$01KPS_{BF}$
Progettazione di applicazioni web

Servlets in the J2EE platform

Fulvio Corno

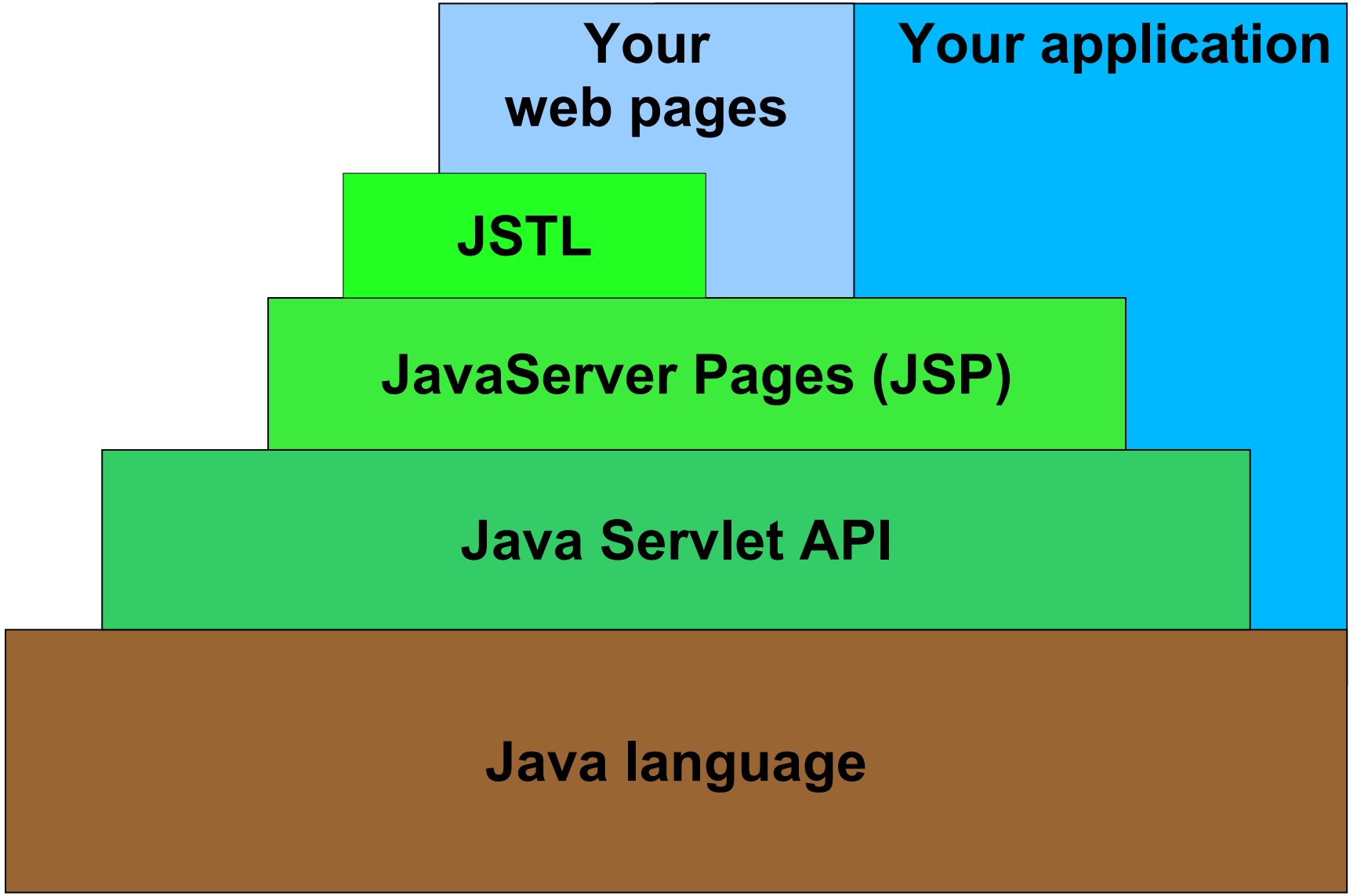
Dipartimento di Automatica e Informatica

Politecnico di Torino

The J2EE Presentation tier

- Servlets
  - Java classes that handle requests by producing responses (e.g., HTTP requests and responses)
- JavaServer Pages (JSP)
  - HTML-like pages with some dynamic content.
  - Translated into servlets automatically
- JSP Standard Tag Library (JSTL)
  - Set of standard components for JSP
  - Used inside JSP pages.

