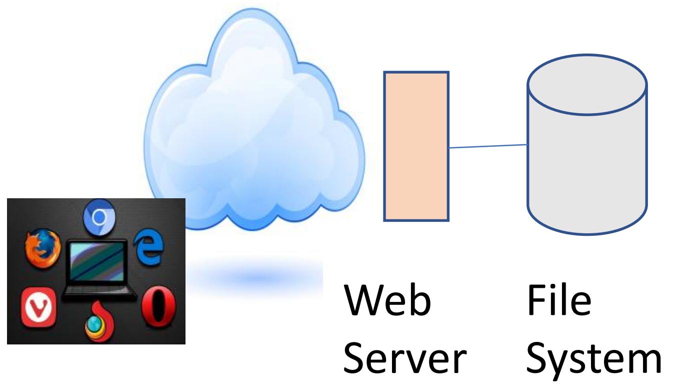
Software Platforms

LM in Computer Engineering

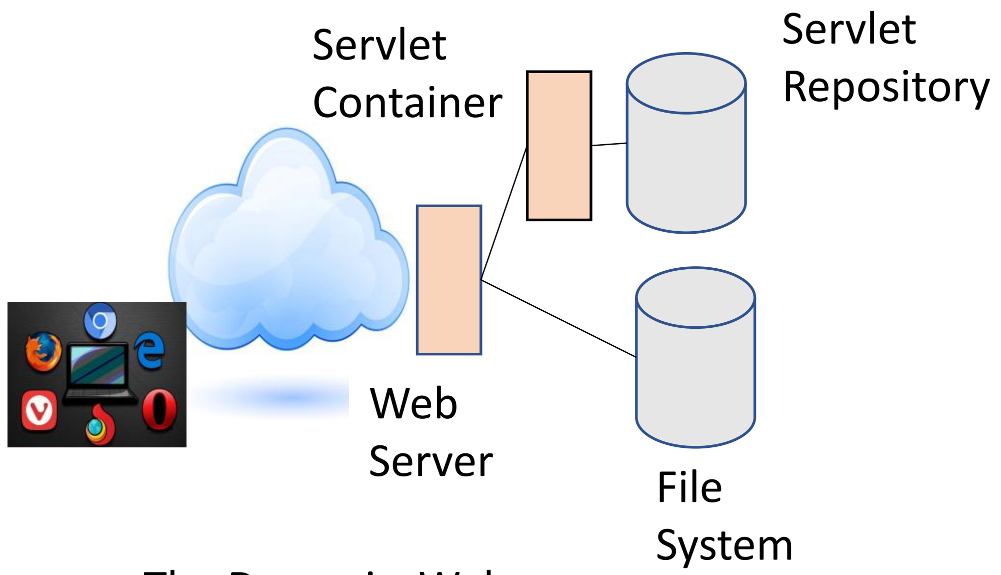
Massimo Maresca

The Web/Webapp paradigm

- Socket is the basic abstraction upon which Software Platforms made up of components are built.
- Any application can open a socket to accept a connection and connect to an open socket.



