A leading institution at the heart of the digital society

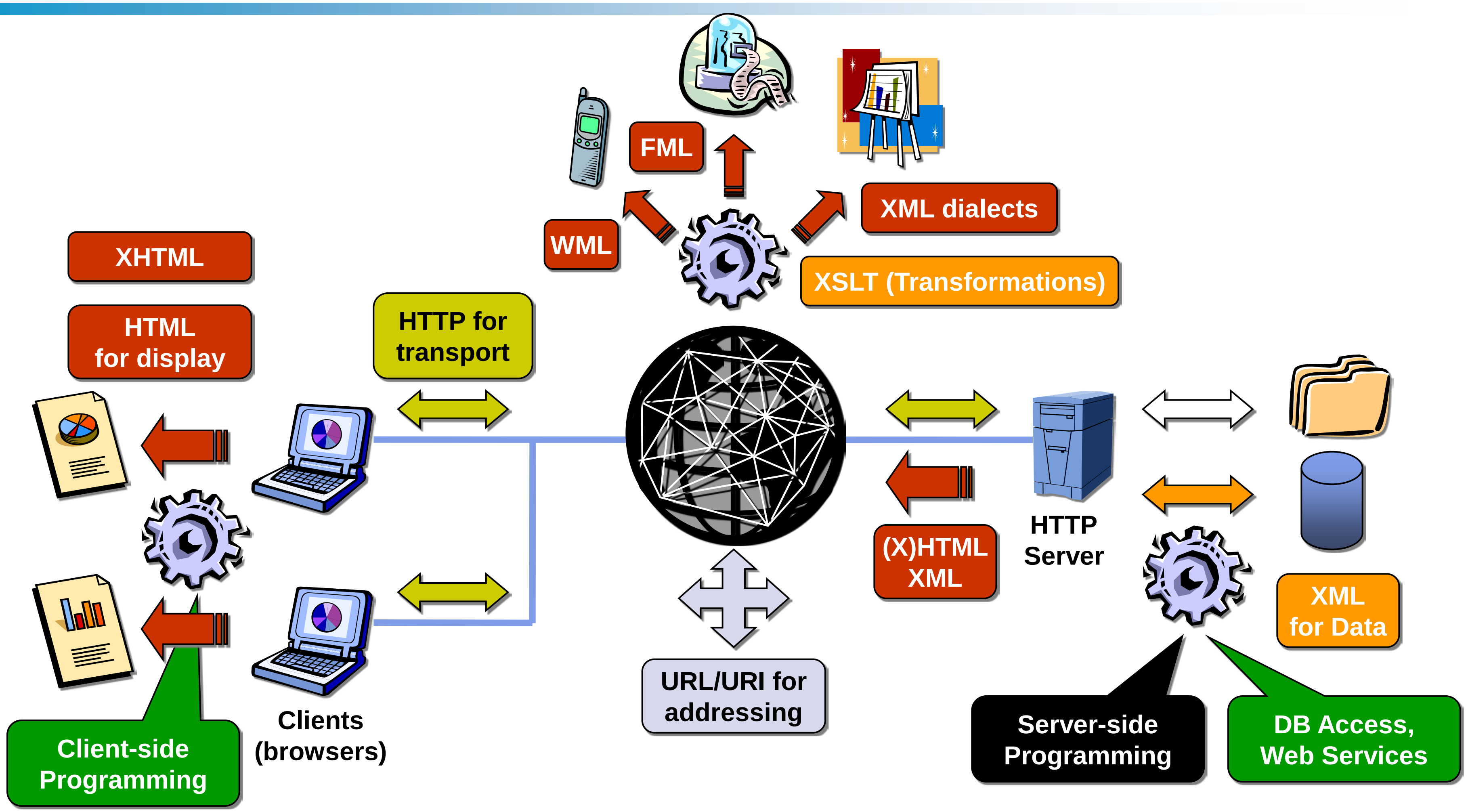
EURECOM

Sophia Antipolis

Lab session
Google Application Engine - GAE

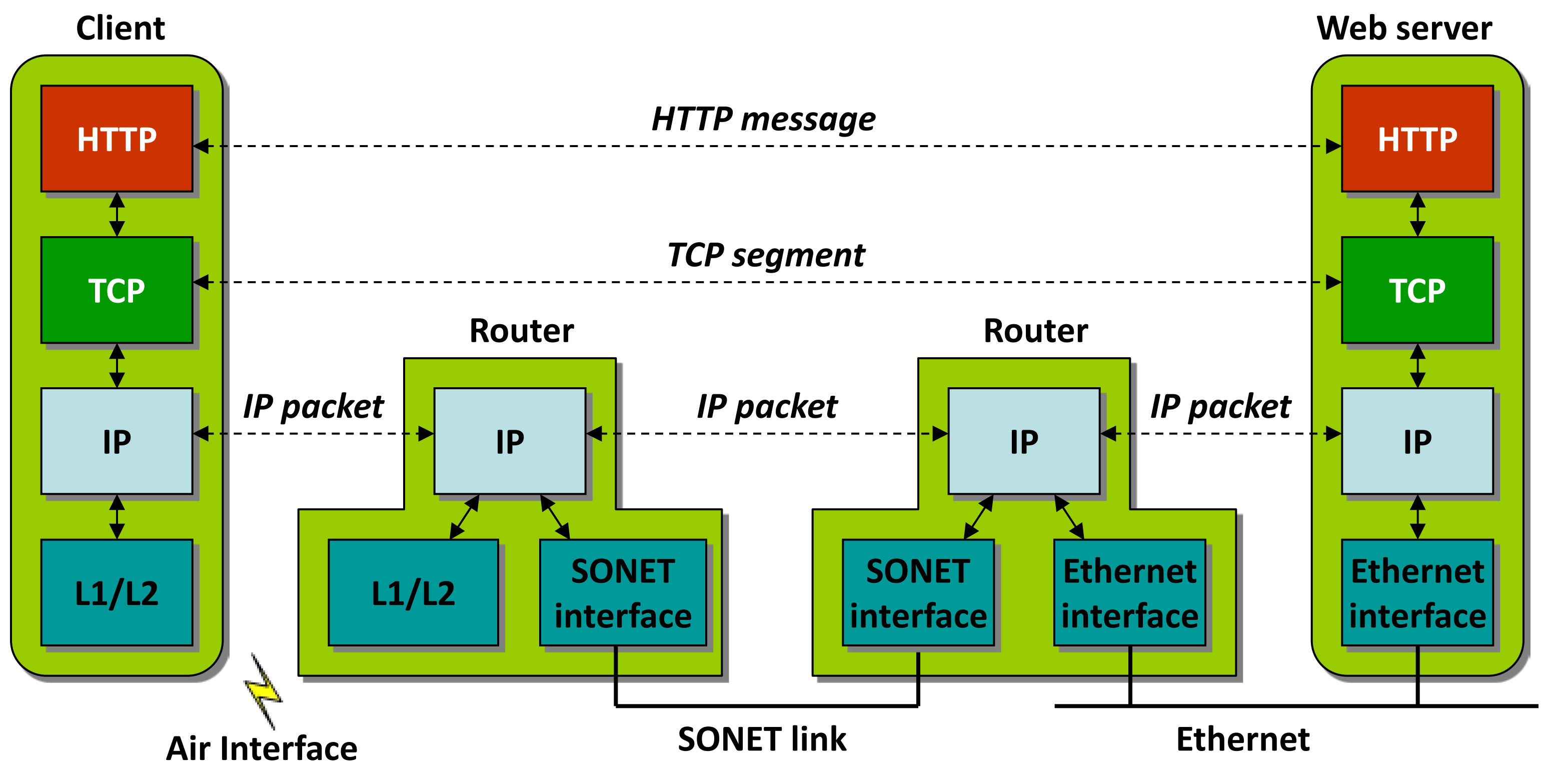
Navid Nikaein

Overall Interactions



Source: P. Michiardi, S. Crosta

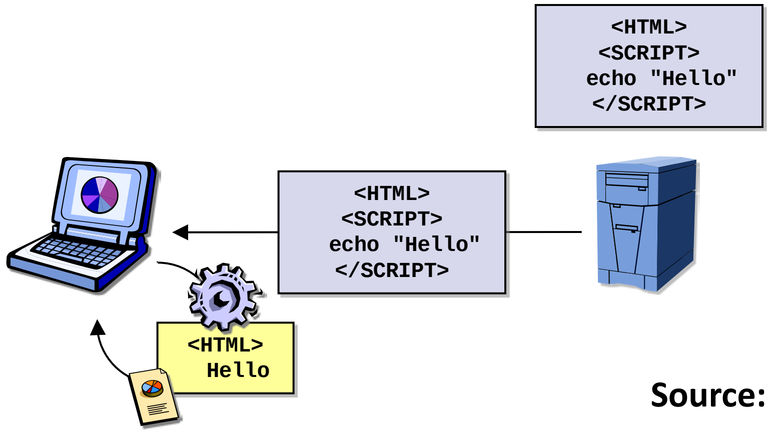
Protocol Interaction