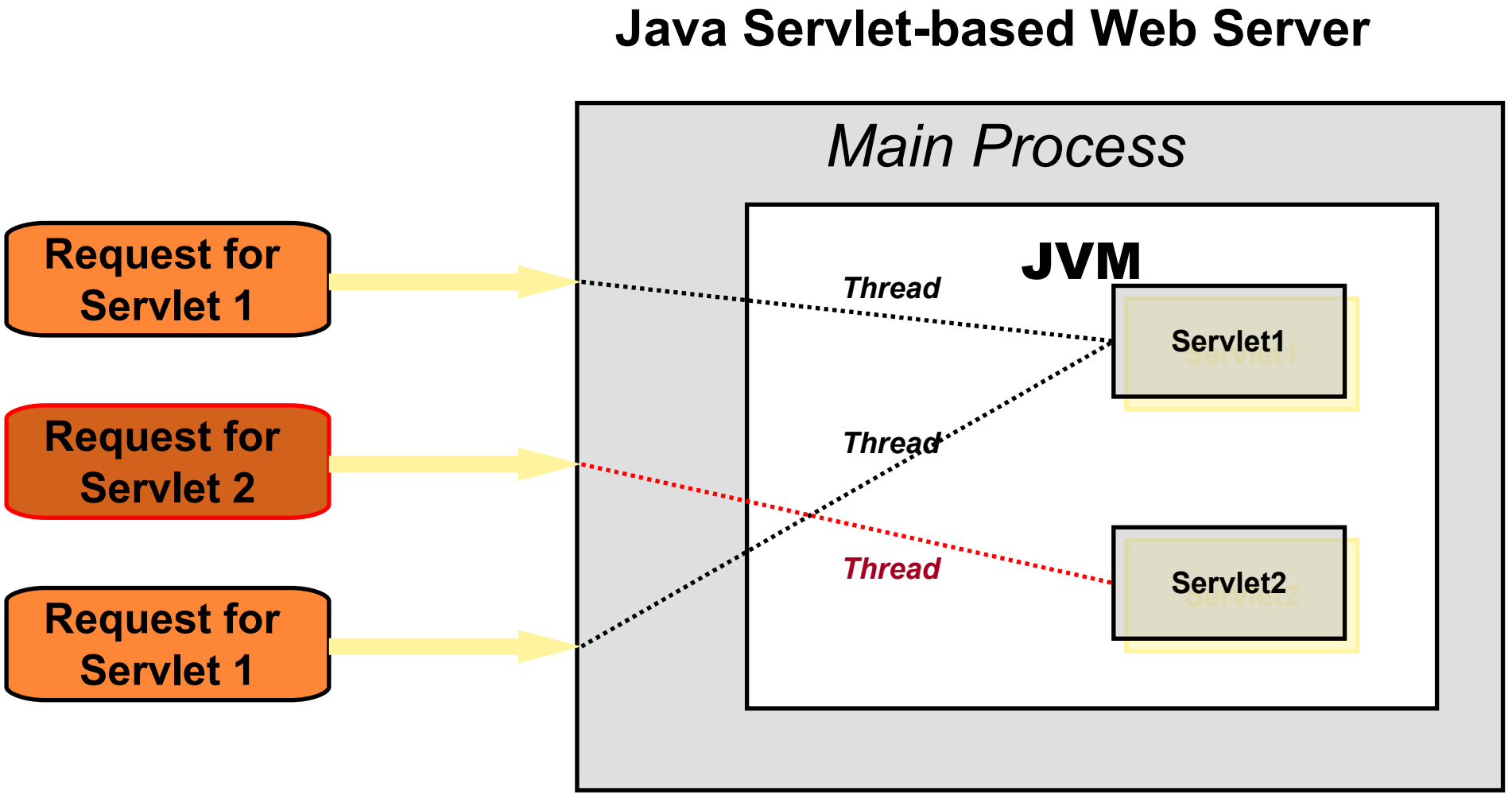
Organization of the platform



A closer look at a servles

- A Java class that extends HttpServlet
- Compiled and placed in the appropriate directory
- When a request for a servlet arrives, the servlet container (a JVM):
  - checks if an instance of that servlet exists
  - if not, it creates/loads an instance
  - calls the servlet's service() method (which in turn may call doGet()/doPost()/doXXX())

Servlet life cycle

