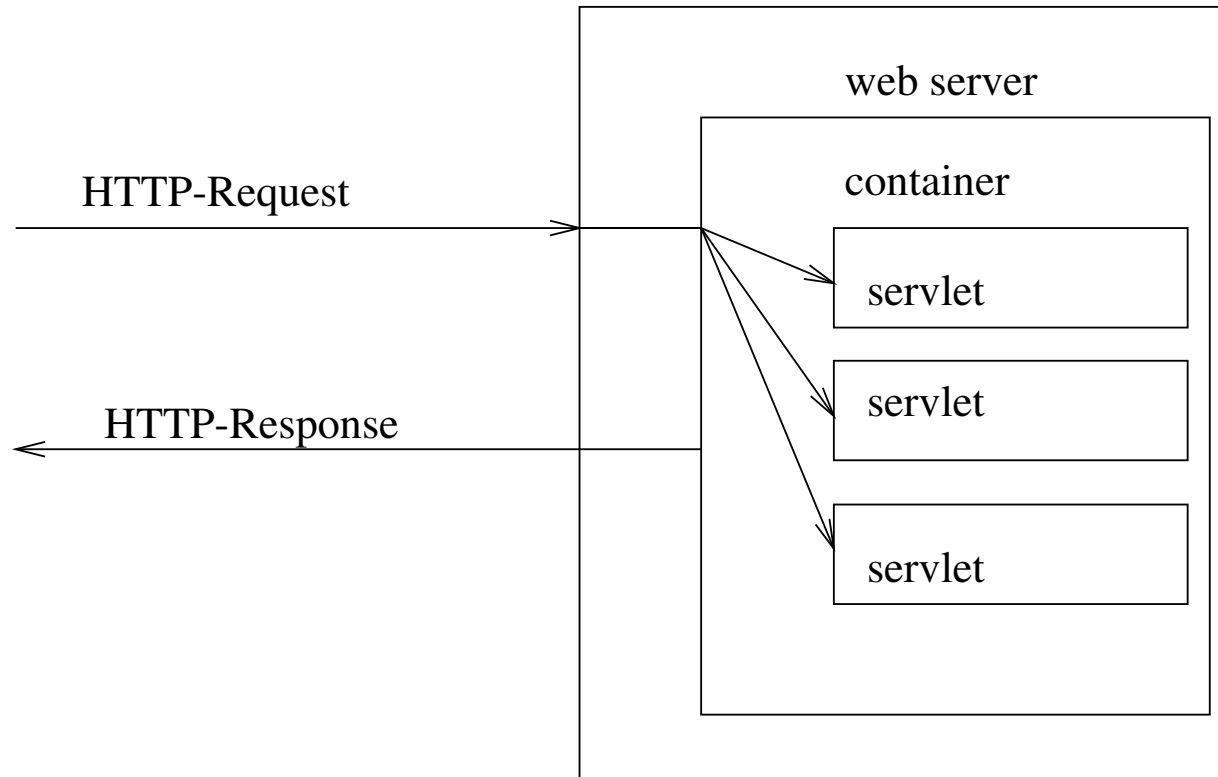


## 9.4 Java Servlet technology - “Servlets”

From the [Java<sup>TM</sup> Servlet Specification, v2.4]:

A servlet is a Java<sup>TM</sup> technology-based Web component, managed by a container, that generates dynamic content. Like other Java technology-based components, servlets are platform-independent Java classes that are compiled to platform-neutral byte code that can be loaded dynamically into and run by a Java technology-enabled Web server. Containers, sometimes called servlet engines, are Web server extensions that provide servlet functionality. Servlets interact with Web clients via a request/response paradigm implemented by the servlet container.

## 9.4.1 Servlet Grundlagen



# Servlet Container

- Standort
  - im Server-Prozess
  - in anderem Prozess auf Server-Maschine
  - auf anderer Maschine
- Aufgaben
  - Dekodierung von Formulardaten
  - Verbindungsparameter
  - Zustandsverwaltung

## 9.4.2 Servlet API

- Lebenszyklus eines Servlets
  - Laden und Instanzieren durch Servlet-Container
  - Initialisieren, Methode `init (...)`
  - Anfrageverarbeitung, Methode `service (...)`  
HTTP-Anfrage durch `HttpServletRequest`-Objekt  
HTTP-Antworten durch `HttpServletResponse`-Objekt
  - Beenden, Methode `destroy ()`
- Pfad eines Servlets
  - Kontext, identifiziert einen Container/Anwendung (Klasse `ServletContext`)
  - Servlet Name
  - zusätzliche Pfadinformation

