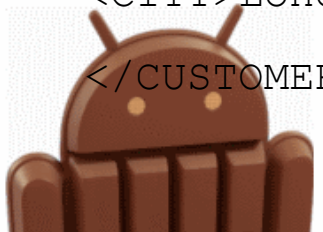
- REST definisce mapping tra operazioni CRUD e metodi HTTP:
  - Creazione di una risorsa sul server (C) → POST
  - Richiesta di trasferimento di una risorsa (R) → GET
  - Cambio di stato (aggiornamento) di una risorsa (U) → PUT
  - Cancellazione di una risorsa (D) → DELETE

URI

- La URI determina ciò che il Web Service è in grado di esporre
- REST risulta intuitivo nell'utilizzo di URI gerarchiche che espongono le caratteristiche principali dei servizi:

  `http://www.service.com/sqlrest/CUSTOMER/50`

```
<CUSTOMER xmlns:xlink="http://www.w3.org/1999/xlink">
<ID>50</ID>
<FIRSTNAME>Name</FIRSTNAME>
<LASTNAME>LastaName</LASTNAME>
<STREET/>
<CITY>London</CITY>
</CUSTOMER>
```



Es. Top Level resource

http://www.thomas-bayer.com/sqlrest/

produce

```
<resource xmlns:xlink="http://www.w3.org/1999/xlink">

<CUSTOMERList xlink:href="http://www.thomas-
bayer.com/sqlrest/CUSTOMER/">CUSTOMER</CUSTOMERList>

<INVOICEList xlink:href="http://www.thomas-
bayer.com/sqlrest/INVOICE/">INVOICE</INVOICEList>

<ITEMList xlink:href="http://www.thomas-
bayer.com/sqlrest/ITEM/">ITEM</ITEMList>

<PRODUCTList xlink:href="http://www.thomas-
bayer.com/sqlrest/PRODUCT/">PRODUCT</PRODUCTList>

</resource>
```

GET

- Ogni singolo oggetto è accessibile mediante uno specifico URL
- Per esempio è possibile ottenere i dati relativi al cliente 50 mediante la HTTP request:

```
GET /customer/50
```

- ottenendo:

```
<CUSTOMER
xmlns:xlink="http://www.w3.org/1999/xlink">
<ID>50</ID>
<FIRSTNAME>Name</FIRSTNAME>
<LASTNAME>LastName</LASTNAME>
<STREET/>
<CITY>London</CITY>
</CUSTOMER>
```

PUT

- Per creare una nuova risorsa si utilizza il metodo PUT

```
PUT /article


<articles>
  <name>SmartPhone</name>
  <description>
    Apple iPhone 5
  </description>
  <price>729</price>
  <unit>112 grams</unit>
</article>
```

PUT

- Il codice HTTP 201 significa che la creazione ha avuto successo.
- E' utilizzato un campo HTTP location header per indicare la locazione della risorsa creata.

HTTP/1.1 201 OK

Content-Type: text/xml;

Content-Length: 30

Location: http://myservice/product/50

Debugging Tools

- Sono disponibili molteplici tool per testare i WS Rest. Tra questi:
  - Chrome REST Console extension
  - Firefox Poster add-on
  - CURL (command line)
  - HTTP4e (Eclipse Plugin)

Android & Rest

- Riferimenti:
  - http://wolfpaulus.com/journal/mac/tomcat7
  - http://tomcat.apache.org/tomcat-7.0-doc/setup.html
  - http://avilyne.com/?p=105
  - `http://www.thomas-bayer.com/sqlrest/`
  - http://www.cheesejedi.com/rest_services/