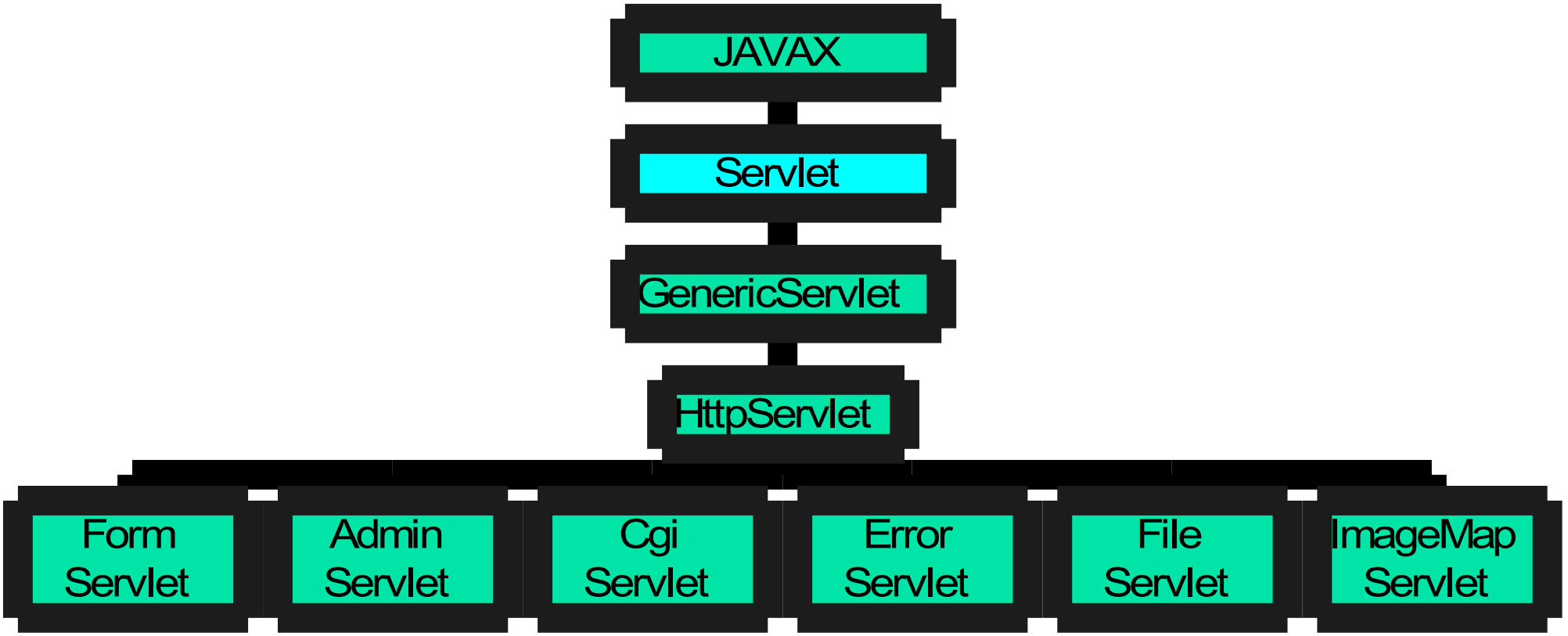


SERVLET BASICS

- Java servlets are the first standard extension to Java, including two packages:
  - javax.servlet
  - javax.servlet.http

http://java.sun.com/products/servlet/2.2/javadoc/

SERVLET PACKAGE FRAMEWORK



SERVLET INTERFACE

```
Servlet interface
{

  void init(ServletConfig sc) throws ServletException;

  void service(ServletRequest req, ServletResponse res);
                    throws ServletException, IOException;

  void destroy();

}
```

