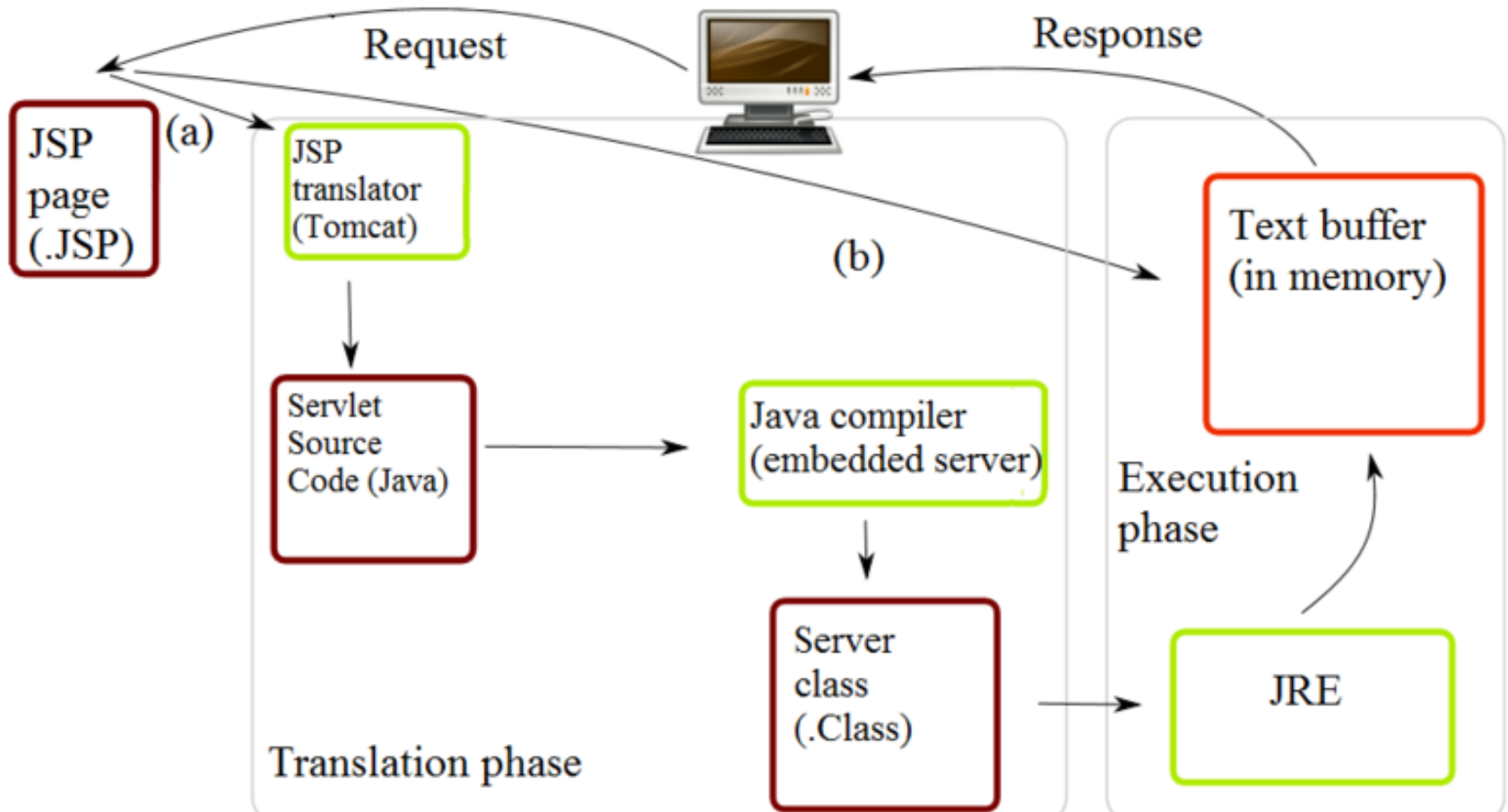


Java servlets



Overview

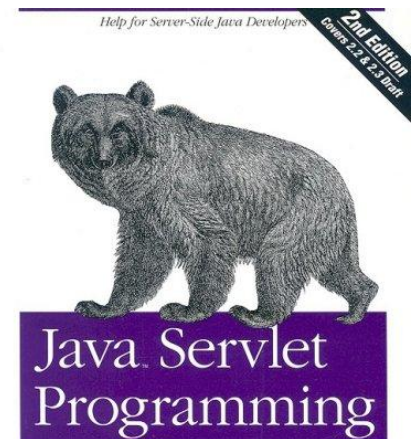
- **Dynamic web content generation** (thus far)
 - CGI
 - Web server modules
 - Server-side scripting, e.g. PHP, ASP, JSP
 - Custom web server
- **Java servlets** (today)
 - Java class implementing a specific interface
 - Override `doGet()`, `doPost()`
 - Hosted in a web-server with servlet support
 - e.g. Apache Tomcat, JBoss, Jetty, IBM Websphere
 - Or embedded in a standalone app

Why servlets?

- **Portability**
 - Written in Java, **works across different OSs**
- **Power**
 - Functionality of Java API and 3rd party classes
- **Efficiency and endurance**
 - Servlets **stay in memory** as single object instance
 - Can maintain **persistent state**
- **Safety**
 - **Strong typing**
 - Exception-handling
 - **Automatic garbage collection**

Why servlets?

- **Elegance**
 - Develop web-app in **high-level OO language**
- **Integration**
 - Tightly **integrated with server** compared to CGI
 - e.g. translate file paths, access to logging
- **Extensibility and flexibility**
 - Easy to extend to specialized needs
 - e.g. WebSocket server



What about applets?

- Applets

- Java **programs embedded in a web page**

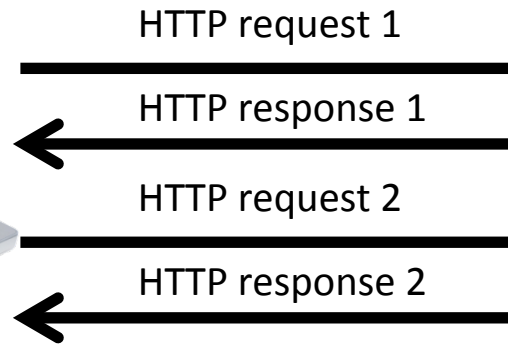
- <applet> tag in the HTML
- Appears in a box like other page elements (e.g. images)
 - But you can interact with it, pushing buttons, etc.
- Requires Java plug-in, not always available (iPhone)

- Operates in a **security "sandbox"**

- Not allowed to access:
 - Local file system, clipboard, arbitrary web sites, etc.