# Beispiel

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

public class Echo extends HttpServlet {
    public void doGet (HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType ("text/html; charset=utf-8");
        PrintWriter out = response.getWriter ();
        out.println ("<html>");
        out.println ("<head><title>Echo Results</title></head>");
        out.println ("<body><h3>Echo Results</h3><ul>");
        Enumeration e = request.getParameterNames ();
        while (e.hasMoreElements ()) {
            String name = (String)e.nextElement ();
            String value = request.getParameter (name);
            out.print ("<li>");
            out.println (name + ": " + value);
        }
        out.println ("</ul></body></html>");
    }
}
```

# Abstrakte Klasse HttpServlet

- vordefinierte Methoden, entsprechend den HTTP-Methoden
  - `void doGet (HttpServletRequest request, HttpServletResponse response)`
  - `doPost (...)`
  - `doPut (...)`
  - `doDelete (...)`
  - `doHead (...)`
  - `doOptions (...)`
  - `doTrace (...)`
  - `long getLastModified (HttpServletRequest req)`
  - `void service (ServletRequest req, ServletResponse res)`

# Verwendung von HttpServlet

- Anwendung überschreibt mindestens eine der Methoden doGet, doPost, doPut, doDelete
- ggf. init und destroy (aus Superklasse GenericServlet)  
Aufgabe zB. Aufbau bzw Abbau einer Datenbankverbindung
- ggf. String getServletInfo () (aus GenericServlet)

Achtung:

- ggf. sind mehre Instanzen des gleichen Servlets aktiv
- Behandlung von Requests **muss** Threadsynchronisation verwenden

## Beispiel für Lebenszyklus

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

public class HelloWorld2 extends HttpServlet {
    public String getServletInfo() {
        return "servlet example, by Møller & Schwartzbach";
    }

    public void init() {
        log("initializing");
    }
}
```



```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws IOException, ServletException {
    log("a thread is entering doGet");

    response.setContentType("text/html; charset=utf-8");
    PrintWriter out = response.getWriter();
    out.println("<html><head>");
    out.println("<title>Servlet Example</title></head>");
    out.println("<body><h1>Hello World!</h1>");
    out.println("This page was last updated: "+new java.util.Date());
    out.println("</body></html>");

    log("a thread is leaving doGet");
}
```

```
public void doPost(HttpServletRequest request,
                    HttpServletResponse response)
    throws IOException, ServletException {
    response.sendError(response.SC_METHOD_NOT_ALLOWED,
                       "Kannitverstan POST!");
}

public void destroy() {
    log("shutting down");
}
}
```

# Log vom Ablauf des Beispiels

```
2004-12-25 12:41:21 StandardContext [/IXWT>HelloWorld2:  
    initializing  
2004-12-25 12:41:25 StandardContext [/IXWT>HelloWorld2:  
    a thread is entering doGet  
2004-12-25 12:41:25 StandardContext [/IXWT>HelloWorld2:  
    a thread is leaving doGet  
2004-12-25 12:41:34 StandardContext [/IXWT>HelloWorld2:  
    a thread is entering doGet  
2004-12-25 12:41:34 StandardContext [/IXWT>HelloWorld2:  
    a thread is entering doGet  
2004-12-25 12:41:34 StandardContext [/IXWT>HelloWorld2:  
    a thread is leaving doGet  
2004-12-25 12:41:34 StandardContext [/IXWT>HelloWorld2:  
    a thread is leaving doGet  
2004-12-25 12:42:47 StandardContext [/IXWT>HelloWorld2:  
    shutting down
```

## 9.4.3 Interface HttpServletRequest

Zugriff auf Header

- Enumeration `getHeaderNames ()`
- `String getHeader (String name)`
- Enumeration `getHeaders (String name)`
  
- `int getIntHeader (String name)`
- `long getDateHeader (String name)`

# HttpServletRequest / ServletRequest

Zugriff auf Formulardaten / HTTP message body

- Enumeration `getParameterNames ()`
- `String getParameter (String name)`
- Enumeration `getParameterValues (String name)`
- `ServletInputStream getInputStream ()`

`getParameter...` und `getInputStream` nicht zusammen verwenden!

Kodierung der Formularparameter ist transparent.

- Zugriff auf Verbindungsinformation
  - `String getRemoteHost ()`
  - `String getRemoteAddr ()`
  - `int getRemotePort ()`
- Zugriff auf Pfadteile
  - `String getContextPath ()`
  - `String getServletPath ()`
  - `String getPathInfo ()`
- Zugriff auf Cookies
  - `Cookie[] getCookies ()`

## Beispiel zu HttpServletRequest

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

public class Requests extends HttpServlet {

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

    // diese Umleitung (Delegation) ist immer möglich
    // auch mit Formulardaten!
```