Structure of a servlet

- A servlet does not have a main() method.
- Certain methods of a servlet are invoked by the server in the process of handling requests.
- Each time the server dispatches a request to a servlet, it invokes the servlet's service() method.

Request / response

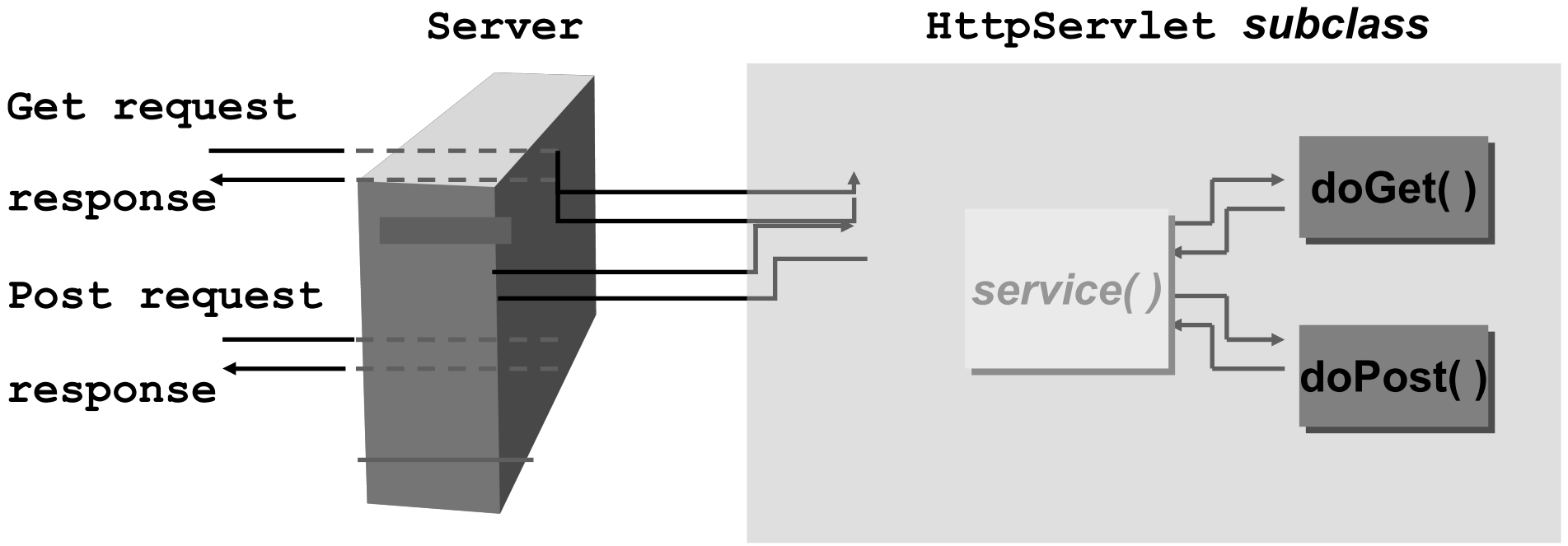
- The service() method accepts two parameters:
  - a **request** object: tells the servlet about the request
  - a **response** object: the response object is used to return a response