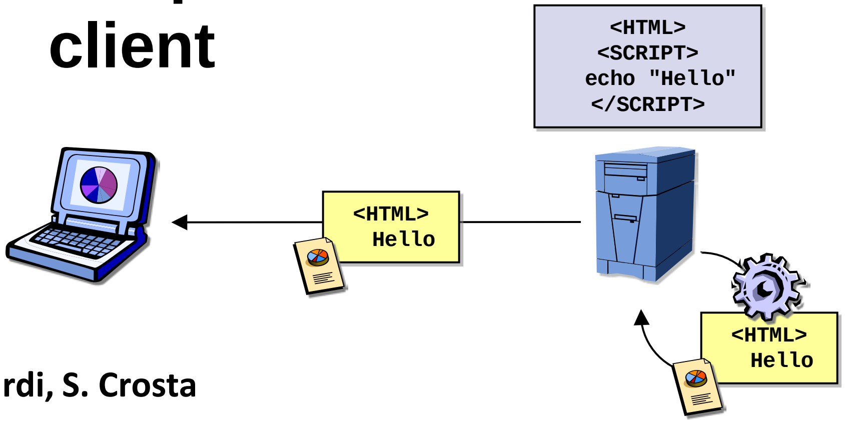
Source: P. Michiardi

Client- vs Server-side Programming

Client-side	Server-side
▪ HTML and script sent to client ▪ Script processed before display ▪ Script visible to client	▪ Script processed before sending HTML to client ▪ Client receives and displays processed HTML ▪ Script hidden from client

