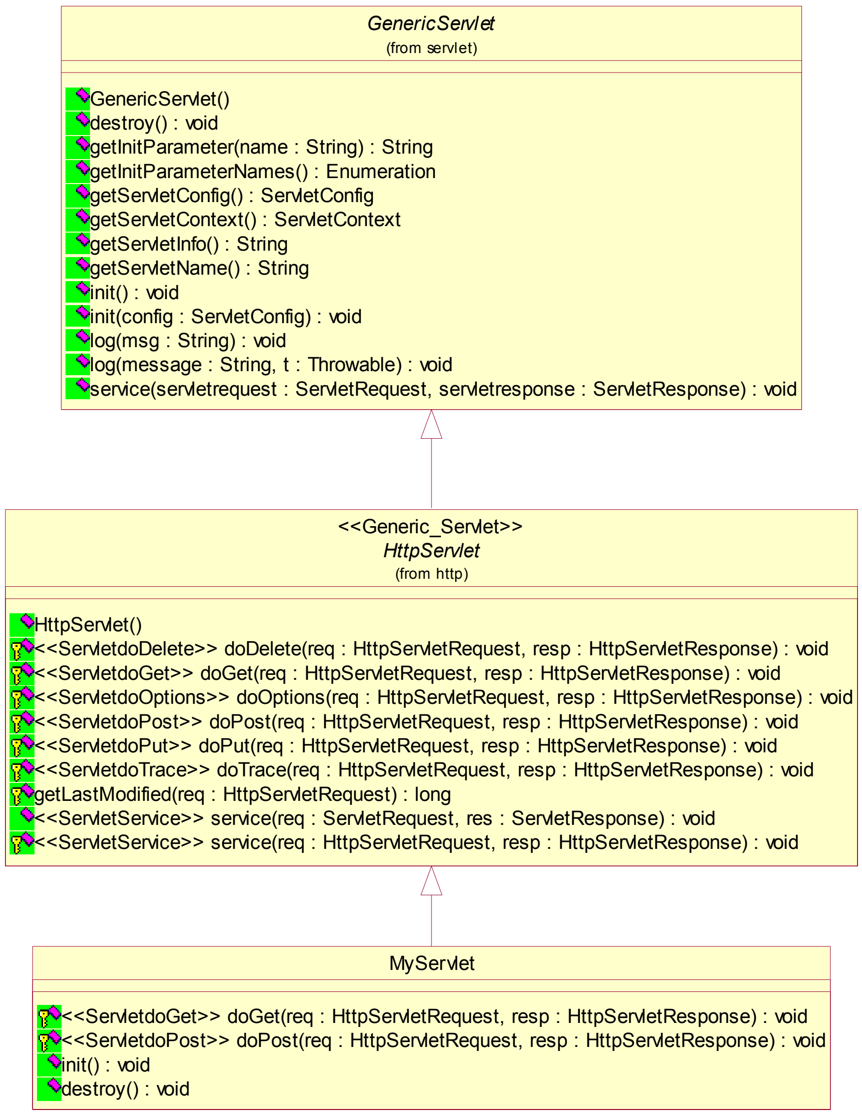
EJB 605 Lecture 3 - Servlet Programming

A servlet is a Java technology based web component, managed by a container, that generates dynamic content.

