Java Servlet Technology

Seminar Medientechnik

Christina Eicher

30. Juni 2003

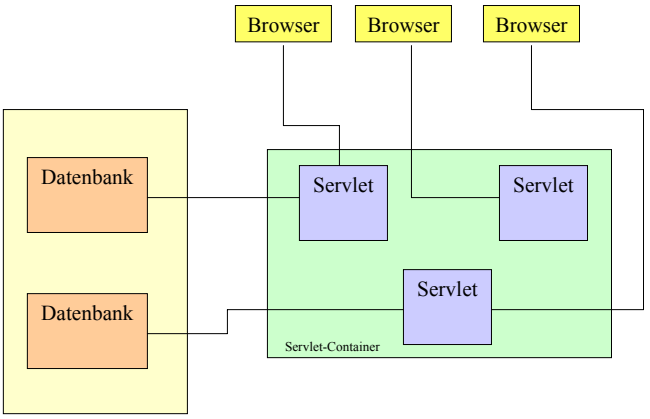
Übersicht:

1. Was ist ein Servlet?
2. Cookies und Sessions
3. Die Servlet-Klassen und das Servlet-Interface
4. Der Servlet-Container
5. Wichtige Interfaces
6. Der Lebenszyklus eines Servlets
7. Filter
8. Multithreading

1. Was ist ein Servlet:

- Webkomponente, die dynamische Inhalte generiert
- kleine unabhängige Java-Klasse
- kommuniziert mit einem Webclient über Anfrage- und Antwort-Objekte
- protokollabhängig oder protokollunabhängig

Beispiel einer Servlet-Konstellation



ServletRequest – die Anfrage des Clients

- Möglichkeit, Informationen von einem Client oder über diesen zu bekommen
- wird vom Servlet-Container erzeugt

Mögliche Informationen:

- Parameter
- Attribute
- InputStream/ InputReader
- protokollspezifische Daten

InputStream und InputReader

- Methoden: getInputStream, getReader
- es kann immer nur einer verwendet werden
- InputReader ist besser für Text
- InputStream ist besser für binäre Daten

Parameter

Methoden:

- getParameterNames(): liefert Auflistung mit allen Parameternamen

- getParameterValues(String name): liefert String[] mit allen zugehörigen Werten

- getParameter(String name): liefert den ersten Wert des Arrays

Unterschied bei POST- und GET-Anfragen:

- POST-Anfrage: im Kopf der Anfrage

- GET-Anfrage: im URI

Bsp.:
http://localhost:8080/seminar/datenservlet? name=Schuster&vorname=Max&strasse=Goethestrasse

Attribute:

- Objekte, die mit einer Anfrage verbunden sind

- zur Übertragung von Informationen, die nicht im API beschrieben sind

- zum Informationsaustausch zwischen Servlets über den Dispatcher

Reservierte Namen:

- java.*

- javax.*

- com.sun.*

ServletResponse – die Antwort an den Client

- sendet Informationen vom Server zum Client

OutputStream und Writer

- Methoden: getOutputstream() bzw. getWriter()
- Festlegen der Art des Inhalts: setContextType(String)
- Erfragen der verwendeten Zeichensatzkodierung: getCharacterEncoding()
- Festlegen von länderspezifischen Attributen; setLocale(Locale)

Beispiele für Inhalts-Arten:

text/plain	Normaler Text
image/jpeg	Jpeg-Bild
application/pdf	PDF-Dokument
text/xml	XML-Dokument
application/x-zip-compressed	ZIP-File

Puffer:

Methoden:

- getBufferSize()/setBufferSize(int)
- isCommited()
- reset()
- flushBuffer()

HttpServletResponse - Header:

- Methoden: addHeader(String name, String wert), setHeader(String name, String wert), containsHeader(String name)
- speziellere Methoden: z.B. setIntHeader(String name, String wert), addIntHeader(String name, String wert)
- Header immer vor Abschluss der Übertragung bearbeiten, sonst wird's vom Servlet-Container ignoriert

