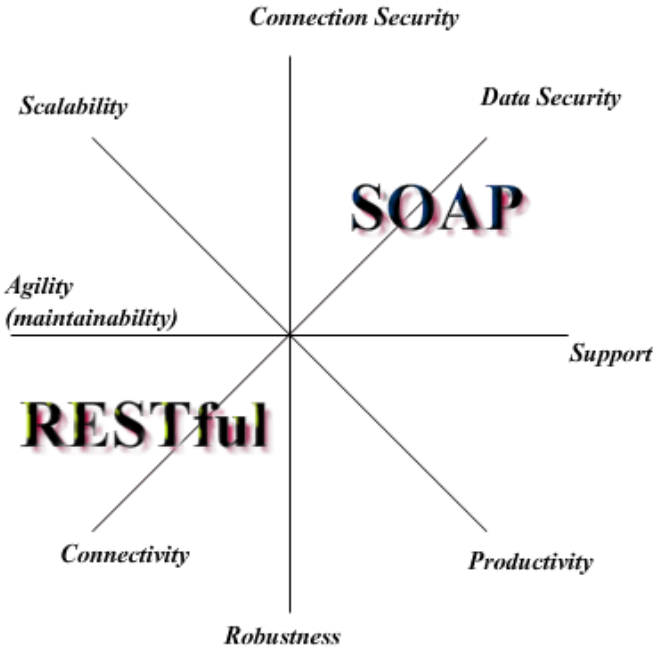
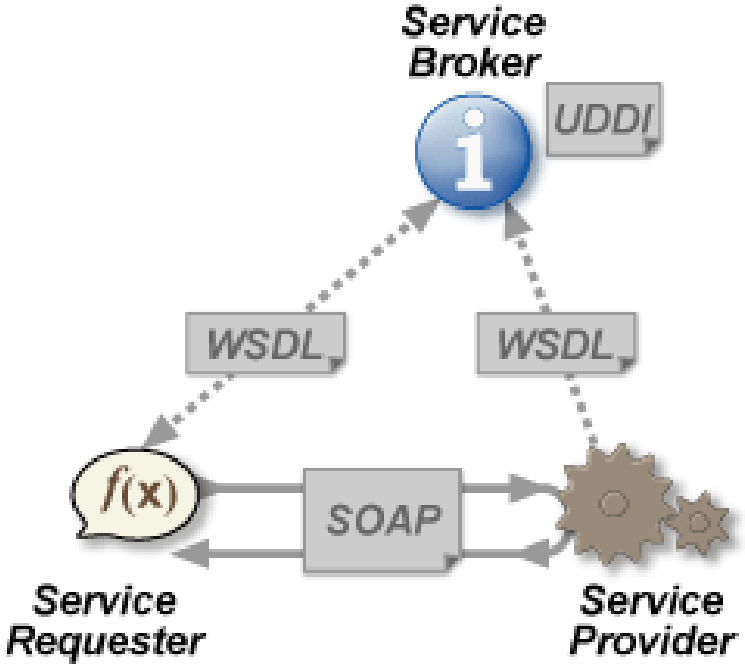
Web Services

Benoît Charroux

benoit.charroux@efrei.fr





Definition

- A Web Service is a program allowing communication and data exchange between applications and heterogeneous systems in a distributed environment:
- There are many technologies behind the term web service:
  - Representational State Transfert (REST):
    - Simple
    - Suitable for application that accesses to resources like databases…
  - The web service SOAP:
    - Complex
    - Suitable for Enterprise Information Systems (Service Oriented Architecture…)