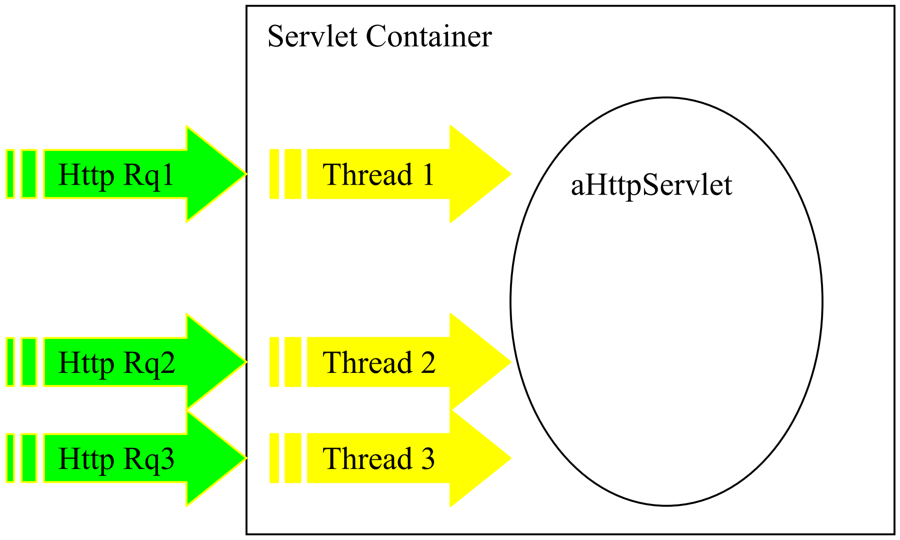
The servlet container is a part of a web server or application server that provides the network services over which requests and responses are sent, decodes MIME based requests, and formats MIME based responses. A servlet container also contains and manages servlets through their lifecycle.



GenericServlet
(from servlet)
GenericServlet()
destroy() : void
getInitParameter(name : String) : String
getInitParameterNames() : Enumeration
getServletConfig() : ServletConfig
getServletContext() : ServletContext
getServletInfo() : String
getServletName() : String
init() : void
init(config : ServletConfig) : void
log(msg : String) : void
log(message : String, t : Throwable) : void
service(servletrequest : ServletRequest, servletresponse : ServletResponse) : void
<<Generic_Servlet>>
HttpServlet
(from http)
HttpServlet()
<<ServletdoDelete>> doDelete(req : HttpServletRequest, resp : HttpServletResponse) : void
<<ServletdoGet>> doGet(req : HttpServletRequest, resp : HttpServletResponse) : void
<<ServletdoOptions>> doOptions(req : HttpServletRequest, resp : HttpServletResponse) : void
<<ServletdoPost>> doPost(req : HttpServletRequest, resp : HttpServletResponse) : void
<<ServletdoPut>> doPut(req : HttpServletRequest, resp : HttpServletResponse) : void
<<ServletdoTrace>> doTrace(req : HttpServletRequest, resp : HttpServletResponse) : void
getLastModified(req : HttpServletRequest) : long
<<ServletService>> service(req : ServletRequest, res : ServletResponse) : void
<<ServletService>> service(req : HttpServletRequest, resp : HttpServletResponse) : void
MyServlet
<<ServletdoGet>> doGet(req : HttpServletRequest, resp : HttpServletResponse) : void
<<ServletdoPost>> doPost(req : HttpServletRequest, resp : HttpServletResponse) : void
init() : void
destroy() : void

Single Instance Servlet

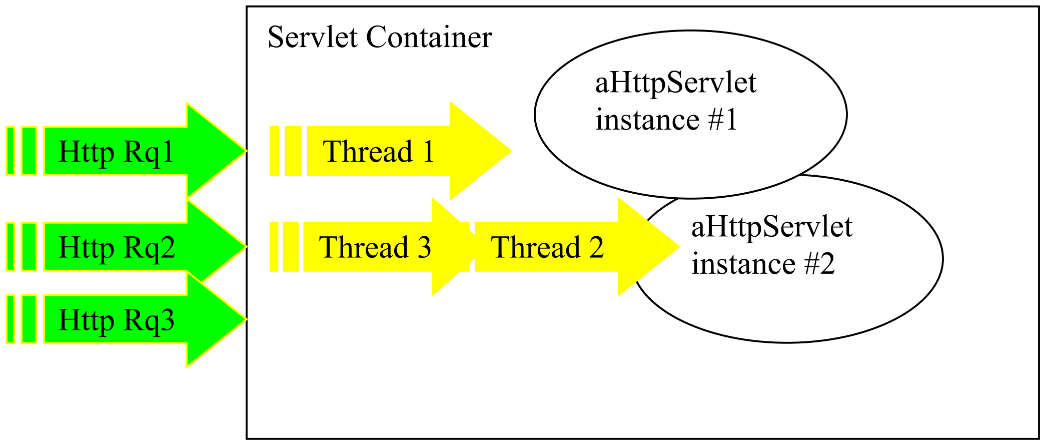
```
public class HelloWorldServlet extends HttpServlet {
…}
```



Servlet container initializes a single instance of a Servlet class. Multiple requests are handled by execution of concurrent threads.