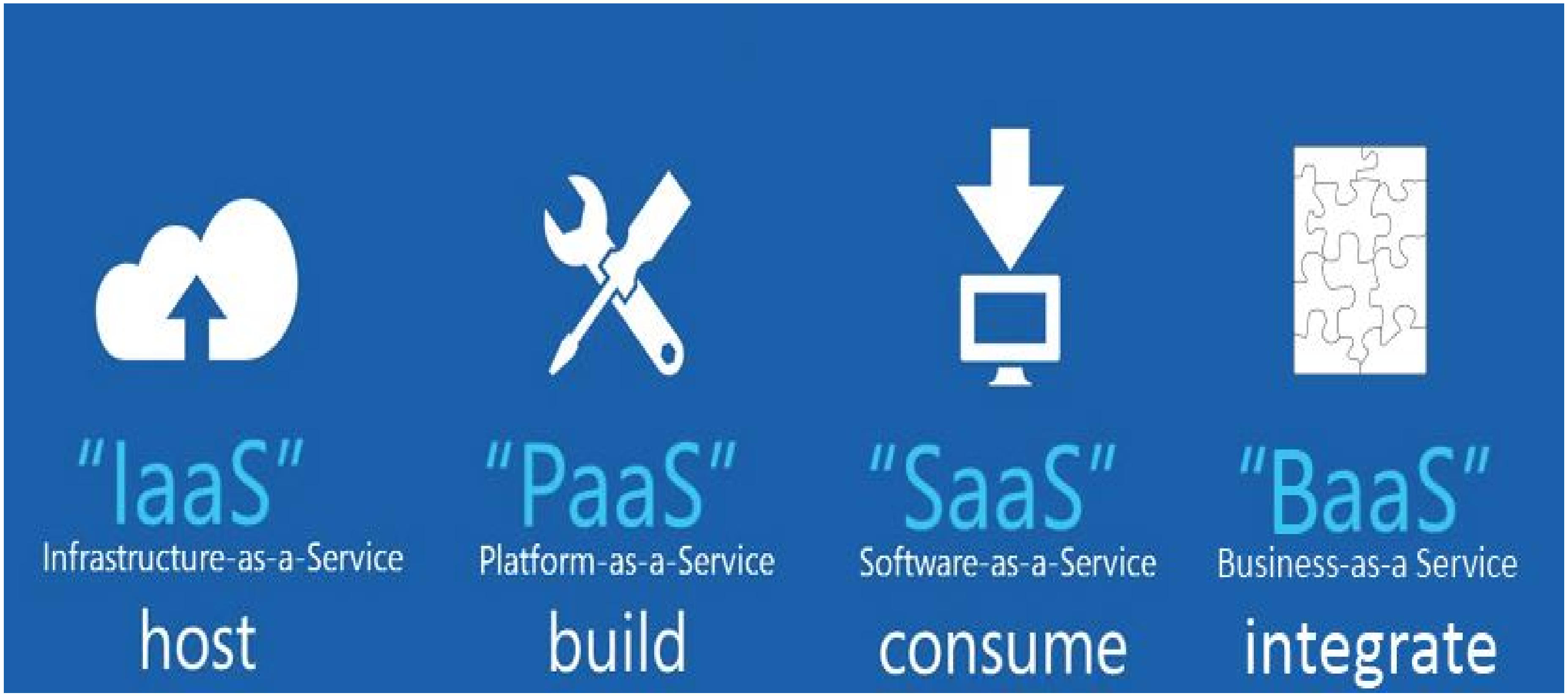


<HTML> <SCRIPT> echo "Hello" </SCRIPT>
<HTML> <SCRIPT> echo "Hello" </SCRIPT>
<HTML> Hello



Source: P. Michiardi, S. Crosta

IaaS vs Paas vs SaaS



Platform as a service

- **Enabling communication and synchronization among devices using cloud**
  - Centralized resources
  - Separate clients
  - Scalability
  - …

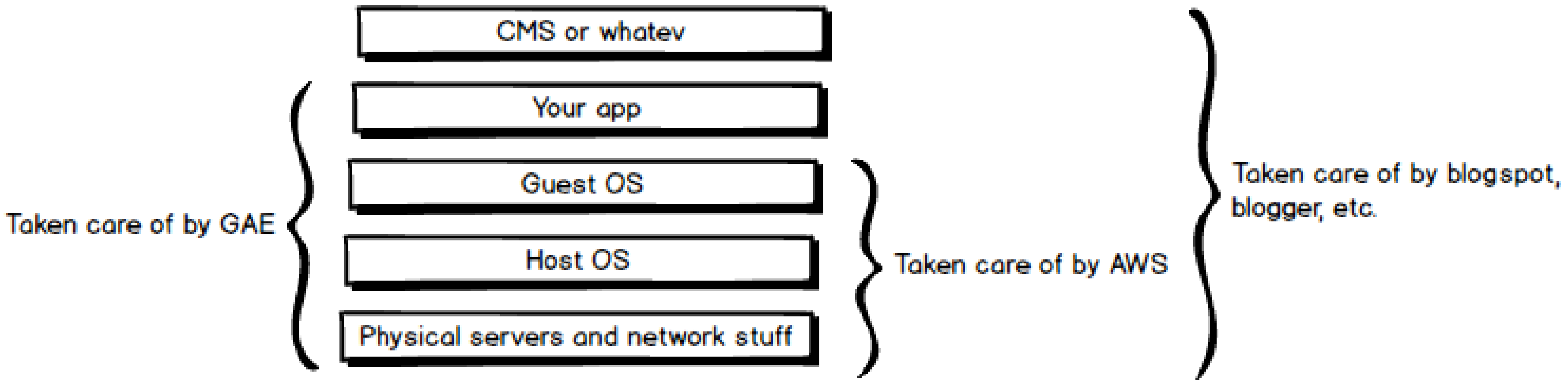