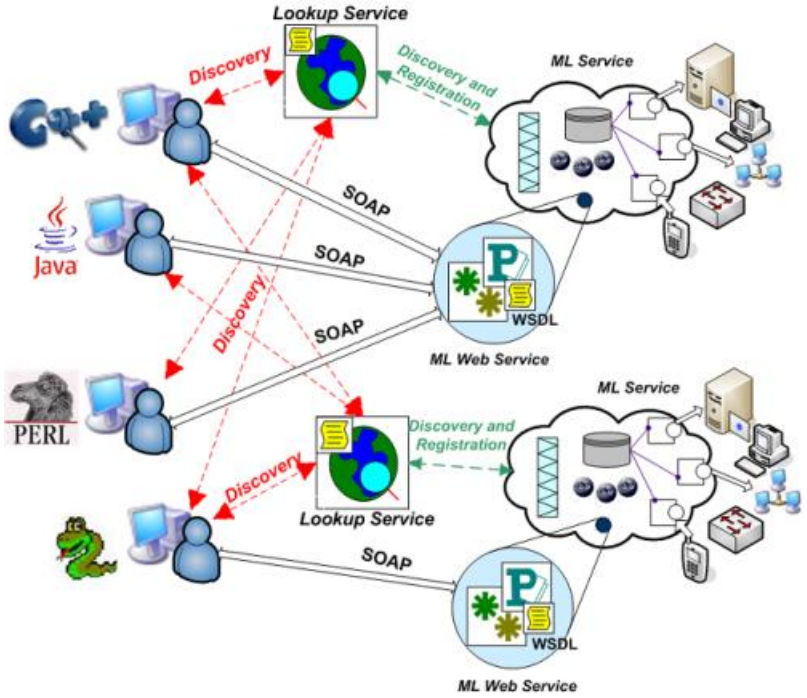
Web Services SOAP

Web Services : properties

- Interoperability (any languages, systems…) can be used
- Based on standard and open protocols (HTTP, SOAP…).
- Exchange data are textual data if possible.
- Can cross firewall.