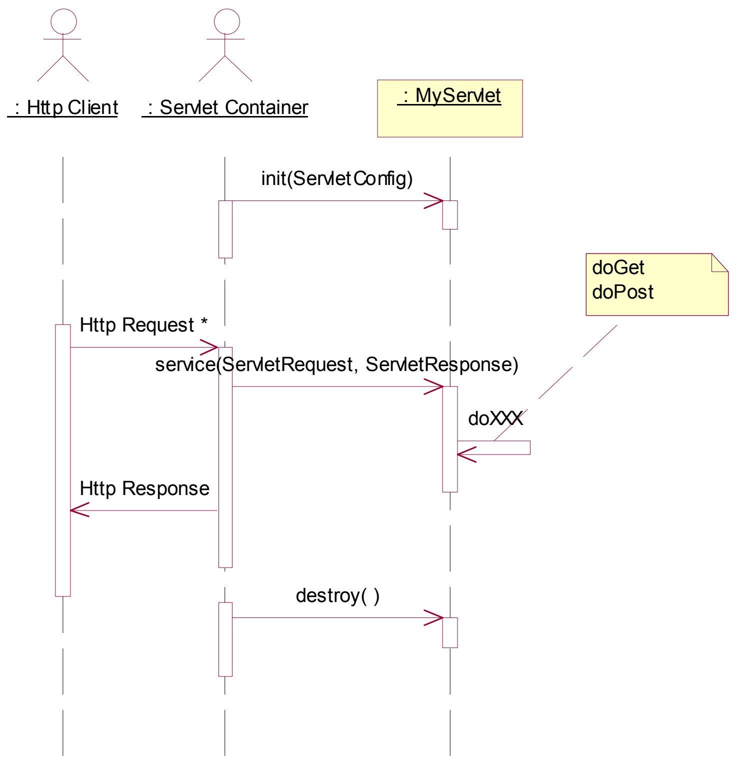
Single Thread Model Servlet

```
public class LoanCalculationServlet
     extends HttpServlet
     implements SingleThreadModel {
…}
```



Servlet container either instantiate multiple instances of a Servlet class (each instance handle one thread) or blocking the second thread until the first thread finish.

