Java–Servlets

Eine Einführung

Mitgelieferte Bibliothekselemente

Inhalt

- Überblick
- Funktionsweise und Architektur
- Interaktion
- Resümee

Mitgelieferte Bibliothekselemente

Überblick

- Was sind Servlets?
- Warum Servlets?
- Servlets schreiben
- Web-Server
- Praktischer Einsatz
- Alternativen

Mitgelieferte Bibliothekselemente

Was sind Servlets?

- Serverseitige Anwendungen
- Ersetzen CGI-Scripts
- Dynamische Erstellung von HTML-Dokumenten

Mitgelieferte Bibliothekselemente

Warum Servlets?

- Effizient
  - Ein Servlet wird nur bei der ersten Ausführung geladen; danach nur Aufruf der Methoden

Mitgelieferte Bibliothekselemente

Warum Servlets?

- Persistent
  - Servlets erhalten ihren Zustand zwischen Requests aufrecht; sie bleiben resident im Speicher

Mitgelieferte Bibliothekselemente

Warum Servlets?

- Portabel
  - Da sie in Java geschrieben sind: „Write once, serve everywhere“
- Robust
  - Dank Zugriff auf das gesamte JDK

Mitgelieferte Bibliothekselemente

Warum Servlets?

- Sicherheit
  - Serverseitige Ausführung: es werden die Sicherheitsmechanismen des Web-Servers übernommen, z.B. SSL
  - Client hat keinen direkten Zugriff
  - Sicherung auch per Java Security Manager möglich

Mitgelieferte Bibliothekselemente

Warum Servlets?

- Erweiterbar
  - Da sie in einer OO-Sprache geschrieben sind können sie beliebig ausgebaut werden
- Akzeptanz
  - Viele Hersteller bieten Web-Server mit Unterstützung für Servlets

Mitgelieferte Bibliothekselemente

Web-Server

- Integrierte Servlet Unterstützung
  - Lotus Domino Go Web Server
  - Live Software Jrun
  - Sun Java Web Server
  - W3C Jigsaw Server
  - IBM Visual Age WebRunner Toolkit
  - Netscape Enterprise Server
  - U.v.m.

Mitgelieferte Bibliothekselemente

Web-Server

- Unterstützung als Add-On
  - Apache Web Server
  - New Atlanta ServletExec
  - Gefion Software WAICoolRunner

Mitgelieferte Bibliothekselemente

Servlets schreiben

- Benötigt ausserdem:
  - JDK (Java Development Kit)
  - JSDK (Java Servlets Development Kit)
    - Ist integriert in das JDK ab v1.2
  - J2EE (Java Enterprise Development Kit)

Mitgelieferte Bibliothekselemente

Servlets schreiben

- Development Tools
  - Symantec Visual Cafe Pro
  - Borland JBuilder

Mitgelieferte Bibliothekselemente

Praktischer Einsatz

- Anwendungen, die dynamisch erstellte HTML-Seiten benötigen
  - Warenkorbsysteme
  - News-/Diskussionsforen
  - Zugriff auf alte Datenbestände