The Static Web

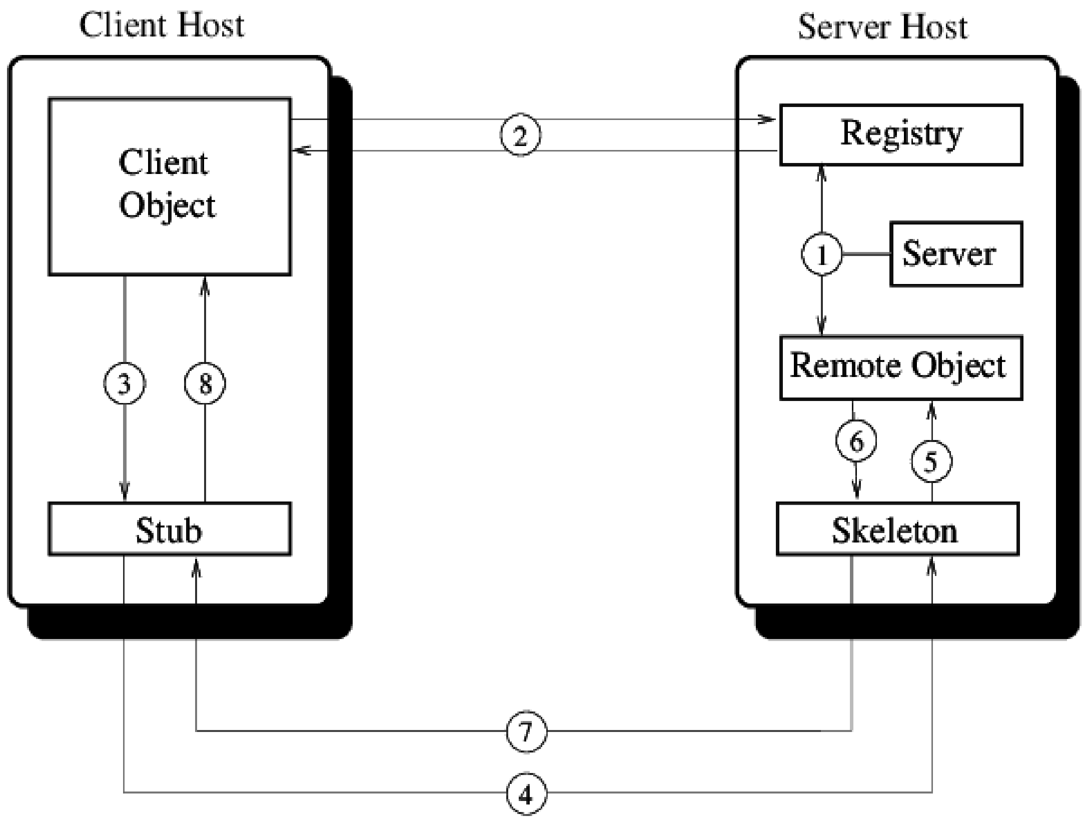


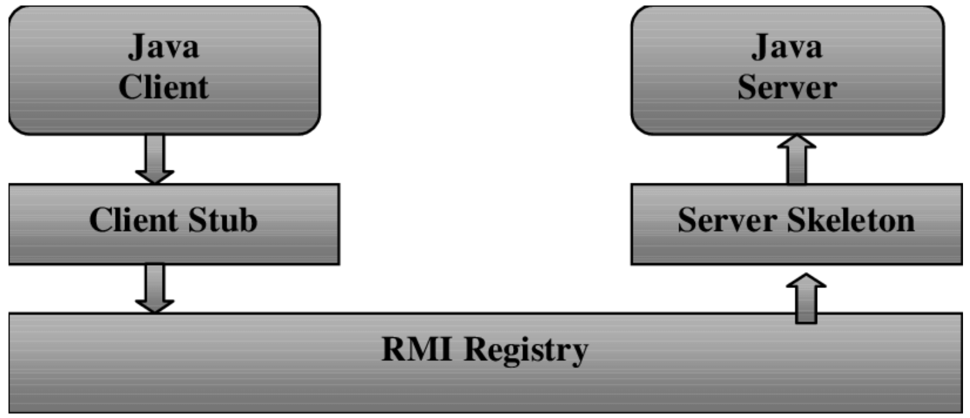
The Dynamic Web

Another Paradigm – Process to Process Communication

- The need to enable processor to communicate is very old. Different types of middleware were proposed and are still in use to enable such a functionality. Examples are:
  - CORBA (Common Object Request Broker Architecture) proposed by OMG to support interoperability
  - DCOM, mainly proposed and used in the Microsoft environment
  - RPC, mainly proposed and used in the Unix operating system
  - RMI, based on Java
- All these middlewares are built upon socket, as an infrastructure, on service publication and on protocols for data encapsulation.
- Data exchange refers to the transmission of parameters/results.

