

Internet-Datenbanken

Grundlagen des WWW

- HTML
- HTTP

Web-Anbindung von Datenbanken

- Servlets
- JSP
- JDBC

XML

- Datenmodell
- Schemabeschreibungssprachen
- Anfragesprachen

Literatur: Kapitel 18 aus ...

A. Kemper/A. Eickler

Datenbanksysteme

Eine Einführung

4., überarbeitete und erweiterte Auflage

Dieses Buch vermittelt eine systematische und umfassende Einführung in moderne Datenbanksysteme. Der Schwerpunkt der Darstellung liegt auf der derzeit marktbeherrschenden relationalen Datenbanktechnologie.

Folgende Aspekte werden ausführlich dargestellt: Datenmodellierung, Datenbankentwurf, SQL, Speichertechnologie, Anfrageoptimierung sowie die Transaktionsverwaltung mit Fehlerbehandlung und Mehrbenutzersynchronisation. Darauf aufbauend stellt das Buch das objektorientierte Datenmodell, die objektrelationalen Modellierungskonzepte und das deduktive Datenmodell vor. Einen neuen Schwerpunkt der nunmehr 4. Auflage bilden die fortschrittlichen Anwendungen von Datenbanken im Internet sowie im betriebswirtschaftlichen Data Warehouse für Decision Support-Anfragen und das Data Mining. Die Anbindung von Datenbanken an das Internet über die Java-Schnittstellen sowie die XML-Datenmodellierung wurden neu ausgearbeitet.

Alle Konzepte werden an einer durchgehenden Beispielanwendung veranschaulicht. Jedes Kapitel enthält zahlreiche Übungsaufgaben. So eignet sich das Buch einerseits für Studenten als Lehrbuch und andererseits für Anwender als Referenz bzw. Anleitung zum Selbststudium.