```
<html>
  <body>
    <h2>Don't panic about frogs!</h2>
    <applet code="PanicAppletParam.class" width="500" height="500">
      <param name="image" value="frog.jpg">
      <param name="sound" value="frog.wav">
Java plugin not installed
    </applet>
  </body>
</html>
```

Java servlets



Java web server and applet container

Web app 1

JSP page

JSP page

Servlet

Servlet

Web app 2

JSP page

JSP page

Servlet

Servlet

JSP = JavaServer Page

Server-side scripting language like PHP

Servlet = Java class that persists in the applet container

Servlet API

- Java servlet

- Normal Java class implementing the interface `javax.servlet.Servlet`

- Usually **extend `HttpServlet`**

- A HTTP protocol specific servlet, override:

```
void doGet(HttpServletRequest request,  
           HttpServletResponse response)  
void doPost(HttpServletRequest request,  
            HttpServletResponse response)
```

- Could extend `GenericServlet`

- Generic protocol independent servlet, override:

```
void service(ServletRequest request,  
            ServletResponse response)
```

- No `main()` method

Hello World servlet

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

public class HelloServlet extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
    {
        String      q      = req.getParameter("q");
        PrintWriter out = resp.getWriter();

        out.println("<html>");
        out.println("<body>");
        out.println("The parameter q was \"" + q + "\".");
        out.println("</body>");
        out.println("</html>");
    }