2. Cookies und Sessions

Session-Cookies

- Ansammlung von Informationen, die von einem Servlet versendet wird
- besteht aus einem Namen, einem Wert und Attribute
- Dienen zur Identifikation und Registrierung eines Clients
- werden bei Beenden des Browsers gelöscht
- Browser sollen eine Mindestzahl an Cookies unterstützen

Nachteile:

- Unsicherheit
- begrenzte Datenmenge
- Daten werden bei jeder Antwort komplett übertragen

Sessions

Vorteile:

- Sessions können von mehreren Fenstern eines Browsers benutzt werden

- Informationen bleiben auf dem Server und werden durch einen kleinen Verweis beim Client gespeichert

2 Möglichkeiten:

- mit Cookies

- mit URL-Rewriting

Sessions mit Cookies:

- Name des Cookies: JSESSIONID

- Der Cookie mit der SessionID wird bei jeder Anfrage des Browser gesendet

- Session ist so lange „neu“, bis der Client sie mit einer Anfrage zurücksendet

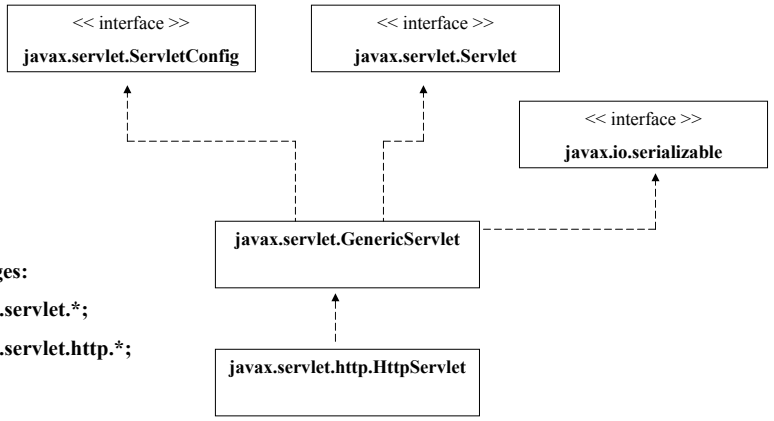
- Es kann u.U. zu Problemen mit Threads kommen

Session mit URL-Rewriting

- Cookies werden nicht von allen Browsern unterstützt
- Anhängen der Session-ID an den URL
- Vorteil: Garantie, dass die Session-Verwaltung Browser-unabhängig funktioniert


```
package seminar;

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

public class FormularServlet extends HttpServlet{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException{

        PrintWriter out = res.getWriter();
        HttpSession session = req.getSession(true);

        if(session.isNew()){
            // html-Code für das Eingabe-Formular ausgeben
        }
        else{
            UserDaten userdaten = (UserDaten)session.getAttribute("userdaten");

            if (userdaten == null){
                try {
                    if( /*  name==null || ...*/) throw new LeeresFeldException();
                    // html-Code für den Bewertungs-Bogen...
                } catch (LeeresFeldException e) {
                    // html-Code für ein neues Fromular mit Fehlermeldung...
                }
            }
            else{
                //nun muss der ausgefüllte Bewertungsbogen verarbeitet und gespeichert werden
            }
        }
    } //Ende doGet
}
```

3. Die Servlet-Klassen und das Servlet-Interface



Packages:

- **javax.servlet.*;**
- **javax.servlet.http.*;**

Die Service-Methode

service(ServletRequest, ServletResponse)

- „Request-Handling-Methode“
- nimmt alle Anfragen entgegen, die vom Servlet-Container an ein bestimmtes Servlet gestellt werden
- leitet die Anfrage an die dafür zuständige Methode weiter (z.B. doGet)

Wie erstellt man ein Servlet?