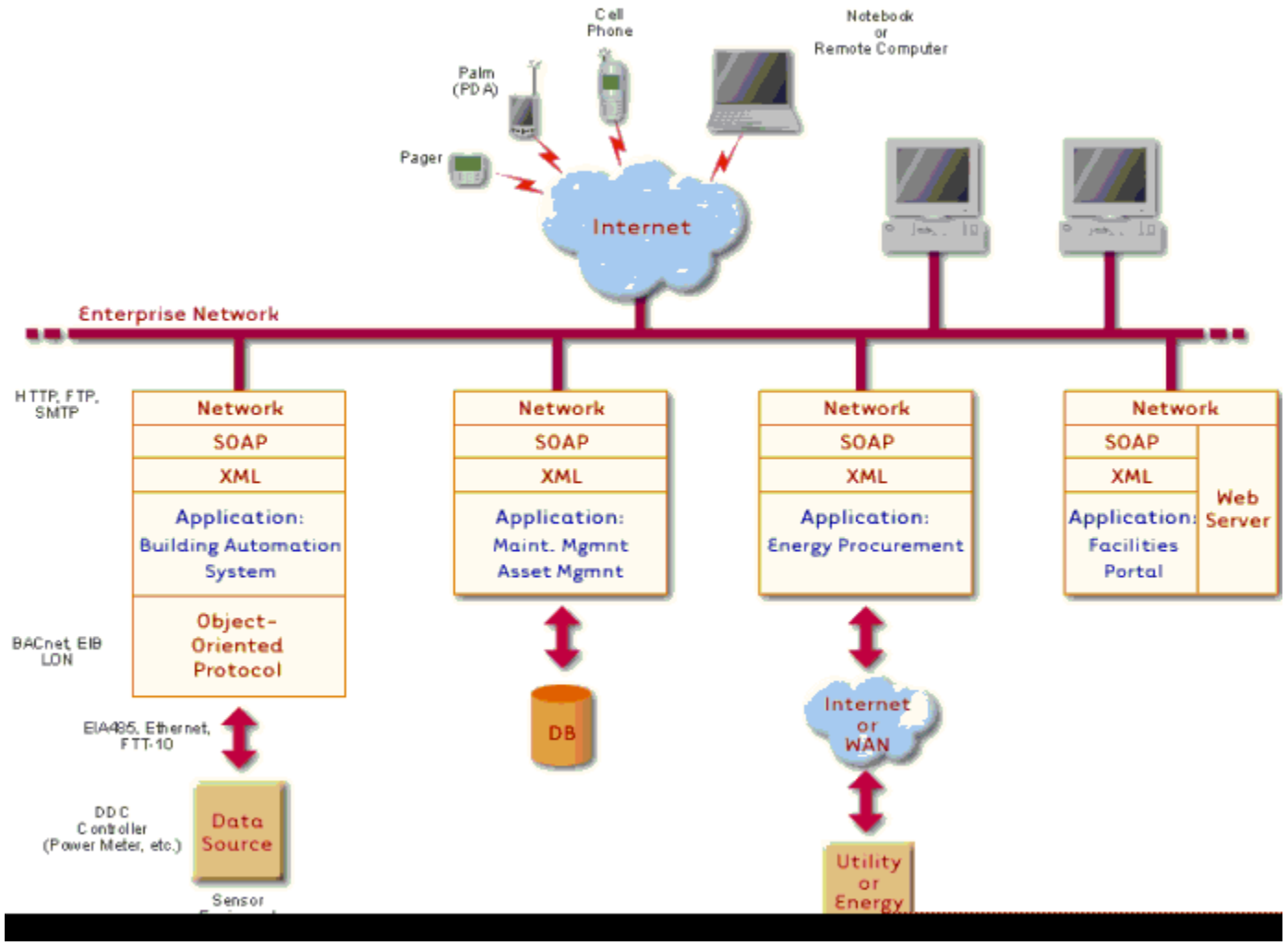
Interoperability

- Interoperability between languages, systems, platforms… :
  - XML is used for all exchanges
  - Simple Object Access Protocol (SOAP) composed of:
    - a set of encoding rules for datatypes
    - a convention procedure calls and responses
  - Registries of services (UDDI, DISCO)
  - Description of services (WSDL)