    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException
    {
        String      field = req.getParameter("field");
        PrintWriter out  = resp.getWriter();

        out.println("<html>");
        out.println("<body>");
        out.println("You entered \"" + field + "\" into the text box.");
        out.println("</body>");
        out.println("</html>");
    }
}
```


Servlet container

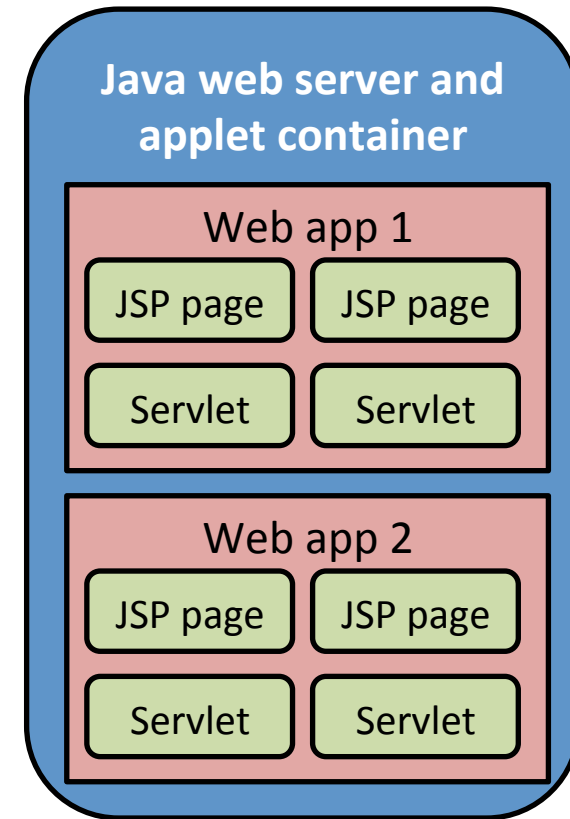
- Servlets live in a container
 - *Standalone* web server with servlet support
 - Bundle application in a Web application ARchive (WAR)
 - e.g. Apache Tomcat, Jetty, GlassFish,



jetty://



- *Add-on* to existing web server
 - e.g. Tomcat plugged into Apache
- *Embedded* in an application
 - e.g. Lightweight servlet deployment embedded in another application



An embedded diversion

- Embedding Jetty

- "Don't deploy your application in Jetty, deploy Jetty in your application"

```
public class HelloWorldEmbedded extends AbstractHandler
{
    public void handle(String target, Request baseRequest,
        HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html;charset=utf-8");
        response.setStatus(HttpServletResponse.SC_OK);
        baseRequest.setHandled(true);
        response.getWriter().println("<h1>Hello World</h1>");
    }

    public static void main(String[] args) throws Exception
    {
        Server server = new Server(8080);
        server.setHandler(new HelloWorldEmbedded());
        server.start();
        server.join();
    }
}
```

A Java application with an embedded web server.

HelloServlet v2.0

```
public class HelloServlet extends HttpServlet
{
    private String greeting = "Hello World";
    private int count = 0;

    public HelloServlet()
    {
    }

    public HelloServlet(String greeting)
    {
        this.greeting = greeting;
    }

    public int getCountIncrement()
    {
        return ++count;
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setContentType("text/html");
        response.setStatus(HttpServletResponse.SC_OK);
        response.getWriter().println("<h1>" + greeting+"</h1>");
        response.getWriter().println("<h1>You are visitor " + getCountIncrement() + "</h1>");
        response.getWriter().println("session=" + request.getSession(true).getId());
    }
}
```

Simple servlet that prints a greeting and a count of hits to the servlet.

Embedded servlet container

```
public class OneServletContext
{
    public static void main(String[] args) throws Exception
    {
        Server server = new Server(8080);

        ServletContextHandler context = new ServletContextHandler(ServletContextHandler.SESSIONS);
        context.setContextPath("/");
        server.setHandler(context);

        context.addServlet(new ServletHolder(new HelloServlet()), "/*");
        context.addServlet(new ServletHolder(new HelloServlet("Buongiorno Mondo")), "/it/*");
        context.addServlet(new ServletHolder(new HelloServlet("Bonjour le Monde")), "/fr/*");