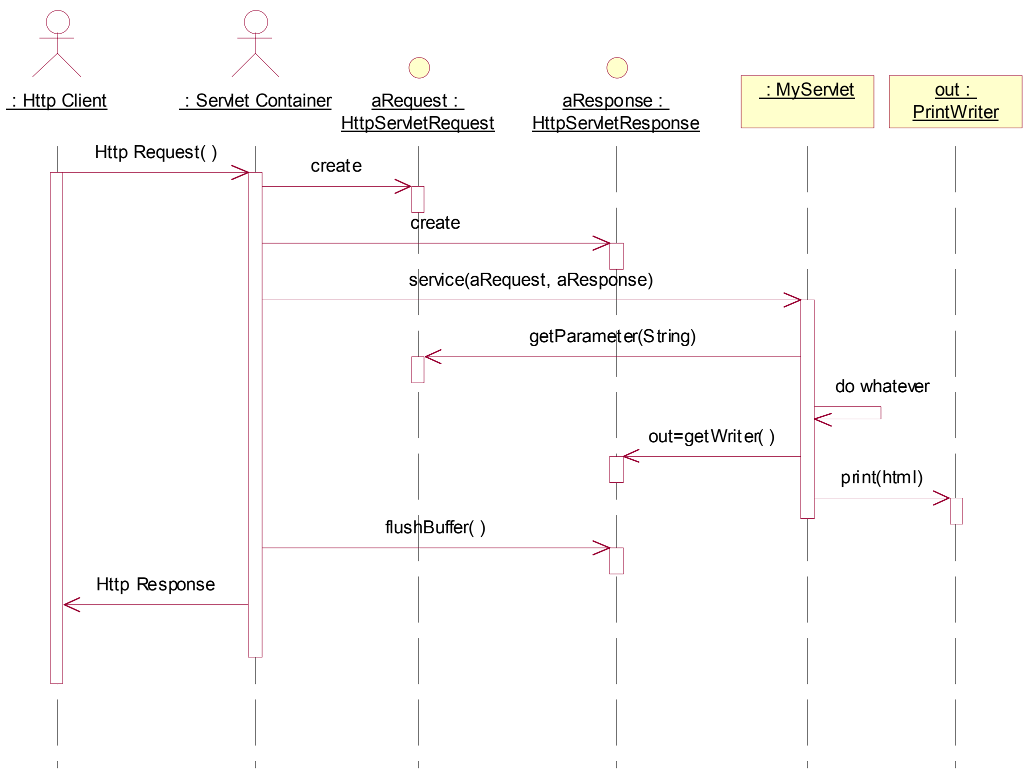
Servlet Life Cycle



Methods handling

	Supporting Functions	HTTP/1.0	HTTP/1.1
service	ALL		
doGet	HTTP GET	Y	Y
doPost	HTTP POST	Y	Y
doPut	HTTP PUT		Y
doDelete	HTTP DELETE		Y
doHead	HTTP HEAD	Y	Y
doOptions	HTTP OPTIONS		Y
doTrace	HTTP TRACE		Y

How Servlet works



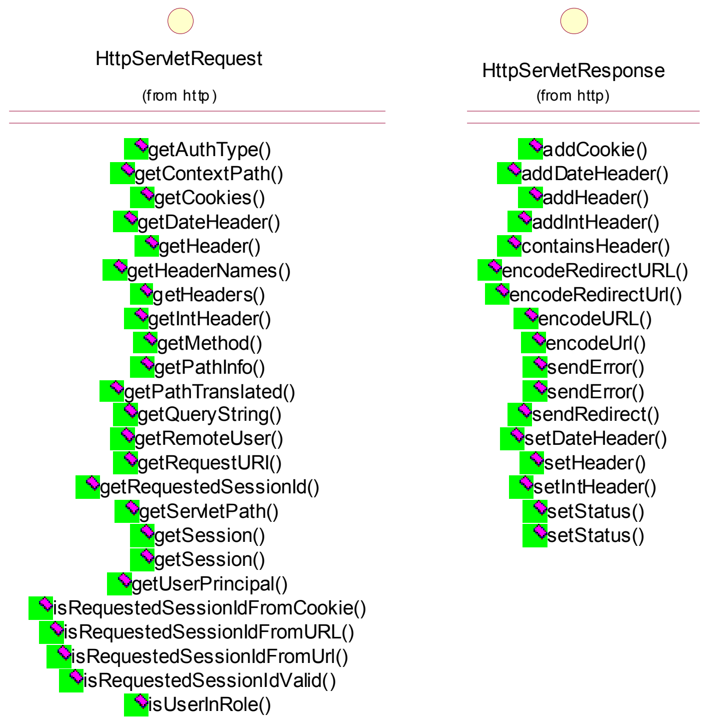
Example of getting Servlet initialization data through init() method.

```
public void init(ServletConfig aConfig) throws ServletException {
        super.init(aConfig);
        System.out.println(this.getClass()+".init("+aConfig+")");
        Enumeration params=aConfig.getInitParameterNames();
        while(params.hasMoreElements()){
                String aName=(String)params.nextElement();
                System.out.println(aName+"="+aConfig.getInitParameter(aName));
        }
}
```

Example of handling service() method.