Mitgelieferte Bibliothekselemente

Alternativen

- CGI Scripts
- ServerSide JavaScript
- Microsoft Active Server Pages

Mitgelieferte Bibliothekselemente

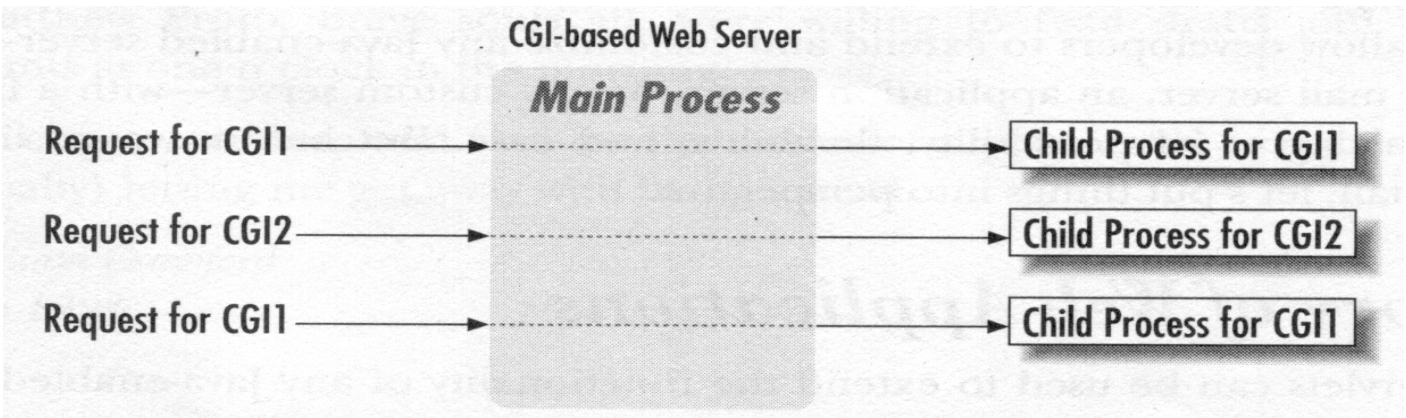
Funktionsweise und Architektur

- Vergleich CGI Scripts & Servlets
- Life-Cycle
- Implementierung
- Servlet Beispiele

Mitgelieferte Bibliothekselemente

Vergleich: CGI Scripts

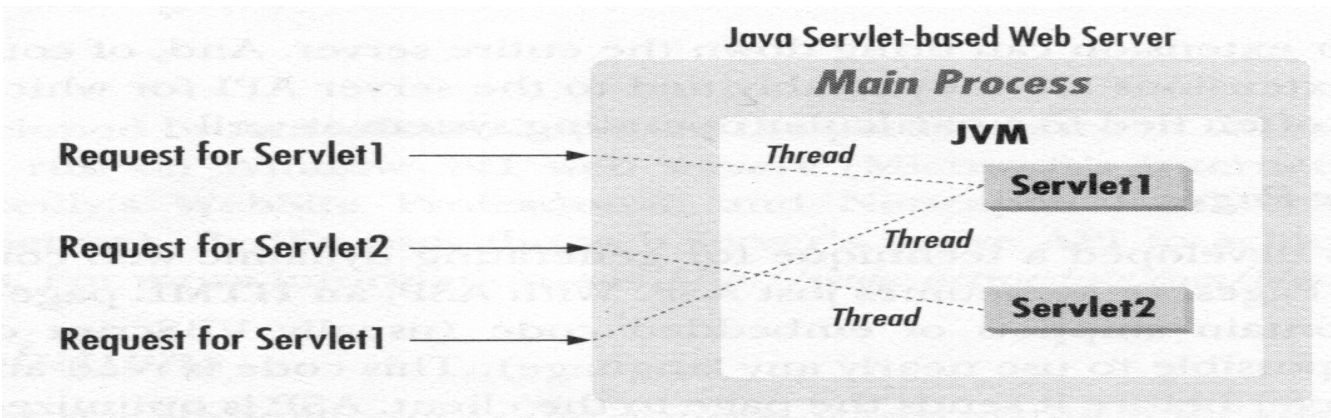
- Für jeden Request wird ein neuer Prozess gestartet
- Prozesse laufen getrennt vom Web-Server
- Bei jedem Request muss neu interpretiert werden
- Bei Perl: jeweils auch ein neuer Prozess für den Interpreter



Mitgelieferte Bibliothekselemente

Vergleich: Servlets

- Für jeden Request wird ein neuer Thread gestartet
- Threads laufen im Server-Prozess
- Kompilierte Binaries müssen nicht mehr interpretiert werden



Mitgelieferte Bibliothekselemente

Life Cycle

- Einfaches OO-Design
- Drei wichtige Methoden im Leben eines Servlets:
  - `init()`
    - Einmaliger Aufruf durch Server nachdem eine Instanz des Servlets erstellt wurde
    - Ressourcen des Servlets werden erzeugt und initialisiert

Mitgelieferte Bibliothekselemente

Life Cycle

- `service()`
  - Behandelt jeden Request eines Clients
  - Aufruf erst nachdem `init()` ausgeführt wurde
  - Meist nicht direkt implementiert; normalerweise in der Klasse `HttpServlet` implementiert
  - `ServletRequest` und `ServletResponse`

Mitgelieferte Bibliothekselemente

Life Cycle

- `destroy()`
  - Beim Schliessen eines Servlets am Ende ausgeführt
  - Garbage collection: Cleanup aller Ressourcen die in `init()` erzeugt wurden
  - Methode, um Informationen für den nächsten Start des Servlets zu speichern