        server.start();
        server.join();
    }
}
```

Main program that hosts three versions of the Hello World servlet.

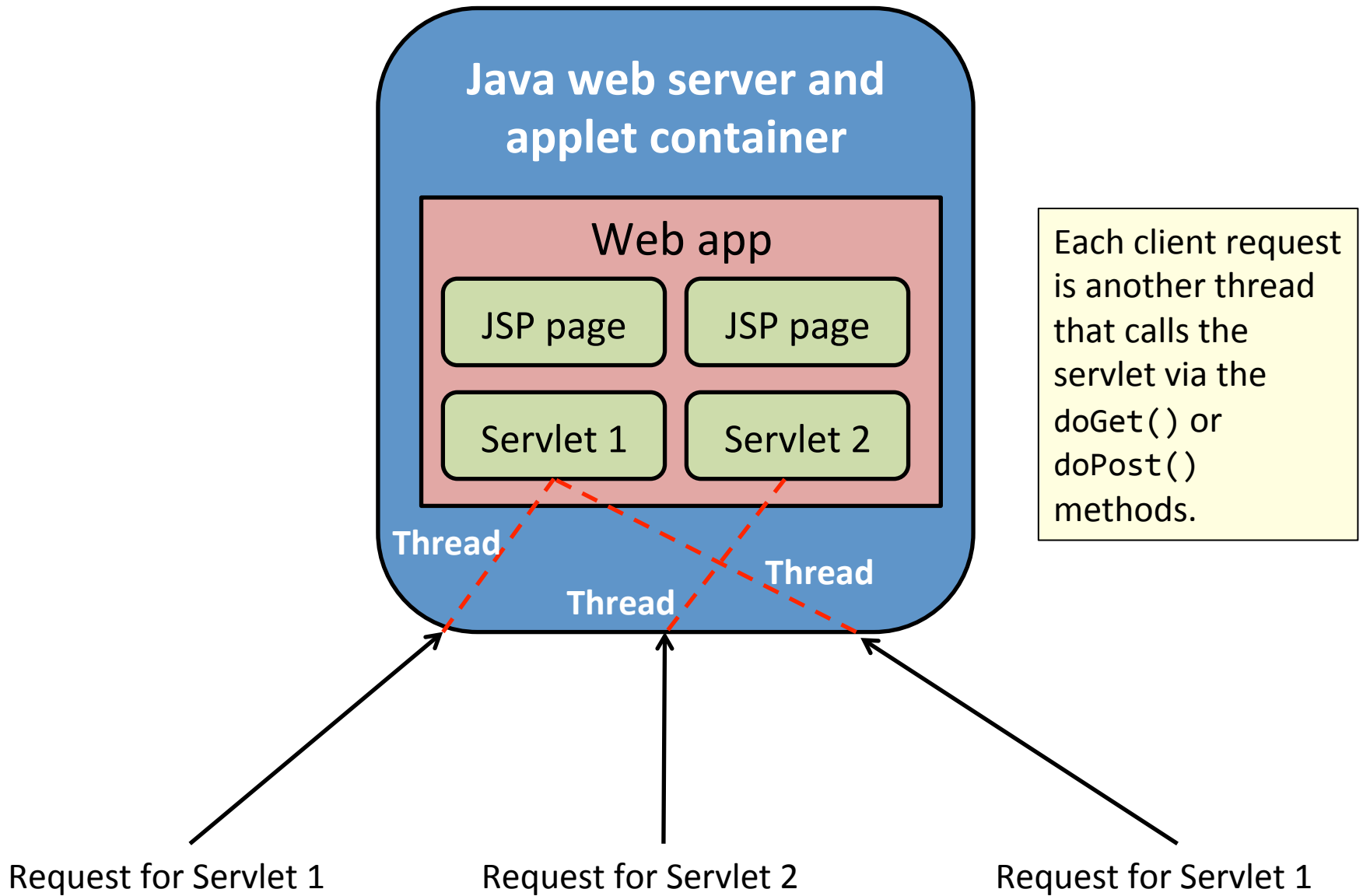
Servlet life cycle

- If instance of servlet does not exist, container:
 - Loads the servlet class
 - Creates an instance of the servlet class
 - Initialized the servlet by call to `init` method
 - May be called when server starts, when first requested, or at request of administrator
- Invokes the `service` method
 - Passing request and response objects
- If container needs to remove servlet:
 - Finalizes by calling `destroy` method

Servlet instance persistence

- **Servlets persist between requests**
 - Container holds an instance of an object
 - Requests to a given servlet serviced by same object
 - Improves performance:
 - Keeps **memory footprint small**
 - **Eliminates object creation** expense
 - **Avoid process startup** expense (as with CGI)
 - Servlet can **reuse resources**

Servlet thread model



Thread safety

- Non-local variables are not thread-safe!
 - e.g. instance variables of the class