www.olderbourg-verlag.de

ISBN 3-486-25706-4



Datenbanksysteme

Kemper / Eickler

A. Kemper/A. Eickler

Datenbank- systeme

Eine Einführung

4. Auflage

Oldenbourg



Oldenbourg



HTML-Grundlagen

<HTML>

<HEAD> <TITLE>Gesamtes Vorlesungs-Verzeichnis</TITLE> </HEAD>

<BODY>

<H1>Die Professoren der Universität</H1>

** Prof. Augustinus**

** 5022: Glaube und Wissen (mit 2SWS)**

** Prof. Curie**

** Prof. Kant**

** 5001: Grundzüge (mit 4SWS)**

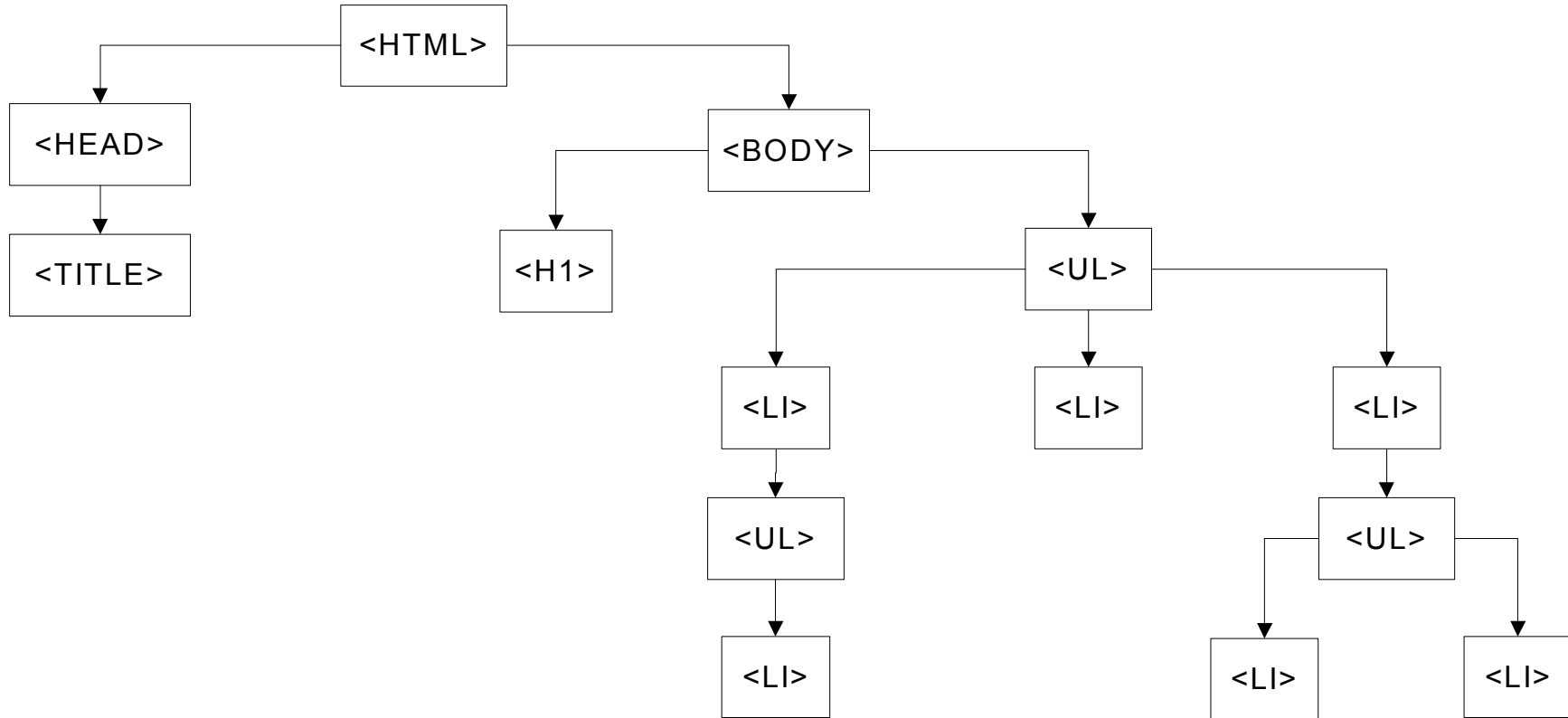
** 4630: Die 3 Kritiken (mit 4SWS)**

...

</BODY>

</HTML>

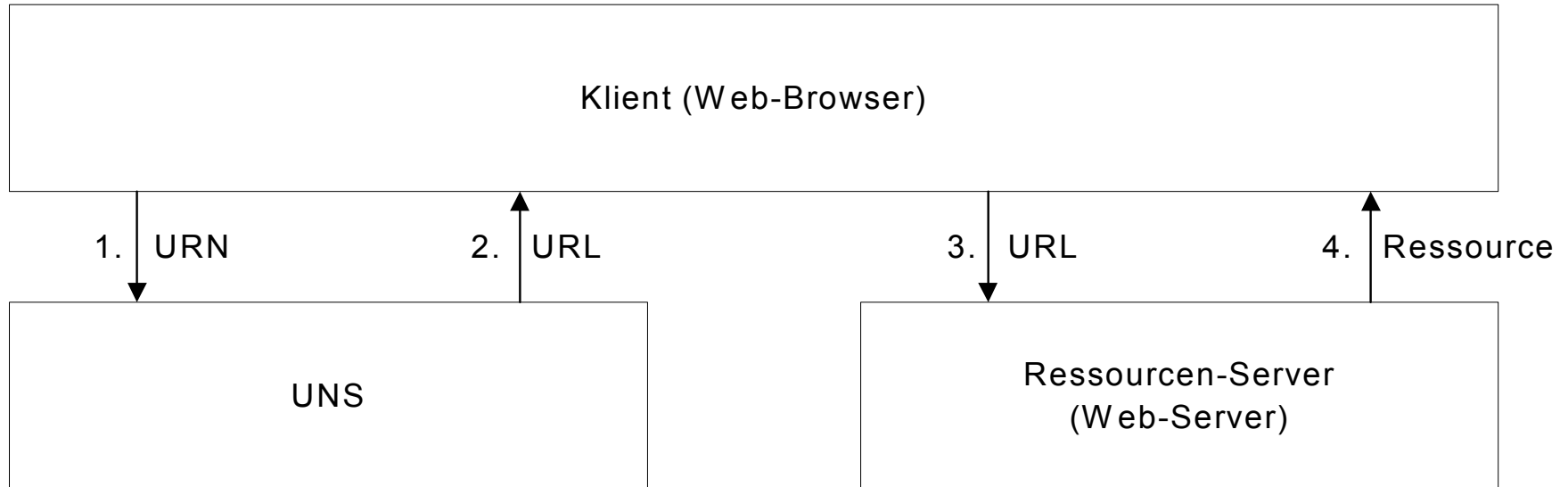
Illustration der hierarchischen Struktur