- **Different solutions exist**

vs. amazon web services™

Python | Java | PHP | Go

GCP/GAE and AWS

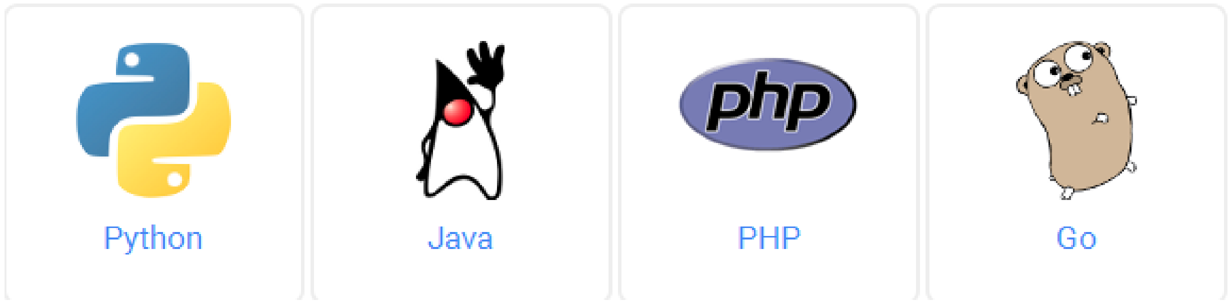
- **IaaS vs Paas**



Read http://notwastingtime.blogspot.fr/2010/05/google-app-engine-gae-versus-amazon-web.html

GAE: a PaaS

- Google App Engine lets you build and run applications on Google's infrastructure.
- App Engine applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change.

