Servlets

Based on Notes by Dave Hollinger & Ethan Cerami

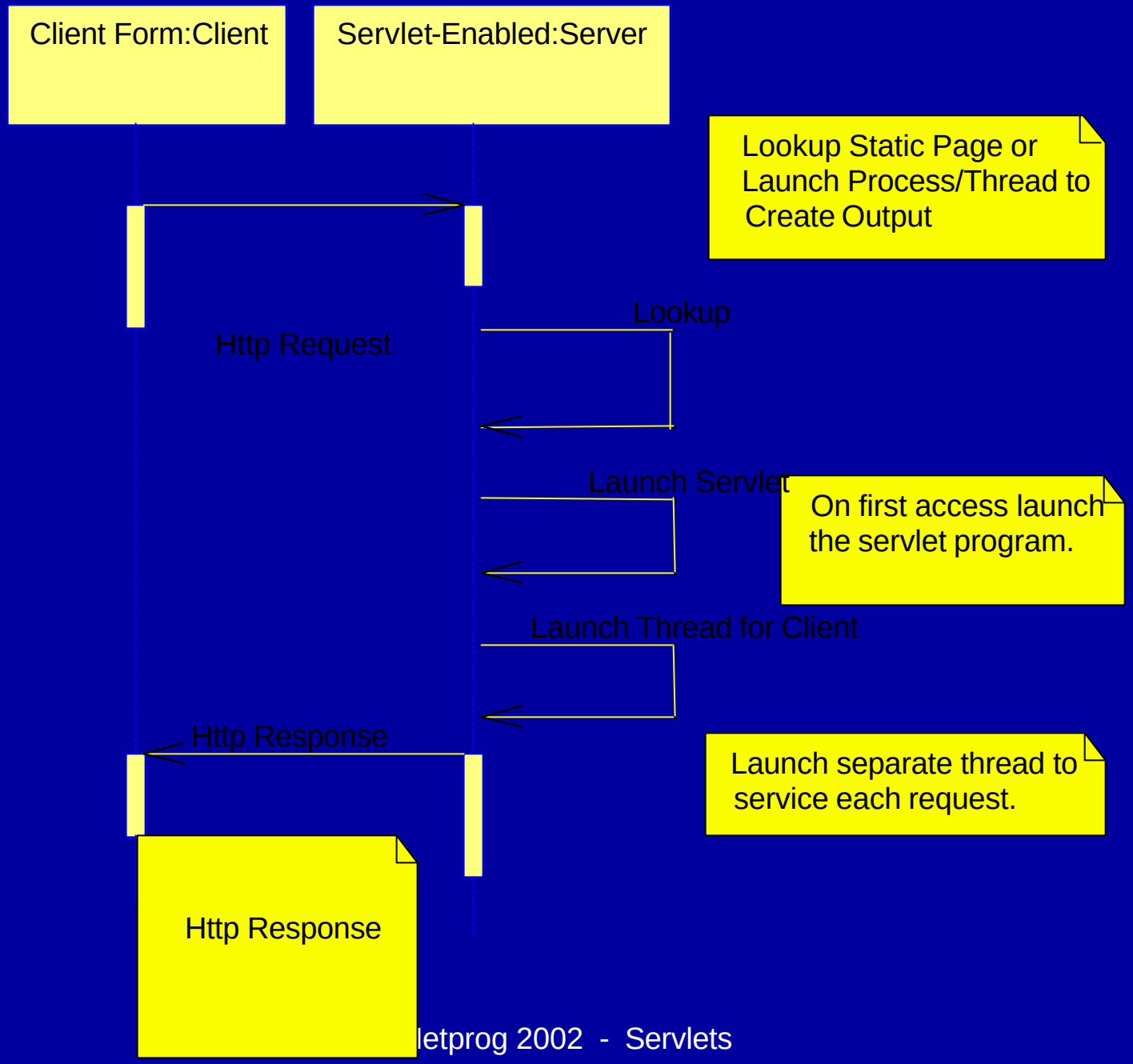
Also, the Online Java Tutorial by Sun

What is a Servlet?

- A Servlet is a Java program that extends the capabilities of servers.
- Inherently multi-threaded.
  - Each request launches a new thread.
- Input from client is automatically parsed into a Request variable.

Servlet Life Cycle

- Servlet Instantiation:
  - Loading the servlet class and creating a new instance
- Servlet Initialization:
  - Initialize the servlet using the init() method
- Servlet processing:
  - Handling 0 or more client requests using the service() method
- Servlet Death:
  - Destroying the servlet using the destroy() method


Client Form:Client
Servlet-Enabled:Server
Lookup Static Page or Launch Process/Thread to Create Output
Http Request
Lookup
Launch Servlet
On first access launch the servlet program.
Launch Thread for Client
Http Response
Launch separate thread to service each request.
Http Response

Writing Servlets

- Install a web server capable of launching and managing servlet programs.
- Install the *javax.servlet* package to enable programmers to write servlets.
- Ensure *CLASSPATH* is changed to correctly reference the *javax.servlet* package.
- Define a servlet by subclassing the *HttpServlet* class and adding any necessary code to the *doGet()* and/or *doPost()* and if necessary the *init()* functions.