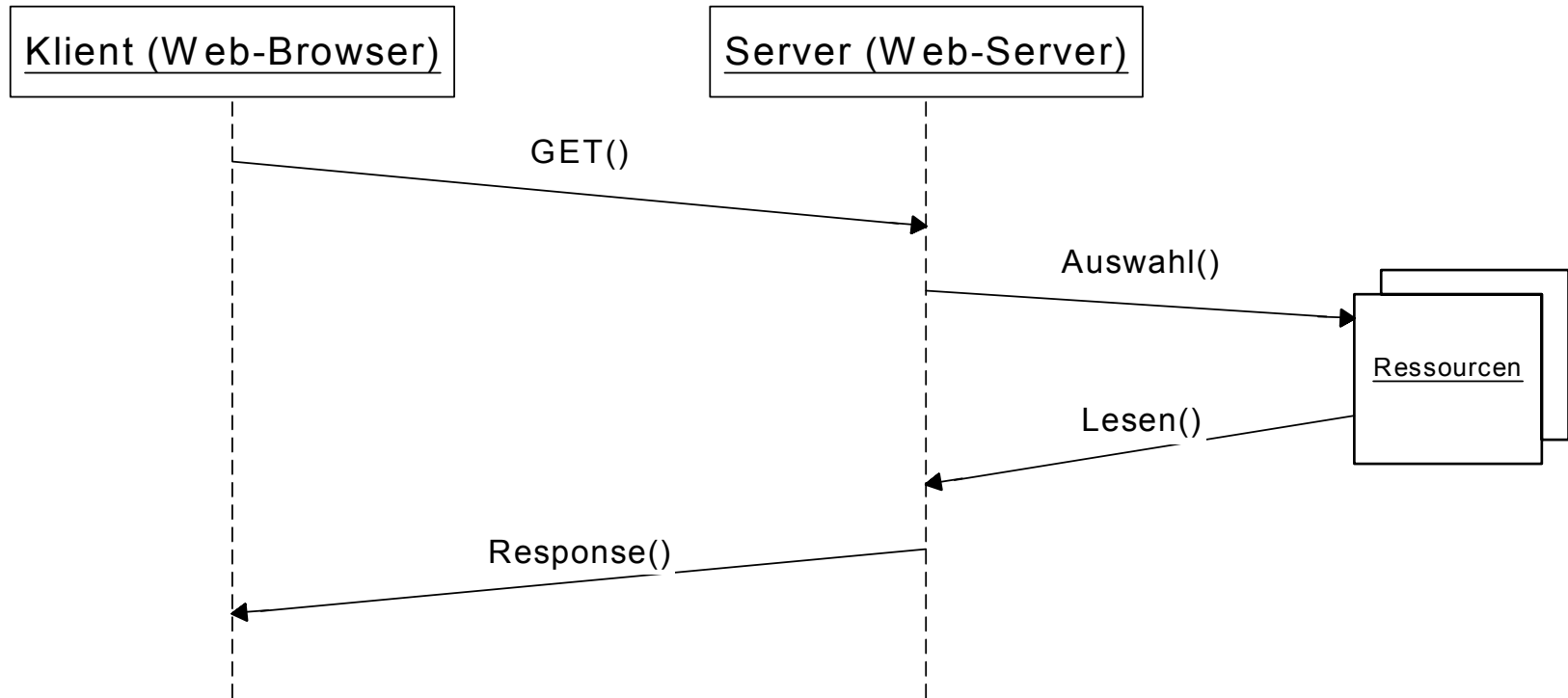
Adressierung im Web

- URLs: Uniform Resource Locators
 - <http://www.db.fmi.uni-passau.de/publications/books/DBMSeinf/index.phtml>
- Entspricht physischen Objektidentifikatoren
 - Speicherort ist in den Identifikator „ein-codiert“
 - Deshalb gibt es so viele Verletzungen der referentiellen Integrität im WWW
 - 404-Fehler (busted link)
- URNs: Uniform Resource Names
 - Logische Identifikatoren
 - Unabhängig vom Speicherort des Dokuments
 - URN Name Server nötig für die Umsetzung auf Adressen
- URI: URLs \cup URNs