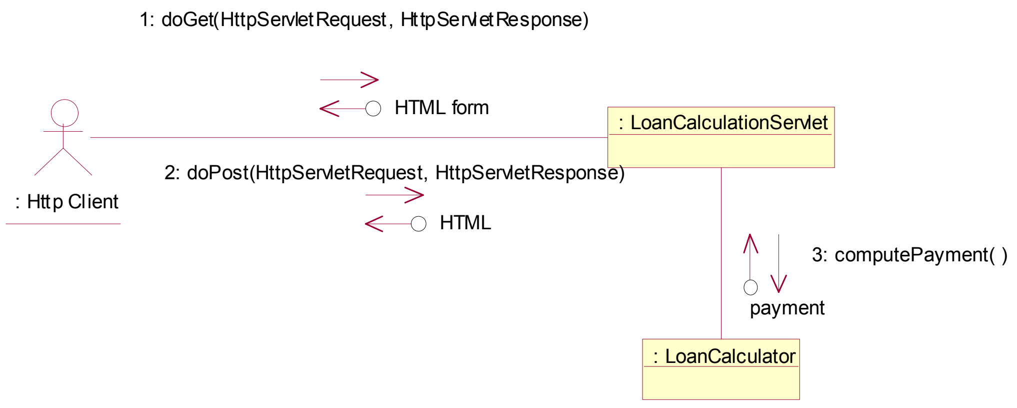
```
protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println(helloHtml());

}
```

Servlet Request and Servlet Response Interface



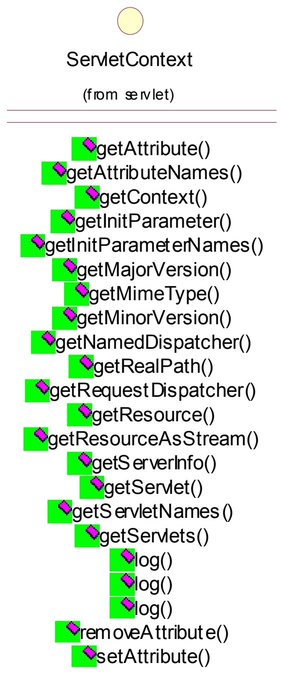
HttpServletRequest and HttpServletResponse are interfaces only

Design Pattern – Use Servlet just for GUI functions. Delegate business function to model class.


1: doGet(HttpServletRequest, HttpServletResponse)
HTML form
: LoanCalculationServlet
2: doPost(HttpServletRequest, HttpServletResponse)
: Http Client
HTML
3: computePayment( )
payment
: LoanCalculator

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {


        double principal=0;
        double interest=0;
        int periods=0;
        int years=0;
        double payment=0;

        try{
                principal=Double.parseDouble((String)request.getParameter("principal"));
                interest=Double.parseDouble((String)request.getParameter("interest"));
                periods=Integer.parseInt((String)request.getParameter("periods"));
                years=Integer.parseInt((String)request.getParameter("years"));

                LoanCalculator calculator=new LoanCalculator();
                calculator.setPrincipal(principal);
                calculator.setAnnualRate(interest);
                calculator.setPeriodsPerYear(periods);
                calculator.setYears(years);

                payment=calculator.computePayment();

        } catch (Exception e){
                throw new ServletException(e);
        }

        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("<html><header><title>Loan Calculation</title></header>");
        out.println("<body>");
        out.println("<p><b>Loan Calculation</b></p>");
        out.println("<table>");
        out.println("<tr><td>Principal</td><td>"+principal+"</td></tr>");
        out.println("<tr><td>Annual Interest</td><td>"+interest+"</td></tr>");
        out.println("<tr><td>Number of periods per year</td><td>"+periods+"</td></tr>");
        out.println("<tr><td>Number of years</td><td>"+years+"</td></tr>");
        out.println("<tr><td>Payment per period</td><td>"+payment+"</td></tr>");
        out.println("</table>");
        out.println("</form>");
        out.println("</body>");
        out.println("</html>");