```
public class HelloServlet extends HttpServlet
{
    private String greeting = "Hello World";
    private int    count    = 0;

    public HelloServlet(String greeting)
    {
        this.greeting = greeting;
    }

    public int getCountIncrement()
    {
        return ++count;
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setContentType("text/html");
        response.setStatus(HttpServletResponse.SC_OK);
        response.getWriter().println("<h1>" + greeting+"</h1>");
        response.getWriter().println("<h1>You are visitor " + getCountIncrement() + "</h1>");
        response.getWriter().println("session=" + request.getSession(true).getId());
    }
}
```


Testing our servlet's safety...

```
% ab -n 1000000 -c 1 http://localhost:8080/
```

```
Benchmarking localhost (be patient)
```

```
Completed 100000 requests
```

```
Completed 200000 requests
```

```
Completed 300000 requests
```

```
Completed 400000 requests
```

```
Completed 500000 requests
```

```
Completed 600000 requests
```

```
Completed 700000 requests
```

```
Completed 800000 requests
```

```
Completed 900000 requests
```

```
Completed 1000000 requests
```

```
Finished 1000000 requests
```

```
...
```

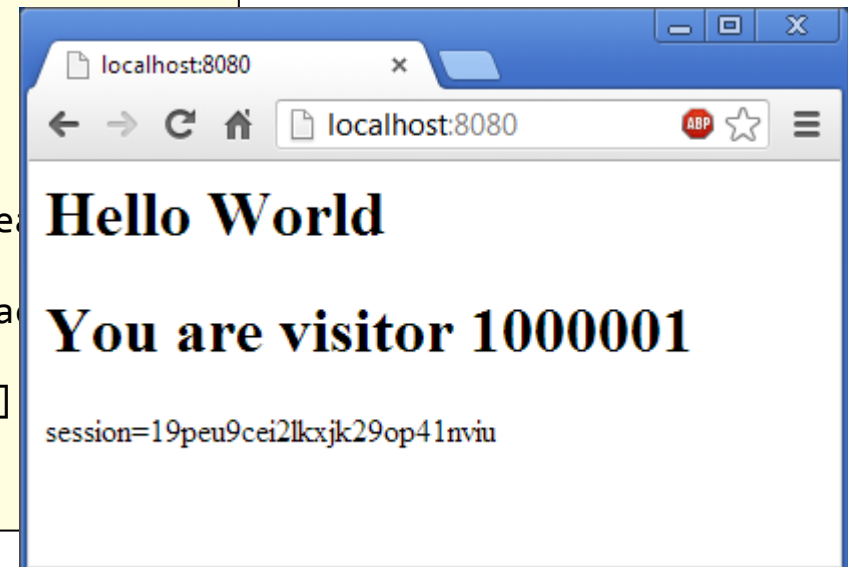
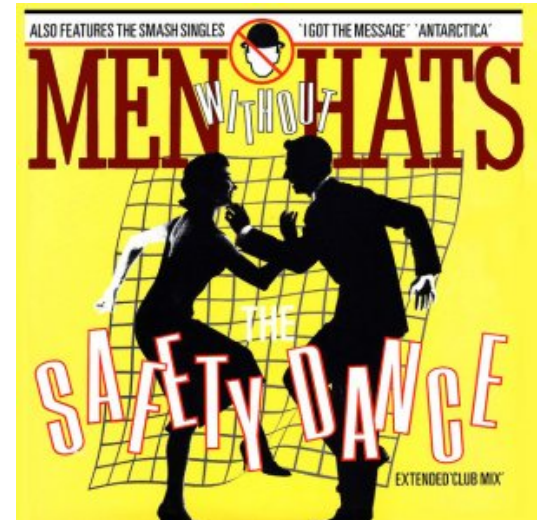
```
Requests per second: 2627.27 [# /sec] (mean)
```

```
Time per request: 0.381 [ms] (mean)
```

```
Time per request: 0.381 [ms] (mean, across  
concurrent requests)
```

```
Transfer rate: 761.43 [Kbytes/sec]
```

```
...
```



Testing our servlet's safety...