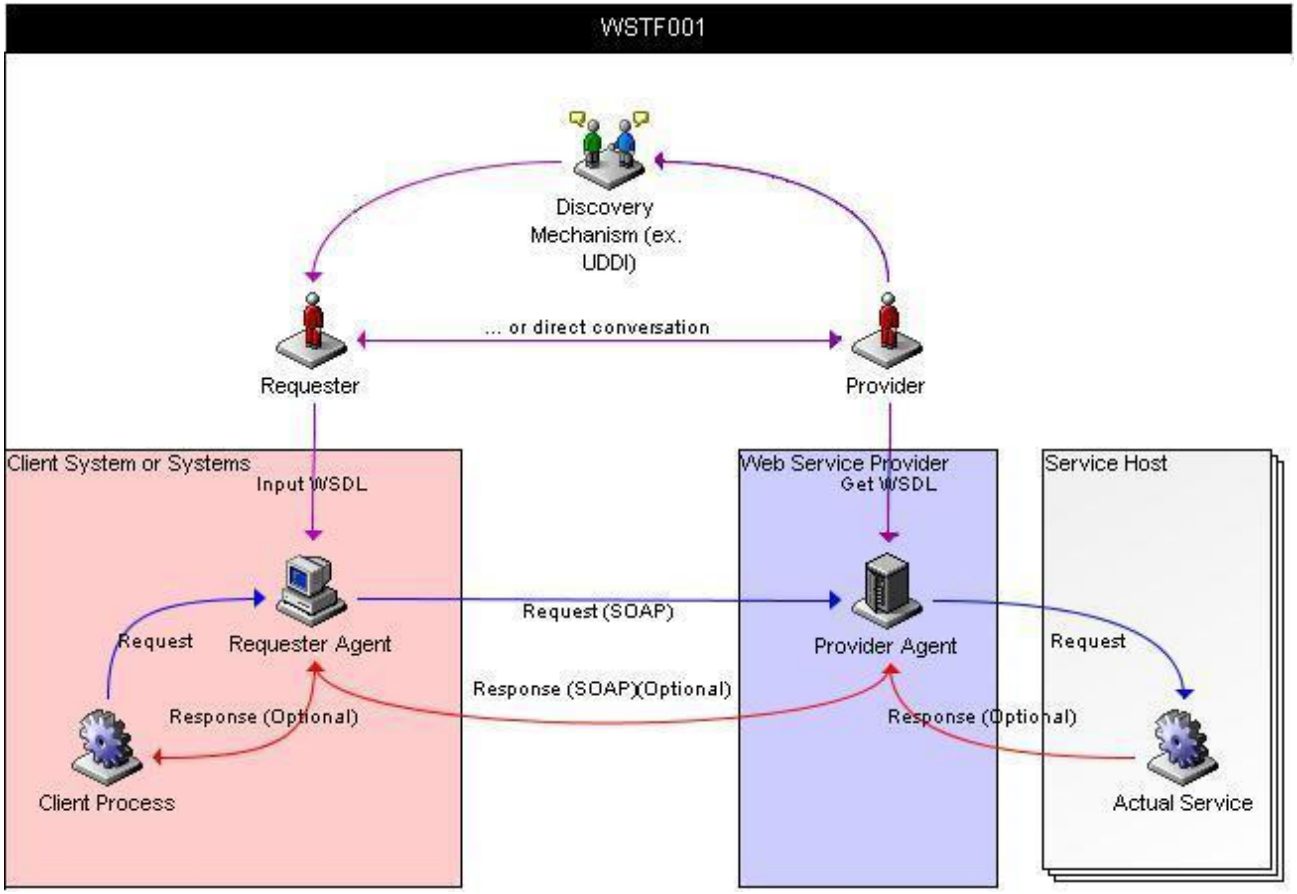
Web Services

- Principle:



Web Services

- Principle: a registry registers and locates each web service, it provides access to a description of the web service (the description written in WSDL comes along with the web service)



Web Services

- Architecture:

Découverte UDDI, DISCO
Description WSDL, XML, Schema, Docs
Format de message SOAP
Encodage XML
Transport HTTP, SMTP, etc

UDDI (Universal Description, Discovery and Integration) is a XML-based protocol to register and locate web services.

WSDL (**Web Services Description Language)** is a language (written in XML) describing the functionality offered by a web service.

SOAP (Simple Object Access Protocol) is a protocole for exchanging structured information in XML over HTTP (mainly).

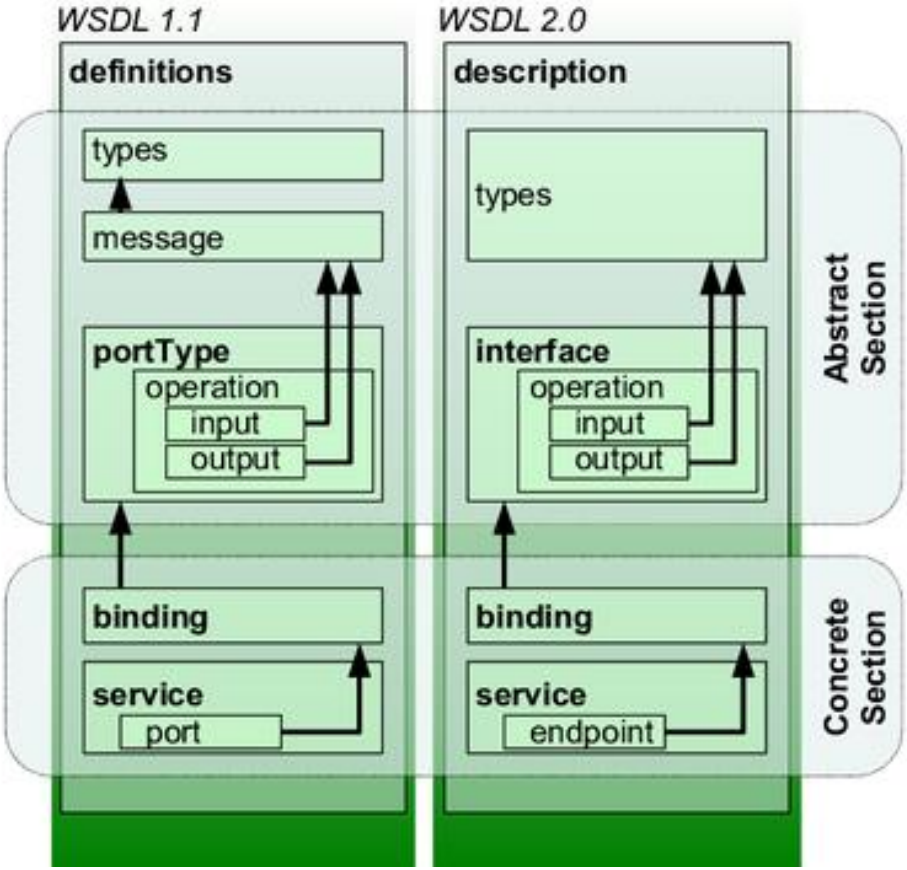
WSDL

Structure of a WSDL document

- WSDL is written in XML.
- WSDL is used to describe a service and its location.
- A client can generate code in any language to access a web service remotely