```

public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
    throws IOException, ServletException {
    response.setContentType("text/html; charset=utf-8");
    PrintWriter out = response.getWriter();
    out.println("<html><head><title>Requests</title></head><body>");
    out.println("<h1>Hello, visitor from "+
                request.getRemoteHost()+"</h1>");
    String useragent = request.getHeader("User-Agent");
    if (useragent!=null)
        out.println("You seem to be using "+useragent+"<p>");
    String name = request.getParameter("name");
    if (name==null)
        out.println("No <tt>name</tt> field was given!");
    else
        out.println("The value of the <tt>name</tt> field is: <tt>" +
                    htmlEscape(name) + "</tt>");
    out.println("</body></html>");
}

```



```

private String htmlEscape(String s) {
    StringBuffer b = new StringBuffer();
    for (int i = 0; i<s.length(); i++) {
        char c = s.charAt(i);
        switch (c) {
            case '<': b.append("&lt;"); break;
            case '>': b.append("&gt;"); break;
            case '"': b.append("&quot;"); break;
            case '\\': b.append("&apos;"); break;
            case '&': b.append("&amp;"); break;
            default: b.append(c);
        }
    }
    return b.toString();
}
}

```

# Über Formulardaten

Eine Webanwendung sollte **niemals** ...

- auf Formularfelder zugreifen ohne vorher deren Anwesenheit zu überprüfen  
vgl. User-Agent  
Warum? Robustheit der Anwendung: Ein bössartiger Anwender/Tester/... kann
  - ein Formular abgespeichern, verändern, wieder in den Browser laden und dann abschicken.
  - ein HTTP-Request unabhängig vom Formular durch ein Programm oder händisch erzeugen.
- den Inhalt einer Formulareingabe (Datenbankausgabe, Datei, ...) ungefiltert in eine XHTML Seite einbinden.  
vgl. `htmlEscape`  
Warum? Sicherheit: Ein bössartiger Anwender/Tester/... kann in die Eingabe eigene XHTML-Elemente einbauen, die zB die Identität eines Benutzers ausspähen können (cross site scripting).

## 9.4.4 Interface HttpServletResponse

- Der Servletcontainer übergibt einen Puffer mit einem vorbereiteten Response-Objekt
- Das Response-Objekt sammelt Information für den Response-Header und für die Kodierung des Response-Body
- Die Information im Response-Objekt wird gültig (*committed*), sobald der Puffer für die Response-Body Daten zum ersten Mal geschrieben wird.
- Wenn das Response-Objekt committed ist, haben einige Methoden keine Wirkung mehr, andere liefern eine Exception.

# HttpServletResponse

- Statuskode und Fehler
  - `void setStatus (int sc)`
  - `void sendRedirect (String url)`
  - `void sendError (int statusCode, String message)`
- Hinzufügen/Überschreiben von Headern
  - `void setHeader (String name, String value)`
  - `void addHeader (String name, String value)`
  - `void setIntHeader (String name, int value)`
  - `void addIntHeader (String name, int value)`
  - `void setDateHeader (String name, long value)`
  - `void addDateHeader (String name, long value)`

- Ausgabe

- `ServletOutputStream getOutputStream()`

- Ausgabe eines Oktettstream

- `PrintWriter getWriter()`

- Ausgabe eines Zeichenstream

- `void setContentType(String type)`

- sollte **immer** gesetzt werden, inklusive character encoding

- Typisches Argument: `text/html; charset=utf-8`

Entweder `getOutputStream` oder `getWriter`. Niemals zusammen.

Explizites oder implizites `flush()` des Ausgabestream bewirkt `commit` des Response-Objekts

- Pufferung

- `flushBuffer ()` bewirkt `commit`

- `bool isCommitted ()`

- `void reset ()` nur *vor* `commit`

## 9.4.5 Zustand in Webanwendungen

Verschiedene Arten von Zustand in einer Webanwendung

- flüchtiger Zustand  
spezifisch für eine Antwort  
⇒ Parameter und lokale Variablen in doGet, doPost, etc
- Sitzungszustand  
spezifisch für eine Gruppe von Anfragen und Antworten  
Bsp: Einkaufswagen  
Lebensdauer: Sitzung bzw Neustart des Servletcontainers  
⇒ Session-Objekt
- globaler Zustand  
spezifisch für Anwendung oder Server  
Lebensdauer: persistent, bestimmt durch Anwendung  
⇒ Kontextattribute, Datenbank

# Globaler Zustand

- Durch ServletContext-Objekt
- `GenericServlet.getServletContext()`
- bestimmt durch den *context path* (URL-Präfix)
- Servletcontainer erzeugt **ein** ServletContext-Objekt pro Anwendung/context path
- Anwendungszustand mit *Attributen* des ServletContext
  - `void setAttribute(String name, Object object)`
  - `Object getAttribute(String name)`
- Attribute besser als Instanzvariable, da mehrere Instanzen eines Servlets existieren können
- Kontext einer anderen Anwendung:  
`ServletContext getContext(String uripath)`  
meist aus Sicherheitsgründen verboten (konfigurierbar)

# Beispiel: Ein Umfragedienst

- Formular zum Start einer Umfrage  
`QuickPollQuestion.html`
- Servlet zur Verarbeitung dieses Formulars  
`QuickPollSetup.java`
- Servlet zum Stellen der Frage  
`QuickPollAsk.java`
- Servlet zum Verarbeiten einer Antwort  
`QuickPollVote.java`
- Servlet zur Anzeige des Ergebnisses  
`QuickPollResults.java`



# QuickPollQuestion.html