Mitgelieferte Bibliothekselemente

Implementierung I

- GET Methode
  - Servlet wird „direkt“ aus dem Browser aufgerufen
  - Es wird eine neue HTML-Seite generiert
  - http://host/servlets/Servlet

Mitgelieferte Bibliothekselemente

Implementierung II

- POST Methode
  - Servlet wird z.B. von einem Formular aufgerufen
  - Es wird eine neue HTML-Seite generiert

  - `<form action=http://host/servlets/Servlet method=POST> ... </form>`

Mitgelieferte Bibliothekselemente

Implementierung III

- Kombination von GET und POST Methode, Beispiel:
  - Beim Aufruf des Servlets wird ein Formular angezeigt: GET Methode
  - Das gleiche Servlets verarbeitet die gesendeten Informationen des Formulars: POST Methode

Mitgelieferte Bibliothekselemente

Implementierung IV

- Server Side Includes – SSI
  - Servlet wird mit Hilfe des <SERVLET> Tags in eine bestehende HTML-Seite (Endung .JHTML bzw. .SHTML) eingebunden
  - Ausgabe des Servlets wird dynamisch in die HTML-Seite eingefügt
  - `<servlet code=Servlet.class> ... </servlet>`

Mitgelieferte Bibliothekselemente

Implementierung V

- JavaServer Pages – JSP
  - Sogenannte Scriplets werden direkt in die HTML-Seite eingebunden (Endung .JSP)
  - Ausgabe des Scriplets wird dynamisch in die HTML-Seite eingefügt

  - `<% request.getRemoteAddress() %>`

Mitgelieferte Bibliothekselemente

Servlet Beispiel I

```
public class ServletName extends HttpServlet
{

  public void init(ServletConfig config)
         throws ServletException
  {
    super.init(config);
  }
```

Mitgelieferte Bibliothekselemente

Servlet Beispiel I – Forts.

```
  public void doGet(HttpServletRequest req,
                    HttpServletResponse resp)
         throws ServletException,IOException
  {
    resp.setContentType(„text/html“);
    PrintWriter = resp.getWriter();
    out.println(„<html><body>“);
    out.println(„Hello World“);
    out.println(„</body></html>“);
    out.close();
  }
}
```

Mitgelieferte Bibliothekselemente

Servlet Beispiel II

```
public class ServletName extends HttpServlet
{

  public void init(ServletConfig config)
         throws ServletException
  {
    super.init(config);
  }
```

Mitgelieferte Bibliothekselemente

Servlet Beispiel II – Forts.

```
public void doPost(HttpServletRequest req,
                   HttpServletResponse resp)
       throws ServletException,IOException
{
  resp.setContentType(„text/html“);
  PrintWriter = resp.getWriter();
  out.println(„<html><body>“);
  Enumeration parameters =
                reg.getParamerNames();
  String param = null;
  String value = null;
```

Mitgelieferte Bibliothekselemente

Servlet Beispiel II – Forts.

```
  while (parameters.hasMoreElements())
  {
    param = (String)
             parameters.nextElement();
    out.println(param+ „:“ +
              req.getParameter(param));
  }
  out.println(„</body></html>“);
  out.close();
}
```