Element:	Definition:
<types>	Data types
<message>	Message to exchange data
<portType>	Operations computed by the service
<binding>	Communication protocols (SOAP…)

WSDL


WSDL 1.1
definitions
types
message
portType
operation
input
output
binding
service
port
WSDL 2.0
description
types
interface
operation
input
output
binding
service
endpoint
Abstract Section
Concrete Section

WSDL Port

- A port defines a Web Service in terms if operations and messages.

Operation type	Definition
One-way	The operation can receive a message but does not return a response
Request-response	The operation can receive a request and return a response
Solicit-response	The operation can send a request and wait for a response
Notification	The operation can send a message but does not wait for a response

Example of operations

- One-way :

```
<message name="newTermValues">
        <part name="term" type="xs:string"/>
        <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
</portType >
```

- Request-Response :

```
<message name="getTermRequest">
        <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
        <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
        <operation name="getTerm">
                <input message="getTermRequest"/>
                <output message="getTermResponse"/>
        </operation>
</portType>
```

WSDL Bindings

- WSDL binding defines message formats and protocols.
- Example of the binding SOAP:

```
<portType name="glossaryTerms">
        <operation name="getTerm">
                <input message="getTermRequest"/>
                <output message="getTermResponse"/>
        </operation>
</portType>

<binding type="glossaryTerms" name="b1">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
        <operation>
                <soap:operation soapAction="http://example.com/getTerm"/>
                <input><soap:body use="literal"/></input>
                <output><soap:body use="literal"/></output>
        </operation>
</binding>
```

UDDI

- UDDI : Universal Description, Discovery and Integration
- UDDI is a directory to store information about Web Services.
- UDDI is a directory composed of Web Services interfaces described in WSDL.
- UDDI communicates by SOAP.

Web service design

- When creating Web services, there are two development styles:
  - *Contract Last*: start with the Java code, and let the Web service contract (WSDL) be generated from that.
  - *Contract First:* start with the WSDL contract, and use Java to implement said contract.

Pro contact last approach: easy

Con contract last approach:

- Each time you change your Java contract and redeploy it, there might be subsequent changes to the web service contract.
- Additionally, not all SOAP stacks generate the same web service contract from a Java contract.
- When a web service contract changes, users of the contract will have to be instructed to obtain the new contract and potentially change their code to accommodate for any changes in the contract.
- Low performance

Contract first example

- Define data in XML:

```
<Holiday xmlns="http://mycompany.com/hr/schemas">
    <StartDate>2006-07-03</StartDate>
    <EndDate>2006-07-07</EndDate>
</Holiday>
```

```
<Employee xmlns="http://mycompany.com/hr/schemas">
    <Number>42</Number>
    <FirstName>Arjen</FirstName>
    <LastName>Poutsma</LastName>
</Employee>
```

Contract first example

- Define a request:

```
<HolidayRequest xmlns="http://mycompany.com/hr/schemas">
    <Holiday>
        <StartDate>2006-07-03</StartDate>
        <EndDate>2006-07-07</EndDate>
    </Holiday>
    <Employee>
        <Number>42</Number>
        <FirstName>Arjen</FirstName>
        <LastName>Poutsma</LastName>
    </Employee>
</HolidayRequest>
```

Contract first example

- Define the data contract:
- Four different ways:
  - DTDs: limited namespace support
  - XML Schema XSD
  - Relax NG: not widely supported across platforms
  - Schematron: not widely supported across platforms