Check:
http://java.sun.com/products/servlet/2.2/javadoc/

HttpServlet class

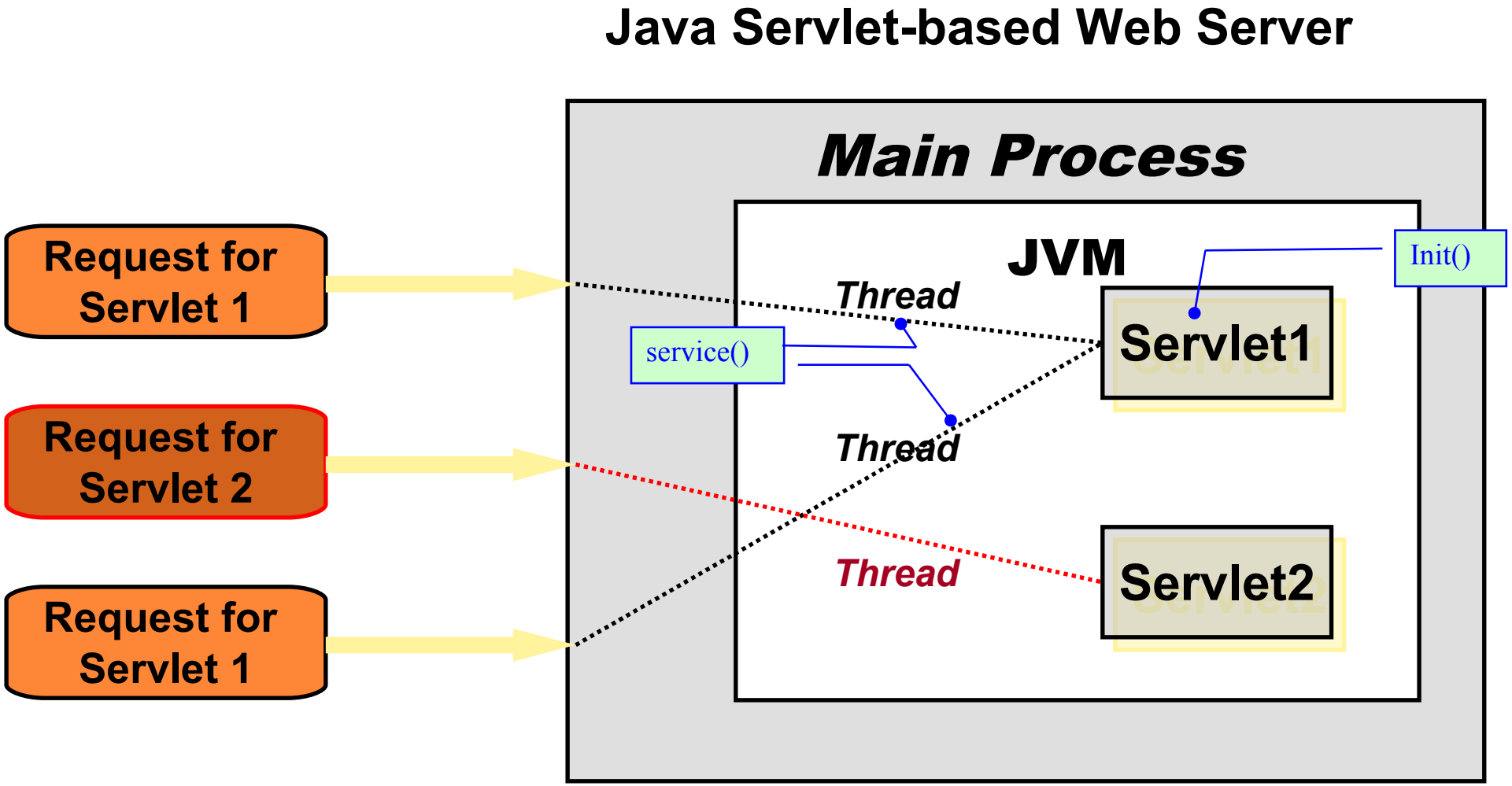
- **Servlet** is a simple Java class which must implement javax.servlet.Servlet interface.
- **GenericServlet** class provides a default implementation of this interface so that we don't have to implement every method of it.
- **HttpServlet** class extends GenericServlet to provide an HTTP protocol specific implementation of Servlet interface.