- Klasse schreiben und compilieren
- Servlet in den Servlet-Container einbinden
- Server starten
- Servlet im Browserfenster aufrufen und anschauen

4. Der Servlet-Container

(z.B. Jakarta Tomcat)

- stellt die Netzwerk-Services für die Anfragen und Antworten der Clients bereit

Einbinden von Servlets in den Container:

- Deployment-Descriptor
- Datei: web.xml

Beispiel für einen Deployment-Descriptor:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
    "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
<web-app>
    <servlet>
        <servlet-name>myServlet</servlet-name>
        <servlet-class>seminar.MyServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>myServlet</servlet-name>
        <url-pattern>/myservlet</url-pattern>
    </servlet-mapping>
</web-app>
```


5. Wichtige Interfaces

Das Interface ServletConfig

- Eigenschaften und Einstellungen eines Servlets müssen manchmal abänderbar sein

- Initialization-Parameter in der Konfigurations-Datei des Servlet-Containers

Methoden:

- getInitParameter()

- getInitParameterNames()

- getServletContext()

- getServletName()

Das Interface ServletContext

- jede Webapplikation, die im Servlet-Container läuft, hat ein ServletContext-Objekt

- der Kontext entsprich in etwa der Laufzeitumgebung eines Servlets

- er dient v.a. dem Zugriff auf externe Ressourcen und zum Abrufen von Informationen über die Umgebung, in der das Servlet läuft

- Zugriff auf statische Ressourcen, Datenquellen,die durch path spezifiziert sind

- Methoden: z.B. getMimeType(String file)

- protokollieren von Exceptions im Log-File

- kann von mehreren Servlets benutzt werden

Das Interface RequestDispatcher

- über Kontext erreichbar

- kann Anfragen an andere Komponenten weiterleiten

Methoden:

- void forward(ServletRequest, ServletResponse): Weiterleiten der Anfrage an die Ressource

- void include(ServletRequest, ServletResponse): Einfügen des von der Ressource gelieferten Ergebnisses

Zu Beachten: Forward muss vor einer Datenübertragung an den Browser passieren, weil sonst die Daten im Puffer gelöscht werden

6. Der Lebenszyklus eines Servlets

Instanziierung und Initialisierung:

- Instanziierung durch den Servlet-Container

- Initialisierung mittels der init-Methode

- es können von einem Servlet mehrere Instanzen gleichzeitig im Container sein

Beantwortung von Anfragen:

- service-Methode

Freigabe des Servlets:

- Der Servlet-Container entscheidet, wann er ein Servlet freigibt

- destroy-Methode

7. Filter

- können als eigenständige Bestandteile einer Webanwendung einem Servlet vorgeschaltet werden

- bieten die Möglichkeit, die Anfrage des Browsers und die Antwort des Servlets zu verändern

- es können mehrere Filter hintereinandergeschaltet werden

- ein Filter kann auch eine Anfrage abfangen und selbst beantworten, ohne sie weiterzuleiten

- Filter müssen das Interface Filter implementieren

- Zuordnen eines Filters: Deployment-Descriptor

z.B.:

```
<filter>
    <filter-name>myFilter</filter-name>
    <filter-class>seminar.MyFilter</filter-class>
</filter>
```

8. Multithreading

- Servlets sind besonders für stark belastete Server geeignet, weil sie sehr schnell sind
- Dies wird u.a. durch Multithreading erreicht
- dadurch kann es evtl. zu Synchronisierungs-Problemen kommen
- Lösung: synchronized

Quellenangaben:

JavaServer Pages und Servlets, Brantner/Schmidt/Wabnitz, DataBecker 1. Auflage 2001

Informative Web-Seiten:

http://java.sun.com/products/servlet/

http://www-i3.informatik.rwth-aachen.de/teaching/ 02/proseminar/prosem-servlets.pdf

http://www.joller-voss.ch/ndkjava/notes/servlets/Servlets.pdf

http://www.ifi.unizh.ch/~riedl/lectures/Zusammenfassung_serv.PDF

http://www.inf.fh-dortmund.de/personen/professoren/achilles/VorlesungFrueher/seminar2001/Seminar/Thoene/servlet.html

http://www.pms.informatik.uni-muenchen.de/lehre/seminar/client-server/01ss/Vortraege/hscs/ausarbeitung/index.html