```
% ab -n 1000000 -c 10 http://localhost:8080/
```

```
Benchmarking localhost (be patient)
```

```
Completed 100000 requests
```

```
Completed 200000 requests
```

```
Completed 300000 requests
```

```
Completed 400000 requests
```

```
Completed 500000 requests
```

```
Completed 600000 requests
```

```
Completed 700000 requests
```

```
Completed 800000 requests
```

```
Completed 900000 requests
```

```
Completed 1000000 requests
```

```
Finished 1000000 requests
```

```
...
```

```
Requests per second: 6569.71 [# /sec] (r
```

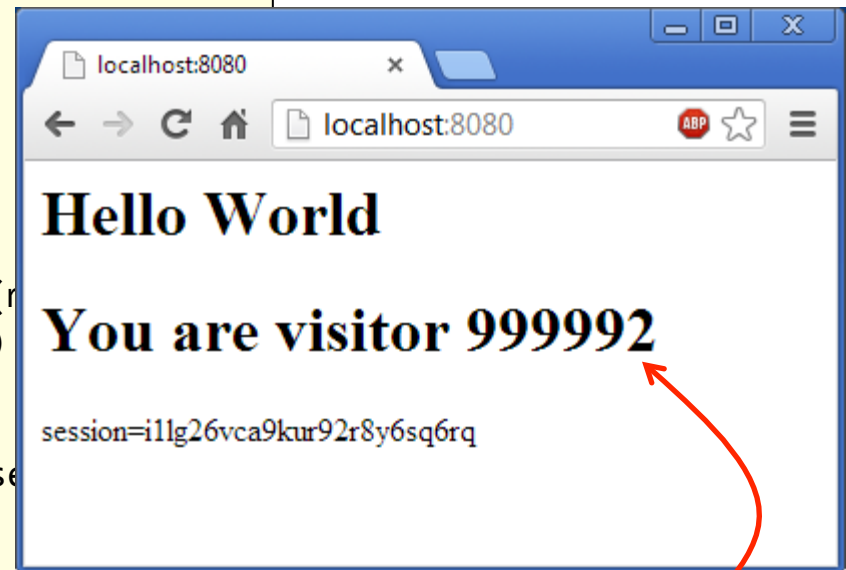
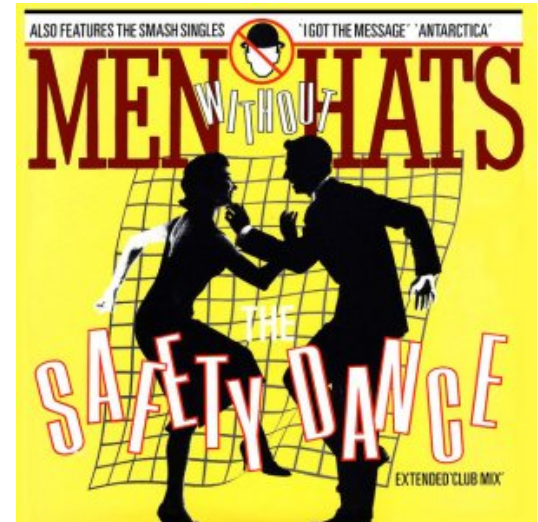
```
Time per request: 1.522 [ms] (mean)
```

```
Time per request: 0.152 [ms] (mean,
```

```
concurrent requests)
```

```
Transfer rate: 1904.04 [Kbytes/s]
```

```
...
```



9 updates went missing!

Thread safety

- Synchronize access to shared data!
 - e.g. instance variables of the class

```
public class HelloServlet extends HttpServlet
{
    private String greeting = "Hello World";
    private int    count    = 0;

    public HelloServlet(String greeting)
    {
        this.greeting = greeting;
    }

    public int synchronized getCountIncrement()
    {
        return ++count;
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setContentType("text/html");
        response.setStatus(HttpServletResponse.SC_OK);
        response.getWriter().println("<h1>" + greeting+"</h1>");
        response.getWriter().println("<h1>You are visitor " + getCountIncrement() + "</h1>");
        response.getWriter().println("session=" + request.getSession(true).getId());
    }
}
```

Now you can really bang on it...