Remote Method Invocation





1. Server registers service and create the Remote Object
2. Client retrieves the remote object interface
3. Client calls the Client Stub
4. Stub calls the Server Skeleton
5. The Server Skeleton runs the Remote Object
6. The Remote Objects returns a result
7. The Server Skeleton gives back the result to the Client Stub
8. The Client Stub gives back the result to the Client

Example RMI – Refer to the attached code

RMI – Registry Launched by Server

- Service folder which includes:
  - The Service Interface, shared with the Client
  - The Service Implementation
  - The Server, which runs the Registry and makes the service available
- Client Folder which includes:
  - The Interface: shared with the Server
  - The Client implementation

RMI – Registry as an Autonomous Process

- Service folder which includes:
  - The Service Interface, shared with the Client
  - The Service Implementation
  - The Server, which runs the Registry and makes the service available
- Client Folder which includes:
  - The Interface: shared with the Server
  - The Client implementation

From RMI to Web Services

- Web Services were initially proposed as a standard and interoperable method to support inter-process communication (RMI style)
- The most evident differences between Remote Method Integration and RPC-Based Web Services are:
  - Language independency

    Web Services achieve language independency through XML. The point is that all data exchange is coded in XML, which so becomes the Web Service Lingua Franca.
  - Security

    Web Services achieve security through the interposition of an application server handling all the connections. Typically, all TCP ports are disabled or blocked while only the ports managed by secure front ends remain open.

Web Service types

- Web Service were initially proposed as a standard and interoperable method for process communication (RMI style)
- Currently Web Services appear in two forms, namely:
  - RPC based Web Services, based on SOAP/WSDL
    - This is the “old” category of Web Services, which enabled the so-called SOA (Service Oriented Architecture). A couple of decades of activity, implementation, deployment have led to a tremendous legacy. SOA is alive and used in products, companies and organizations.
  - REST based Web Services, based simple Web access
    - This is the “new ” category of Web Services, which enables the so called Microservice architecture.

SOAP Web Services: First of all XML

- XML is considered the universal format for structured documents and data on the Web (W3C)
- Structured document examples include spreadsheets, address books, financial transactions, configuration parameters, etc.
- XML resembles HTML. While HTML is mainly for rendering documents in browsers, XML just provides a structure and leaves the interpretation to applications.
- XML is textual, verbose and human readable. However it is not supposed to be read (which is exactly what we plan to do…)

XML Example: web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">
  <servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>Hello</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/hello</url-pattern>
  </servlet-mapping>
</web-app>
```

Web Services – Paradigm and Protocols

- Inter Process Communication (RPC style)
- **SOAP**: Simple Object Access Protocol
- **SOAP** is used to encapsulate the interactions among components running on different computer systems.
- **WSDL**: Web Services Description Language
- **WSDL** is used to describe the public interface of a Web Service.
- Upon Web Service publication the corresponding WSDL file is made available on the Web, so that clients can download it and use it.
- When building a client, the WSDL description must be used to generate automatic stub code for the client.

- See for example: https://www.w3schools.com/xml/xml_wsdl.asp ed
  https://www.w3schools.com/xml/xml_soap.asp

Web Services Implementation and Deployment

- Directly listening on specific ports. See examples on port 7654
  - Example #1: WS-1 Converter
    - In this case the service and the client share the definition of the common interface.
  - Example #2: WS-2 Converter-wsimport
    - In this case the client downloads the interface from the server.
    - Download wsimport
- Running inside a Service Container (e.g., Apache Axis 2) which in turn runs as a Web Applications in a Servlet Container (e.g., Apache Tomcat).
  - Example #3: WS-3-AdderonAxis
    - In this case the service run under Axis 2 under Tomcat.
      Download and install Apache Ant and Apache Axis 2 and follow the instructions.

WSDL – General Structure

Element	Description
<types>	Defines the (XML Schema) data types used by the web service
<message>	Defines the data elements for each operation
<portType>	Describes the operations that can be performed and the messages involved.
<binding>	Defines the protocol and data format for each port type

In XML