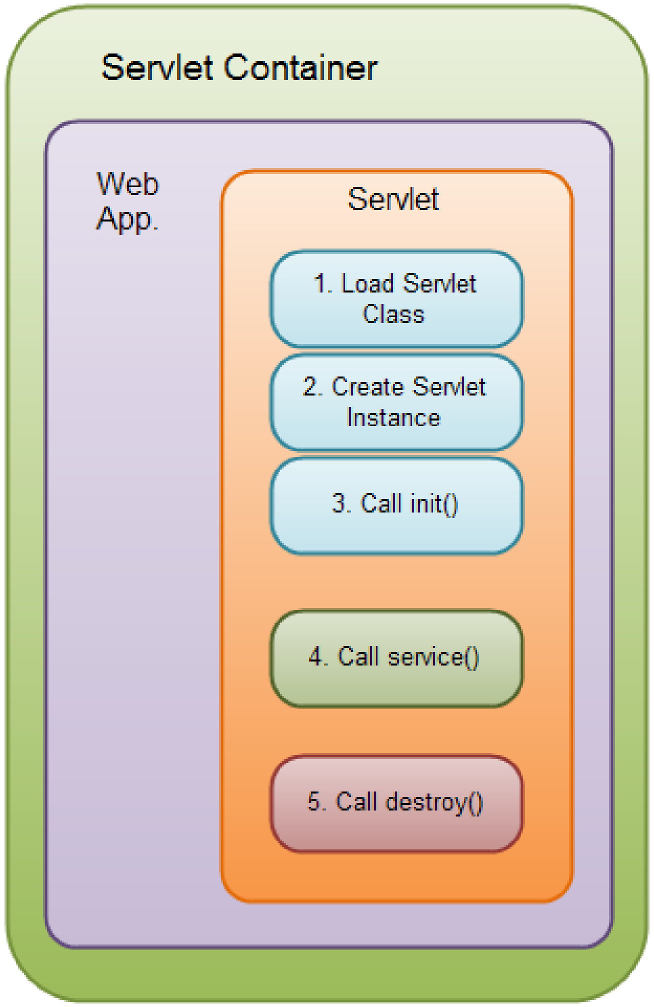


- With App Engine, there are no servers for you to maintain. You simply upload your application and it's ready to go