```
% ab -n 1000000 -c 10 http://localhost:8080/
```

```
Benchmarking localhost (be patient)
```

```
Completed 100000 requests
```

```
Completed 200000 requests
```

```
Completed 300000 requests
```

```
Completed 400000 requests
```

```
Completed 500000 requests
```

```
Completed 600000 requests
```

```
Completed 700000 requests
```

```
Completed 800000 requests
```

```
Completed 900000 requests
```

```
Completed 1000000 requests
```

```
Finished 1000000 requests
```

```
...
```

```
Requests per second: 6690.28 [# /sec] (mean)
```

```
Time per request: 1.495 [ms] (mean)
```

```
Time per request: 0.149 [ms] (mean, across all concurrent requests)
```

```
Transfer rate: 1938.98 [Kbytes/sec]
```

```
...
```



HttpServletRequest

- Encapsulates all info from client request
 - HTTP request header and body
- Retrieve data using methods
 - Inherited from ServletRequest:
 - String getParameter(String name)
 - Enumeration<String> getParameterNames()
 - String[] getParameterValues(String name)
 - ServletInputStream getInputStream()
 - BufferedReader getReader()

```
protected void doGet(HttpServletRequest request, HttpServletResponse response);  
protected void doPost(HttpServletRequest request, HttpServletResponse response);
```

HttpServletResponse

- Encapsulates all data returned to client
 - Set HTTP response header:
 - void setStatus(int sc)
 - void setHeader(String name, String value)
 - void setContentType(String type)
 - void sendRedirect(String location)
 - void sendError(int sc)
 - Set HTTP response body:
 - Obtain PrintWriter or ServletOutputStream to return data to client
 - PrintWriter getWriter()
 - ServletOutputStream getOutputStream()

```
protected void doGet(HttpServletRequest request, HttpServletResponse response);  
protected void doPost(HttpServletRequest request, HttpServletResponse response);
```

Session management: adding cookies

- Tracking user's state with cookies
 - Cookie class:
 - Cookie(String name, String value)
 - void setMaxAge(int expiry)
 - void setDomain(String pattern)
 - void setPath(String uri)
 - void setSecure(boolean flag)
 - HttpServletResponse class:
 - void addCookie(Cookie cookie)

```
HTTP/1.1 200 OK
Date: Thu, 17 Nov 2011 15:54:10 GMT
Server: Jetty(9.0.0.RC2)
Content-Length: 285
Set-Cookie: sessionId=528fa623; path=/; Expires=Wed, 09 Mar 2014 11:00:00 GMT

<html><body>
...
```

Session management: retrieving cookies

- Check what cookies are set in HTTP request
 - HttpServletRequest class:
 - `Cookie[] getCookies()`
 - Cookie class:
 - `String getName()`
 - `String getValue()`
 - `String getDomain()`
 - `String getPath()`
 - `boolean getSecure()`

```
GET / HTTP/1.1
Host: katie.mtech.edu
User-agent: Mozilla/4.0
Cookie: sessionId=528fa623
```


Servlet's built-in session tracking

- Most servers support session tracking
 - HTTP cookie provides session key
 - Servlet uses key to retrieve session state
- Session objects maintained in memory
 - Some servers allow writing to file system or database
 - Objects stored in session need to be serializable
- User associated with HttpSession object:
 - HttpSessionRequest class
 - Get current session or create new one:
 - HttpSession getSession()

HttpSession

- Getting data:

- Object `getAttribute(String name)`
- Enumeration<String> `getAttributeNames()`

- Setting data:

- void `setAttribute(String name, Object val)`

- Moving data:

- void `removeAttribute(String name)`

Session lifecycle

- Sessions do not last forever
 - Automatically expire after period of inactivity
 - `int getMaxInactiveInterval()`
 - `void setMaxInactiveInterval()`
 - Explicitly invalidated by servlet
 - `void invalidate()`
- Inactive or invalidated session
 - `HttpSession` object removed along with data values it contains

Summary

- Java servlets
 - Another way to do **dynamic content generation**
 - Run a web server with servlet support or embedded in an existing app
 - Development in **high-level object-oriented language**
 - Persistent servlet object that services requests
 - Efficient **reuse of the same object**
 - Can be hit in parallel by multiple threads
 - **Protect concurrent access** to shared data!
 - Support for tracking session state