AN HTTP SERVLET HANDLING GET AND POST REQUEST



Any subclass overrides the doGet()/doPost method(s), not the service() method

SERVLET LIFE CYCLE



HttpServlet methods (1/2)

- init()
  - Called only once during the initialization of the Servlet.
- destroy()
  - Called only once when Servlet instance is about to be destroyed.
- service()
  - Do not override this method!!.
- doGet(), doPost(), doPut(), doDelete(), doOptions(), doTrace()
  - These methods are called according to the type of HTTP request received. Override them to generate your own response.
- log()
  - Writes messages to the Servlet's log files.

HttpServlet methods (2/2)

- getLastModified()
  - Override this method to return your Servlet's last modified date.
- getServletInfo()
  - Override this method to provide a String of general info about your Servlet such author, version, copyright etc.
- getServletName()
  - Override this method to return name of the Servlet.
- getInitParameter(), getInitParameterNames()
  - Return value(s) of initialization parameter(s)
- getServletConfig()
  - Returns a reference to ServletConfig object.
- getServletContext()
  - Returns reference to ServletContext object.

Do not override...

- service()
- getServletConfig()
- getServletContext()

The three methods which stated above should not be overridden as their default implementation is more than enough to suffice.

A “HELLO WORLD” SERVLET

```
public class HelloServlet extends HttpServlet {

public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
      throws ServletException, IOException {

   response.setContentType("text/html");
   PrintWriter out = response.getWriter();
   out.println("CIAO MONDO!");

}

}
```

A “HELLO WORLD” HTML SERVLET

```
protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
response.setContentType("text/html");
PrintWriter out=response.getWriter();
String docType="<!DOCTYPE HTML PUBLIC \"-//W3C/ /DTD HTML 4.0 "+
   "Transitional //IT\" > \n";
out.println(docType);
out.println(
"<HTML>\n" +
"<HEAD><TITLE> CIAO MONDO HTML </TITLE><HEAD> \n" +
"<BODY>\n" +
"<H1> CIAO MONDO </H1> \n" +
"</BODY>" +
"</HTML>");
}
```

Servlet input processing

- `protected void doGet(`
  - **`HttpServletRequest request,`**
  - `HttpServletResponse response`
  - `) throws ServletException, IOException`

- A servlet may access the HTTP request information trough the request object

- See: http://java.sun.com/javaee/5/docs/api/
  - HttpServletRequest
  - ServletRequest

Request parameters

- **String getParameter(String name)**
  - Returns the value of a request parameter as a String, or null if the parameter does not exist.
- **Map getParameterMap()**
  - Returns a java.util.Map of the parameters of this request.
- Enumeration getParameterNames()
  - Returns an Enumeration of String objects containing the names of the parameters contained in this request.
- String[] getParameterValues(String name)
  - Returns an array of String objects containing all of the values the given request parameter has, or null if the parameter does not exist.

HTTP headers information (1/2)

- String getHeader(String name)
  - Returns the value of the specified request header as a String.
- Int getIntHeader(String name)
  - Returns the value of the specified request header as an int.
- Enumeration getHeaderNames()
  - Returns an enumeration of all the header names this request contains.
- Enumeration getHeaders(String name)
  - Returns all the values of the specified request header as an Enumeration of String objects.

HTTP headers information (2/2)

- long getDateHeader(String name)
  - Returns the value of the specified request header as a long value that represents a Date object.
- String getMethod()
  - Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.
- String getQueryString()
  - Returns the query string that is contained in the request URL after the path.
- String getRemoteUser()
  - Returns the login of the user making this request, if the user has been authenticated, or null if the user has not been authenticated.

HTTP session handling

- Cookie[] getCookies()
  - Returns an array containing all of the Cookie objects the client sent with this request.
- String getRequestedSessionId()
  - Returns the session ID specified by the client.
- **HttpSession getSession()**
  - Returns the current session associated with this request, or if the request does not have a session, creates one.
- boolean isRequestedSessionIdValid()
  - Checks whether the requested session ID is still valid.