```
<html>
  <head><title>QuickPoll</title></head>
  <body>
    <h1>QuickPoll</h1>
    <form method="post" action="setup">
      What is your question?<br />
      <input name="question" type="text" size="40" /><br />
      <input name="submit" type="submit"
        value="Register my question" />
    </form>
  </body>
</html>
```

```

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

public class QuickPollSetup extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        String q = request.getParameter("question");
        ServletContext c = getServletContext();
        c.setAttribute("question", q);
        c.setAttribute("yes", new Integer(0));
        c.setAttribute("no", new Integer(0));
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        out.print("<html><head><title>QuickPoll</title></head><body>" +
            "<h1>QuickPoll</h1>" +
            "Your question has been registered.<br />" +
            "Let the voting begin!" +
            "</body></html>");
    }
}

```

```

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

public class QuickPollAsk extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        out.print("<html><head><title>QuickPoll</title></head><body>" +
            "<h1>QuickPoll</h1>" +
            "<form method=\"post\" action=\"vote\">");
        String question = (String)getServletContext().getAttribute("question");
        out.print(question+"?<br />");
        out.print("<input name=\"vote\" type=\"radio\" value=\"yes\" /> yes<br />" +
            "<input name=\"vote\" type=\"radio\" value=\"no\" /> no<br />" +
            "<input name=\"submit\" type=\"submit\" value=\"Vote\" />" +
            "</form>" +
            "</body></html>");
    }
}

```

```

// imports ...

public class QuickPollVote extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        String vote = request.getParameter("vote");
        ServletContext c = getServletContext();
        if (vote.equals("yes")) {
            int yes = ((Integer)c.getAttribute("yes")).intValue();
            c.setAttribute("yes", new Integer(yes++));
        } else if (vote.equals("no")) {
            int no = ((Integer)c.getAttribute("no")).intValue();
            c.setAttribute("no", new Integer(no++));
        }
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        out.print("<html><head><title>QuickPoll</title></head><body>"+
            "<h1>QuickPoll</h1>Thank you for your vote!</body></html>");
    }
}

```

```
// import...

public class QuickPollResults extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        ServletContext c = getServletContext();
        String question = (String)c.getAttribute("question");
        int yes = ((Integer)c.getAttribute("yes")).intValue();
        int no = ((Integer)c.getAttribute("no")).intValue();
        int total = yes+no;
    }
}
```

```

response.setContentType("text/html; charset=utf-8");
response.setDateHeader("Expires", 0);
response.setHeader("Cache-Control", "no-store, no-cache, must-revalidate");
response.setHeader("Pragma", "no-cache");
PrintWriter out = response.getWriter();
out.print("<html><head><title>QuickPoll</title></head><body>"+
        "<h1>QuickPoll</h1>");
if (total==0)
    out.print("No votes yet on: " + question + "!");
else {
    out.print(
        question + "?<br />"+
        "<table border=\"0\">"+
        "<tr><td>Yes:</td><td>"+drawBar(300*yes/total)+"</td><td>"+yes+"</td></tr>"+
        "<tr><td>No:</td> <td>"+drawBar(300*no/total) +"</td><td>"+no+ "</td></tr>"+
        "</table>");
}
out.print("</body></html>");
}

```

```
String drawBar(int length) {  
    return "<table><tr><td bgcolor=\"black\" height=\"20\" width=\""+  
        length+"\"></td></tr></table>";  
}  
}
```

# Bemerkungen zu QuickPoll

- doGet zur Abfrage, doPost zur Veränderung verwendet
- Keine Authentisierung für QuickPollSetup
- Keine Filterung der eingegebenen Frage (cross-site scripting)
- Kein Test, ob getParameter null liefert
- Was passiert, wenn die Servlets in der falschen Reihenfolge ablaufen?
- Kein Test, ob die Attribute Objekte der erwarteten Typen enthalten
- *Race Conditions*: Falls mehrere QuickPollVote gleichzeitig laufen, können Stimmen verloren gehen
- Zustand (Frage und Stimmen) kann bei Servercrash verloren gehen
- Redundante XHTML Generierung