```
<definitions>
        <types>
                data type definitions........
        </types>
        <message>
                definition of the data being communicated....
        </message>
        <portType>
                set of operations......
        </portType>
        <binding>
                protocol and data format specification....
        </binding>
</definitions>
```

WSDL – portType

The <portType> element defines **a web service**, the **operations** that can be performed, and the **messages** that are involved.

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

WSDL – Converter Service Example

```
<types>
  <xsd:schema>
  <xsd:import namespace="http://swplatforms.unige.it/" schemaLocation="http://localhost:7654/Converter?xsd=1"/>
  </xsd:schema>
</types>
<message name="c2f">
  <part name="parameters" element="tns:c2f"/>
</message>
…
<portType name="ConverterInterface">
  <operation name="c2f">
    <input wsam:Action="http://swplatforms.unige.it/ConverterInterface/c2fRequest" message="tns:c2f"/>
    <output wsam:Action="http://swplatforms.unige.it/ConverterInterface/c2fResponse" message="tns:c2fResponse"/>
  </operation>
….
</portType>
```

WSDL – Converter Service Example

```
<binding name="ConverterPortBinding" type="tns:ConverterInterface">
 <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="c2f">
   <soap:operation soapAction=""/>
    <input>
     <soap:body use="literal"/>
    </input>
    <output>
     <soap:body use="literal"/>
    </output>
   </operation>
   </binding>
    <service name="ConverterService">
     <port name="ConverterPort" binding="tns:ConverterPortBinding">
                   <soap:address location="http://localhost:7654/Converter"/>
     </port>
```

Binding SOAP to HTTP

```
POST /item HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
```

SOAP Example - Request

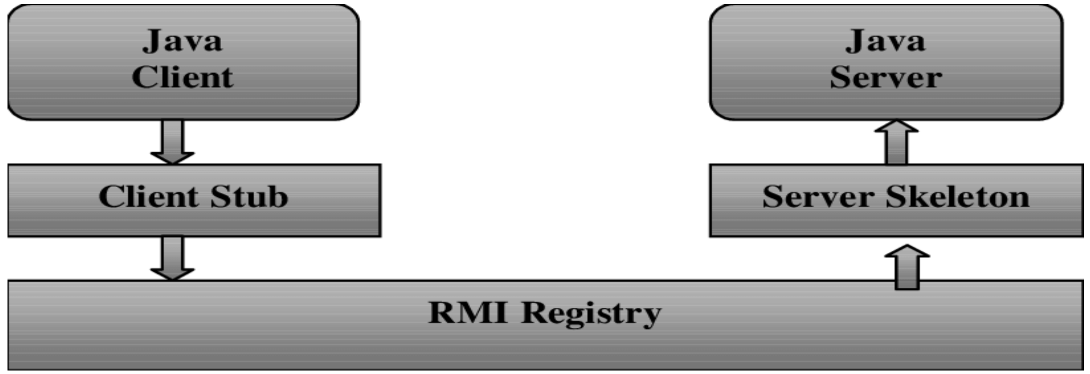
```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

SOAP Example - Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

Discussion (1)

- Inter Process Communication/Service Exchange
- Remote Procedure Call /Remote Method Invocation Paradigm
- Reference Architecture
  - Server exposes Services through a Skeleton
  - Client accesses Services through a Stub



- Issues:
  - Protocol Stack
  - Service Description
  - Data Encapsulation
  - Service Publication / Registration

Discussion (2)

- RMI: Service Exchange over TCP
  - Just TCP data streams
  - Technology Dependent Service Description
  - Data Encapsulation
  - Technology Dependent Registration

- Web Services: Service Exchange over the Web
  - Going to the application layer: Http - POST
  - Technology Independent Service Description: WSDL
  - Technology Independent Data Encapsulation: SOAP
  - Technology independent Registration: Servlet Container+Service Container