    }
```

Servlet Context



ServletContext Interface

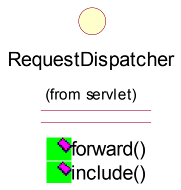
The ServletContext interface defines a servlet's view of the web application within which the servlet is running. The Container Provider is responsible for providing an implementation of the ServletContext interface in the servlet container. Using the ServletContext object, a servlet can log events, obtain URL references to resources, and set and store attributes that other servlets in the context can access. A ServletContext is rooted at a known path within a web server. For example a servlet context could be located at http://www.mycorp.com/catalog. All requests that begin with the /catalog request path, known as the *context path*, are routed to the web application associated with the ServletContext.

There is one instance object of the ServletContext interface associated with each web application deployed into a container. In cases where the container is distributed over many virtual machines, a web application will have an instance of the ServletContext for each VM.

Getting the ServletContext by:

ServletContext aCtx=request.getSession().getServletContext();

Request dispatching and forwarding



RequestDispatcher Interface

When building a web application, it is often useful to forward processing of a request to another servlet, or to include the output of another servlet in the response. The RequestDispatcher interface provides a mechanism to accomplish this.

Getting a Request Dispatcher from ServletContext either by Name or Path.

- getRequestDispatcher(aPath)
- getNamedDispatcher(aName)

```
ServletContext aCtx=request.getSession().getServletContext();
// RequestDispatcher aDispatcher=aCtx.getRequestDispatcher("/hello");
RequestDispatcher aDispatcher=aCtx.getNamedDispatcher("HelloworldServlet");
aDispatcher.include(request,response);
```

Another example:

```
String path = "/raisons.jsp?orderno=5";
RequestDispatcher rd = context.getRequestDispatcher(path);
rd.include(request, response);
```

Using the include() method

The include method of the RequestDispatcher interface may be called at any time. The target servlet of the include method has access to all aspects of the request object, but its use of the response object is more limited:
It can only write information to the ServletOutputStream or Writer of the response object and commit a response by writing content past the end of the response buffer, or by explicitly calling the flushBuffer method of the ServletResponse interface. It cannot set headers or call any method that affects the headers of the response.

Using the forward() method

The forward method of the RequestDispatcher interface may be called by the calling servlet only when no output has been committed to the client. If output data exists in the response buffer that has not been committed, the content must be cleared before the target servlet's service method is called.

Example code in LoanCalculationServlet.

The following code fragment illustrates the LoanCalucationServlet dispatching the request to BannerServlet and FooterServlet.

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

        response.setContentType("text/html");
```

```
			ServletContext aCtx=request.getSession().getServletContext();
			RequestDispatcher aDispatcher=aCtx.getNamedDispatcher("BannerServlet");
			aDispatcher.include(request,response);

			PrintWriter out=response.getWriter();
			out.println("<form name=\"LoanCalculationForm\" method=\"post\" 
action=\"loanCalc\">");
			out.println("<table>");
			out.println("<tr><td>Principal</td><td><input type=\"text\" name=\"principal\" 
value=\""+defaultPrincipal+"\"/></td></tr>");
			out.println("<tr><td>Annual Interest</td><td><select name=\"interest\"><option 
value=\"0.04\">4%</option>");
			out.println("<option value=\"0.05\">5%</option><option 
value=\"0.06\">6%</option></select></td></tr>");
			out.println("<tr><td>Periods per year</td><td><input type=\"text\" name=\"periods\" 
value=\""+defaultPeriods+"\"/></td></tr>");
			out.println("<tr><td>Number of years</td><td><input type=\"text\" name=\"years\" 
value=\""+defaultYears+"\"/></td></tr>");
			out.println("<tr/>");
			out.println("<tr><td/><td><input type=\"submit\" name=\"submit\" 
value=\"submit\"/></td></tr>");
			out.println("</table>");
			out.println("</form>");

			aDispatcher=aCtx.getNamedDispatcher("FooterServlet");
			aDispatcher.include(request,response);
		}
```