Mitgelieferte Bibliothekselemente

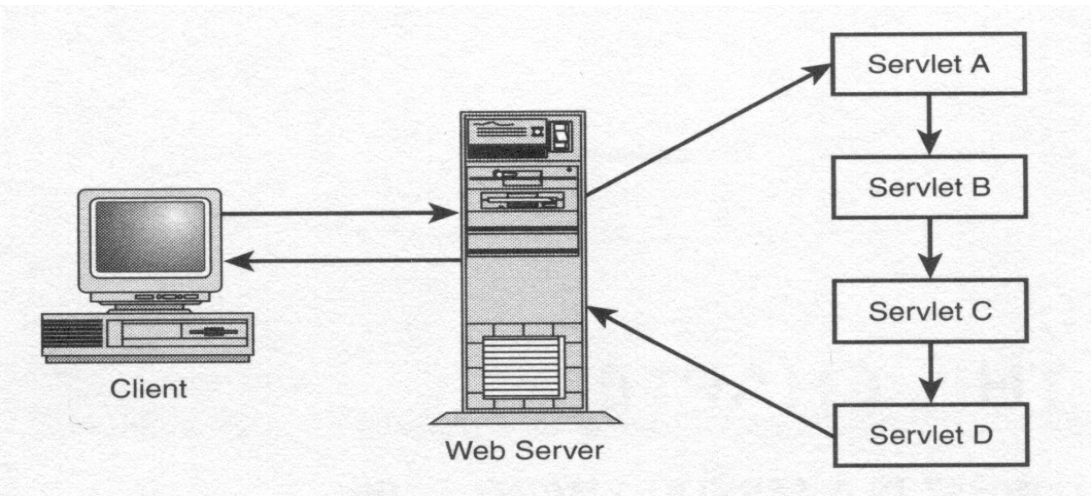
Interaktion

- Servlet Chaining
- HTTP Tunneling
- RMI (Remote Method Invocation)
- JDBC (Java Database Connectivity)

Mitgelieferte Bibliothekselemente

Servlet Chaining

- Mehrere Servlets arbeiten an der Bedienung eines Requests zusammen
- Ähnlichkeit zum „piping“ unter UNIX
  - Die Ausgabe des ersten Servlets wird von einem zweiten Servlet weiterverarbeitet



Mitgelieferte Bibliothekselemente

Servlet Chaining

- Anwendungsbeispiele:
  - Übersetzung von Texten auf internationalen Web-Sites
  - Filterfunktion für Datenbankabfragen nach einem spzifischen Benutzerprofil
  - Herausfiltern von angepassten HTML-Tags und Einsetzen des gewünschten Textes, z.B. Firmenname
  - Contentfitting: Servlet konvertiert Pages zu einheitlichem Standard, z.B. frame-based nach table-based

Mitgelieferte Bibliothekselemente

HTTP Tunneling

- Lesen und Schreiben serialisierter Objekte mittels einer HTTP-Verbindung
- Unterprotokoll innerhalb des HTTP-Protokolls (tunneling)
- Objekte können jeweils vom Client/Server gelesen/geschrieben werden

Mitgelieferte Bibliothekselemente

HTTP Tunneling

- Vorteile:
  - Wegen des „serializeable interface“ einfach zu nutzen
  - Schnell für einfache Objekte
  - Programmierer muss nicht mit Kommunikationslayern arbeiten
  - Tunneling funktioniert auch durch Firewalls

Mitgelieferte Bibliothekselemente

HTTP Tunneling

- Nachteile:
  - Alle Objekte müssen das „serializeable interface“ implementiert haben
  - Das interagierende Servlet muss sich auf dem selben Server befinden wie das Applet
  - Da Applet als Client genutzt wird müssen kompatible Browser genutzt werden
  - Langsam bei grösseren Objekten

Mitgelieferte Bibliothekselemente

RMI

- API für verteilte Anwendungen in reinen Java Implementationen
- Erlaubt es einem Client und einem Server durch Objekte über ein Netzwerk zu kommunizieren
  - Client führt Methoden eines Proxy Objekts aus, welches dann die selben Methoden auf dem Server ausführtSicht des Clients: gleiche Java VM
- Sicht des Clients: gleiche Java VM

Mitgelieferte Bibliothekselemente

RMI

- Vorteil:
  - Applets sind nicht mehr nur auf den einen Server beschränkt, auf dem sie laufen

Mitgelieferte Bibliothekselemente

JDBC

- Kommunikation zwischen einem Servlet und einer Rel. Datenbank
- Servlet kann mit einer DB auf dem selben Server oder auf einem anderen kommunizieren
  - Vgl. Applets: Kommunikation nur mit dem Server, von welchem das Applet geladen wurde

Mitgelieferte Bibliothekselemente

Resümee

- Vorteile
- Nachteile
- Zukunftsausblick

Mitgelieferte Bibliothekselemente

Vorteile

- Clientunabhängig
- Plattformunabhängig
- Gesicherte Umgebung
- Nur eine Programmiersprache (Java): keine weitere Skriptsprache nötig

Mitgelieferte Bibliothekselemente

Nachteile

- (Noch) Keine grosse Verbreitung

Mitgelieferte Bibliothekselemente

Zukunftsausblick

- CGI Scripts noch stark vertreten

Mitgelieferte Bibliothekselemente

Noch Fragen?

- Servlet Tutorial
  - http://java.sun.com/docs/books/tutorial/servlets/
- Sun Java Site
  - http://java.sun.com/products/servlet/