Contract first example

- XML Schema:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:hr="http://mycompany.com/hr/schemas"
        elementFormDefault="qualified"
        targetNamespace="http://mycompany.com/hr/schemas">
    <xs:element name="HolidayRequest">
        <xs:complexType>
            <xs:all>
                <xs:element name="Holiday" type="hr:HolidayType"/>
                <xs:element name="Employee" type="hr:EmployeeType"/>
            </xs:all>
        </xs:complexType>
    </xs:element>
    <xs:complexType name="HolidayType">
        <xs:sequence>
            <xs:element name="StartDate" type="xs:date"/>
            <xs:element name="EndDate" type="xs:date"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="EmployeeType">
        <xs:sequence>
            <xs:element name="Number" type="xs:integer"/>
            <xs:element name="FirstName" type="xs:string"/>
            <xs:element name="LastName" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Contract first example

- Generate the WSDL from the XML Schema

```
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:schema="http://mycompany.com/hr/schemas"
                  xmlns:tns="http://mycompany.com/hr/definitions"
                  targetNamespace="http://mycompany.com/hr/definitions">
    <wsdl:types>
        <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:import namespace="http://mycompany.com/hr/schemas"
                schemaLocation="hr.xsd"/>
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="HolidayRequest">
        <wsdl:part element="schema:HolidayRequest" name="HolidayRequest"/>
    </wsdl:message>
    <wsdl:portType name="HumanResource">
        <wsdl:operation name="Holiday">
            <wsdl:input message="tns:HolidayRequest" name="HolidayRequest"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="HumanResourceBinding" type="tns:HumanResource">
        <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="Holiday">
            <soap:operation soapAction="http://mycompany.com/RequestHoliday"/>
            <wsdl:input name="HolidayRequest">
                <soap:body use="literal"/>
            </wsdl:input>
        </wsdl:operation>
```

REST Web Services

What is REST ?

- Representational State Transfert is an abstraction of the architecture of the World Wide Web
- Concept proposed by Roy Fielind en 2000
- Alternative to SOAP Web Services
- Designed to insure performance, scalability, simplicity, modifiability, visibility, portability and reliability.

RESTful

- RESTful is a Web Service API that respects the REST architecture:
  - Each resource has its own URI
  - Resource are represented using internet media type (often Json, but also WML, Atom…)
  - Communicate using the HTTP protocol (GET, PUT, POST, DELETE):
    - GET: send back data to the client from the server
    - PUT: update data to the server
    - POST: create data on the server side
    - DELETE: delete data on the server
  - Hypertext links to reference states and related resources, messages self-describing

Design a Rest application: a car rental service

- The resources: cars
- Resources URI: http://www.carentalservice.com/
- Protocole:
  - Get the list of cars to be rented:
    - URI: …/cars
    - http GET
    - Json response: [{"plateNumber" : "11AA22" , " nomberOfSeats" : 5}, { …}…]
  - Get the features of a car:
    - URI: …/cars/plateNumber
    - http GET
    - Json response: { " plateNumber" : "11AA22" , " nomberOfSeats" : 5, "price" : 100 }

Design a Rest application: a car rental service

- Protocole:
  - Rent a car:
    - URI: …/plateNumber?rent=true
    - http PUT
    - Send Json inside the http body: { "begin" : "11/11/2017" , " end" : "1/1/2018" }
  - Get back the car:
    - URI: …/plateNumber?rent=false
    - http PUT

A server side example using Spring

- Rent a car:
  - URI: …/plateNumber?rent=true
  - http PUT
  - Sent Json inside the http body: { "begin" : "11/11/2017" , " end" : "1/1/2018" }

```
@RequestMapping(value="/car/{plateNumber}", method=RequestMethod.PUT)
@ResponseStatus(HttpStatus.OK)
public void rentAndGetBack(
        @PathVariable("plateNumber") String plateNumber,
        @RequestParam(value="rent", required = true)boolean rent,
        @RequestBody Dates dates
        ) throws Exception{
        …
}
```

Properties of REST

- Stateless: each request from the client to the server must contain all information to understand the request without using any context stored on the server. The session management must be made at the client side.