SERVLET

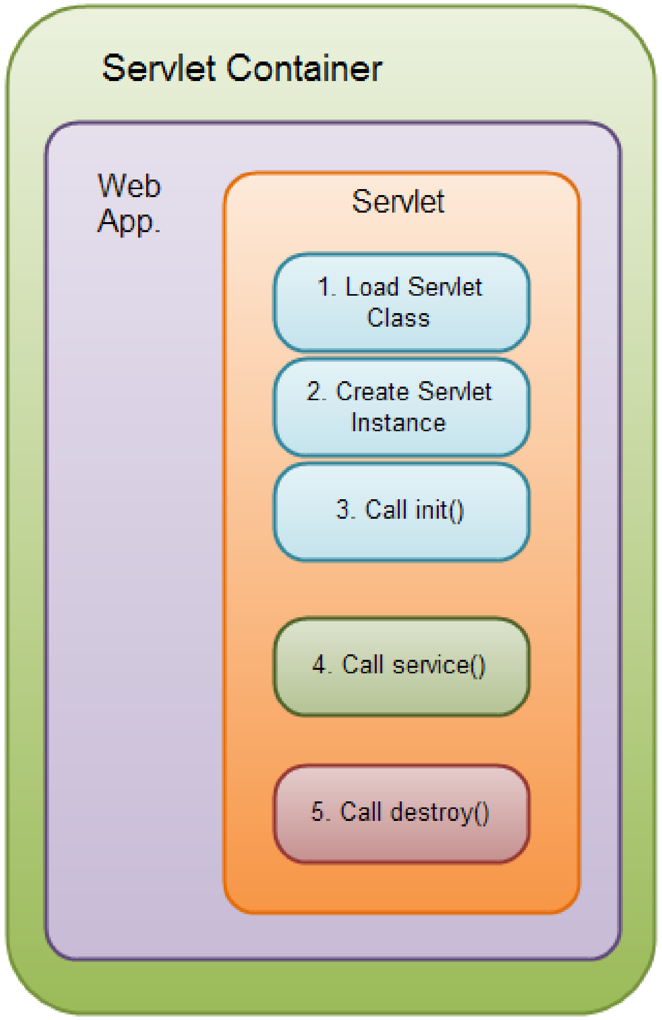
Servlet job and life cycle

- **Extend the server capabilities by means of request-response programming model**
- **Usage**
  - Read and store data and process request from the client
  - Send data / response back to the client
  - Build a dynamic content
  - Manage state information
- **See http://en.wikipedia.org/wiki/Java_Servlet & http://en.wikibooks.org/wiki/J2EE_Programming/Servlet**



Servlet

- **Servlet container**
  - Manage the life cylce
  - Dispatch/mapping the URL
- **Servlet API**
  - **Packege : javax.servlet.***
- **Different data format may exist**
  - Html, xml, json



Servlet

- **Three methods to manages the servlet lifecycle**
  - Init(),service(), destroy()

- **Service method of HttpServlet class dispatches requests to the methods**
  - **doGet(), doPost(),** doPut(), doDelete(), etc according to the HTTP request

Project structure and files

Project structure	Web.xml

- **src/**
- **war/**
  - WEB-INF/
    - appengine-generated/
    - classes/
    - lib/
    - appengine-web.xml
    - web.xml

```
<servlet>
<servlet-name>Helloworldgae</servlet-name>
<!-- class name -->
<servlet-class>eurecom.fr.helloworldgae.HelloworldgaeServlet</servlet-class>
<!-- class tree -->
</servlet>
<servlet-mapping>
<servlet-name>Helloworldgae</servlet-name>
<!-- class name -->
<url-pattern>/helloworldgae</url-pattern>
</servlet-mapping>
<!-- Class pattern in the URL-->
```

Servlet

- **Java class that runs on the server side that receives HTTP data and does a processing or more following the constraints of the HTTP protocol**

```
@WebServlet("/Hello")
public class Hello extends HttpServlet {
// called at the reception of the http POST request
protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {
         response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("Hello World");
        } finally {
            out.close();
        }
    } // called at the reception of the http  GET request
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/plain");
        String msg = "Hello, world";
         if (request.getParameter("name") != null) {
            msg += " from " + request.getParameter("name");
         }
         response.getWriter().println(msg);
}
```

HTTP Response

- Response type (see http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html)
  - 1XX: informative
  - 2XX: success
  - 3XX: redirection
  - 4XX: client error
  - 5XX: server error

- Generate error

```
protected void doXXX(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
   response.sendError(HttpServletResponse.SC_SERVICE_UNAVAILABLE, "the server is
            overloaded. Please try later.");
  } finally {
   out.close();
  }
}
```

- See http://en.wikibooks.org/wiki/J2EE_Programming/Servlet

Java server pages - JSP

- **The JavaServer Pages is a technology for inserting dynamic content into a HTML or XML page using a Java servlet container**
  - **Declaration:** `<%! int serverInstanceVariable = 1; %>`
  - **Scriptlet :** `<% int localStackBasedVariable = 1; out.println(localStackBasedVariable); %>`
  - **Expression**: `<%= "expanded inline data " + 1 %>`
  - **Comment:** `<%-- This is my first JSP. --%>`
  - **Directive :** `<%@ page import="java.util.*" %>`

Lab Session Steps

1. **Helloworldgae [10 minutes]**
2. **Hellomoongae [40 minutes]**
    1. Create a servlet
    2. JSP
    3. Publish and deploy the code
3. **AddressBook [1h30]**
    1. Database
    2. POST and GET
    3. List of contact and contact details
    4. Modify contact
    5. Output format in json
4. **Website [10 minutes]**

Technology

- **Google App Engine GAE**
  - PaaS cloud computing platform
  - Developing and hosting web applications in google data center

- **AWS**
  - IaaS cloud computing infrastructure
  - Developing and hosting web applications in Amazon web service

- **Appache Tomcat (formerly *Jakarta Tomcat*)**
  - open source web server and servlet container
  - Implements java servlet and javaServer pages