Handler Functions

- Each HTTP Request type has a separate handler function.
  - GET -> doGet(HttpServletRequest, HttpServletResponse)
  - POST -> doPost(HttpServletRequest, HttpServletResponse)
  - PUT -> doPut (HttpServletRequest, HttpServletResponse)
  - DELETE -> doDelete (HttpServletRequest, HttpServletResponse)
  - TRACE -> doTrace (HttpServletRequest, HttpServletResponse)
  - OPTIONS -> doOptions (HttpServletRequest, HttpServletResponse)

A Servlet Template

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class ServletTemplate extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {

    // Use "request" to read incoming HTTP headers
    // (e.g. cookies) and HTML form data (e.g. data the user
    // entered and submitted).

    // Use "response" to specify the HTTP response status
    // code and headers (e.g. the content type, cookies).

    PrintWriter out = response.getWriter();
    // Use "out" to send content to browser
  }
}
```

Important Steps

- Import the Servlet API:
```
import javax.servlet.*;
import javax.servlet.http.*;
```
- Extend the `HTTPServlet class`
  - Full servlet API available at:

`http://www.java.sun.com/products/servlet/2.2/javadoc/index.html`
- You need to overrride at least one of the request handlers!
- Get an output stream to send the response back to the client
  - All output is channeled to the browser.

doGet and doPost

- The handler methods each take two parameters:
  - `HTTPServletRequest`: encapsulates all information regarding the browser request.
    - Form data, client host name, HTTP request headers.
  - `HTTPServletResponse`: encapsulate all information regarding the servlet response.
    - HTTP Return status, outgoing cookies, HTML response.
- If you want the same servlet to handle both GET and POST, you can have doGet call doPost or vice versa.

```
Public void doGet(HttpServletRequest req, HttpServletResponse res) throws IOException
{doPost(req,res);}
```

Hello World Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWWW extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      out.println("<HTML>\n" +
                  "<HEAD><TITLE>Hello WWW</TITLE></HEAD>\n" +
                  "<BODY>\n" +
                  "<H1>Hello WWW</H1>\n" +
                  "</BODY></HTML>");
  }
}
```

getParameter()

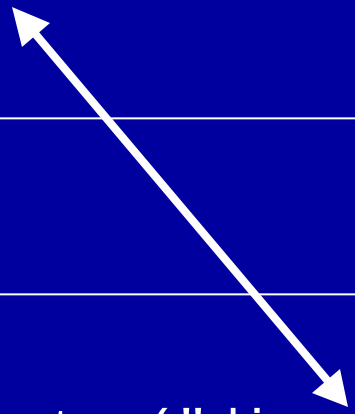
- Use *getParameter()* to retrieve parameters from a form by name.

Named Field values HTML FORM

```
<INPUT TYPE="TEXT" NAME="diameter">
```

In a Servlet

```
String sdiam = request.getParameter("diameter");
```



getParameter() *cont'd*

- getParameter() can return three things:
  - String: corresponds to the parameter.
  - Empty String: parameter exists, but no value provided.
  - null: Parameter does not exist.

getParameterValues()

- Used to retrieve multiple form parameters with the same name.
- For example, a series of checkboxes all have the same name, and you want to determine which ones have been selected.
- Returns an array of Strings.

getParameterNames()

- Returns an Enumeration object.
- By cycling through the enumeration object, you can obtain the names of all parameters submitted to the servlet.
- Note that the Servlet API does not specify the order in which parameter names appear.

Circle Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
public class circle extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
         response.setContentType("text/html");
         PrintWriter out = response.getWriter();
         out.println(   "<BODY><H1 ALIGN=CENTER> Circle Info </H1>\n");
         try{
           String sdiam = request.getParameter("diameter");
           double diam = Double.parseDouble(sdiam);
           out.println("<BR><H3>Diam:</H3>" + diam +
                  "<BR><H3>Area:</H3>" + diam/2.0 * diam/2.0 * 3.14159 +
                  "<BR><H3>Perimeter:</H3>" + 2.0 * diam/2.0 * 3.14159);
          } catch ( NumberFormatException e ){
           out.println("Please enter a valid number");
         }
    out.println("</BODY></HTML>");
  }
```

Subclass HttpServlet.

Specify HTML output.

Attach a PrintWriter to Response Object

Cookies and Servlets

- The HttpServletRequest class includes the “getCookies()” function.
  - This returns an array of cookies, or null if there aren’t any.
- Cookies can then be accessed using three methods.
  - String getName()
  - String getValue()
  - String getVersion()

Cookies & Servlets cont'd

- Cookies can be created using HttpServletResponse.addCookie() and the constructor new Cookie(String name, String value);
  - Expiration can be set using setMaxAge(int seconds)

Sessions & Servlets

- Servlets also support simple transparent sessions
  - Interface HttpSession
  - Get one by using HttpServletRequest.getSession()
- You can store & retrieve values in the session
  - putValue(String name, String value)
  - String getValue(String name)
  - String[] getNames()

Sessions & Servlets cont'd

- Various other information is stored
  - long getCreationTime()
  - String getId()
  - long getLastAccessedTime()
- Also can set timeout before session destruction
  - int getMaxInactiveInterval()
  - setMaxInactiveInterval(int seconds)