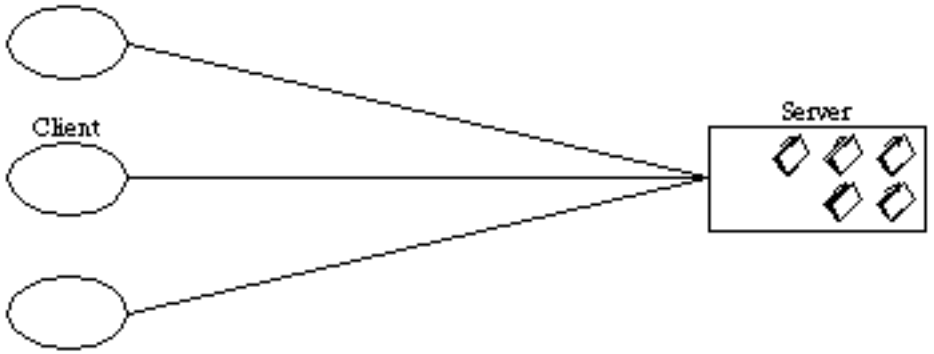


Figure 5-3. Client-Stateless-Server

Images de Thèse de Roy T. Fielding

Properties of REST

- Stateless with cache: client may cache data.

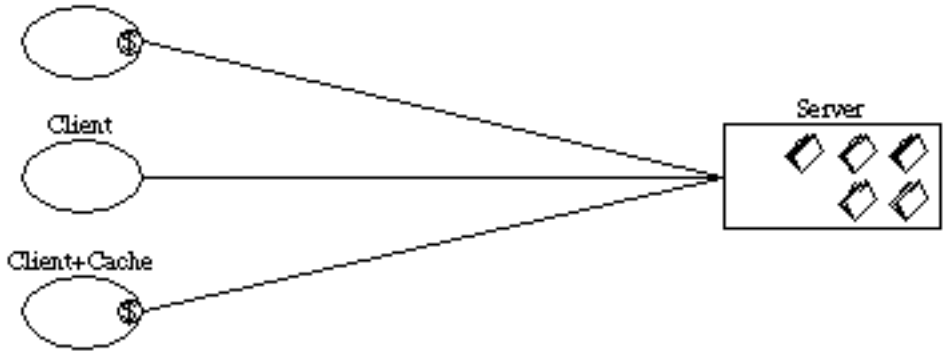


Figure 5-4. Client-Cache-Stateless-Server

Images de Thèse de Roy T. Fielding

Properties of REST

- Layered system: each component can't see beyond the layer it interact with:
  - Better independence
  - Possible to make load balancing in intermediate layer

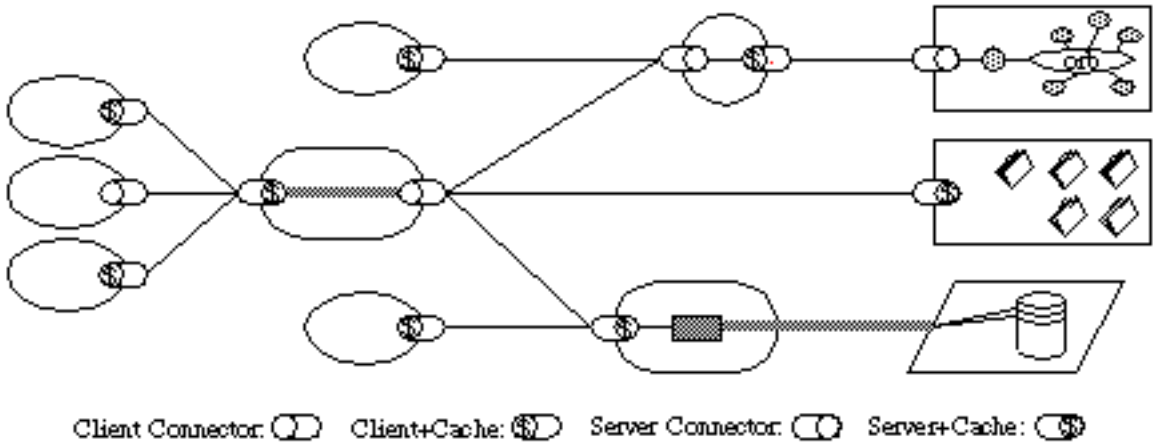


Figure 5-7. Uniform-Layered-Client-Cache-Stateless-Server

ages de Thèse de Roy T. Fielding

Properties of REST

- On demand code (optional): possibility to download and compute code on the client side

Limitation of REST

- limitations of REST:
  - Poor service description (unlike WSDL)
  - No discovery protocol (unlike UDDI or DISCO).