Other interesting information

- String getRemoteAddr()
  - Returns the Internet Protocol (IP) address of the client or last proxy that sent the request.
- String getRemoteHost()
  - Returns the fully qualified name of the client or the last proxy that sent the request.
- int getRemotePort()
  - Returns the Internet Protocol (IP) source port of the client or last proxy that sent the request.
- String getServerName()
  - Returns the host name of the server to which the request was sent.
- int getServerPort()
  - Returns the port number to which the request was sent.

HttpSession class (1/3)

- getAttribute(), getAttributeNames(), setAttribute(), removeAttribute()
  - These methods are used to set, get and remove objects from a user session.
- getId()
  - Every session created by the server has a unique 'id' associated with it in order to identify this session from other sessions. This method returns the 'id' of this session.
- getCreationTime()
  - Simple returns a long value indicating the date and time this session was created, meaning there by that you get the time this user first accessed your site.

HttpSession class (2/3)

- getLastAccessedTime()
  - Returns a long value indicating the last time user accessed any resource on this server.
- getMaxInactiveInterval(), setMaxInactiveInterval()
  - Return and set the maximum inactive interval in seconds for this session respectively. Every session has a maximum inactive interval during which if user doesn't make request to the server, the session is invalidated.
- isNew()
  - Returns a boolean value indicating if the session is new. Either it is the first page of the site user has hit so his session is new and has just been created or that user is not accepting cookies required for managing sessions so this value will then always return true.

HttpSession class (3/3)

- invalidate()
  - Simply invalidates a session. You can use this method on a 'logout' page allowing user to end his session. If after invalidation of his session user accesses some resource on the site then a new session will be created for it.

ServletConfig class

- ServletConfig object is used to pass information to the Servlet during its initialization
  - Inistialization information is stored in web.xml
- Servlet can obtain information regarding initialization parameters and their values
- using different methods of ServletConfig class.

ServletConfig methods

- getInitParameter(String paramName)
  - Returns value of the given parameter. If value of parameter could not be found in web.xml file then a null value is returned.
- getInitParameterNames()
  - Returns an Enumeration object containing all the names of initialization parameters provided for this Servlet.
- getServletContext()
  - Returns reference to the ServletContext object for this Servlet. It is similar to getServletContext() method provided by HttpServlet class.
- getServletName()
  - Returns name of the Servlet as provided in the web.xml file or if none is provided then returns complete class path to the Servlet.

A “hello world” servlet with init

```
public void init(ServletConfig config) throws ServletException {
        super.init(config);
        String sRipetizioni =
                config.getInitParameter("ripetizioni");
        Ripetizioni = Integer.parseInt(sRipetizioni);
}
```

How To... JSP vs Servlet

- Useful JSP constructs
  - <%@page pageEncoding=”text/html” %>
  - <jsp:forward page=”a.jsp” %>
  - <jsp:useBean … scope=”request” />
  - <jsp:useBean … scope=”session” />
- How to translate them into Servlet instructions?

HOW TO... output HTML

- response.setContentType("text/html");
- PrintWriter out = response.getWriter();
- out.print("<html><head>...");

HOW TO... forward to another page

- request.getRequestDispatcher(*url*).**forward**(request, response);

- Step-by-step:
  - // request is HttpServletRequest object
  - **RequestDispatcher** rd;
  - rd = request.**getRequestDispatcher**("**pathToServlet**");
  - rd.forward(request, response);

HOW TO... define and use a request Bean

- Create a new bean with request scope
  - UserDataBean userData = new UserDataBean() ;
  - **request.setAttribute("userData", userData);**
- Retrieve a bean from the request scope
  - UserDataBean userData = ((UserDataBean)request.**getAttribute**("userData"));

HOW TO... define and use a session Bean

- Linking to the current session
  - HttpSession session = **request.getSession**(true);
- Creating a new bean with session scope
  - CounterBean counter;
    if(session.isNew()) {
      counter = new CounterBean();
      session.setAttribute("counter", counter);
    }
- Retrieving an existing bean with session scope
  - Counter =
    ((CounterBean)session.**getAttribute**("counter"));

For more info...

- http://www.unix.org.ua/orelly/java-ent/servlet/ch01_01.htm
- http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/servlet/