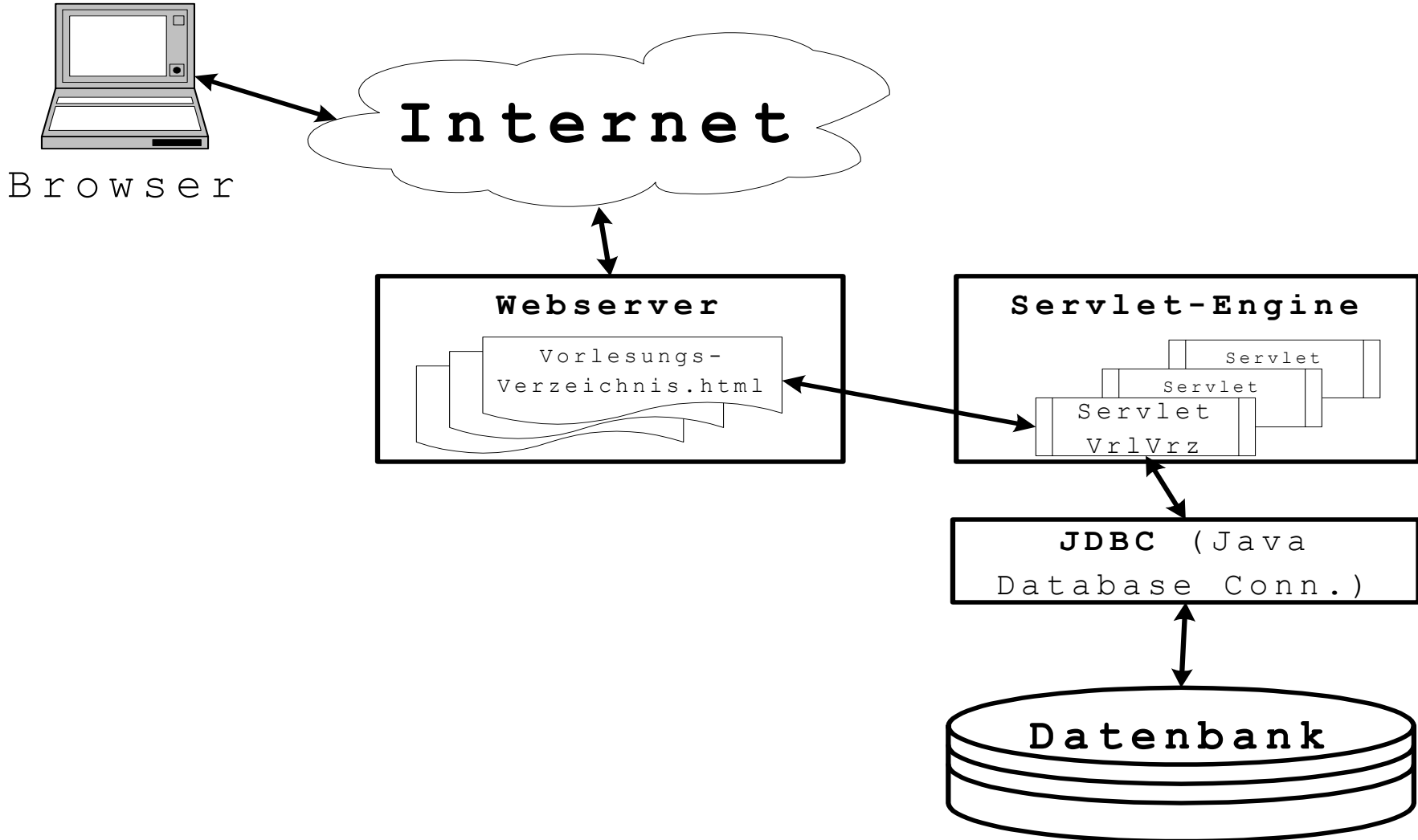
URN-Transformation



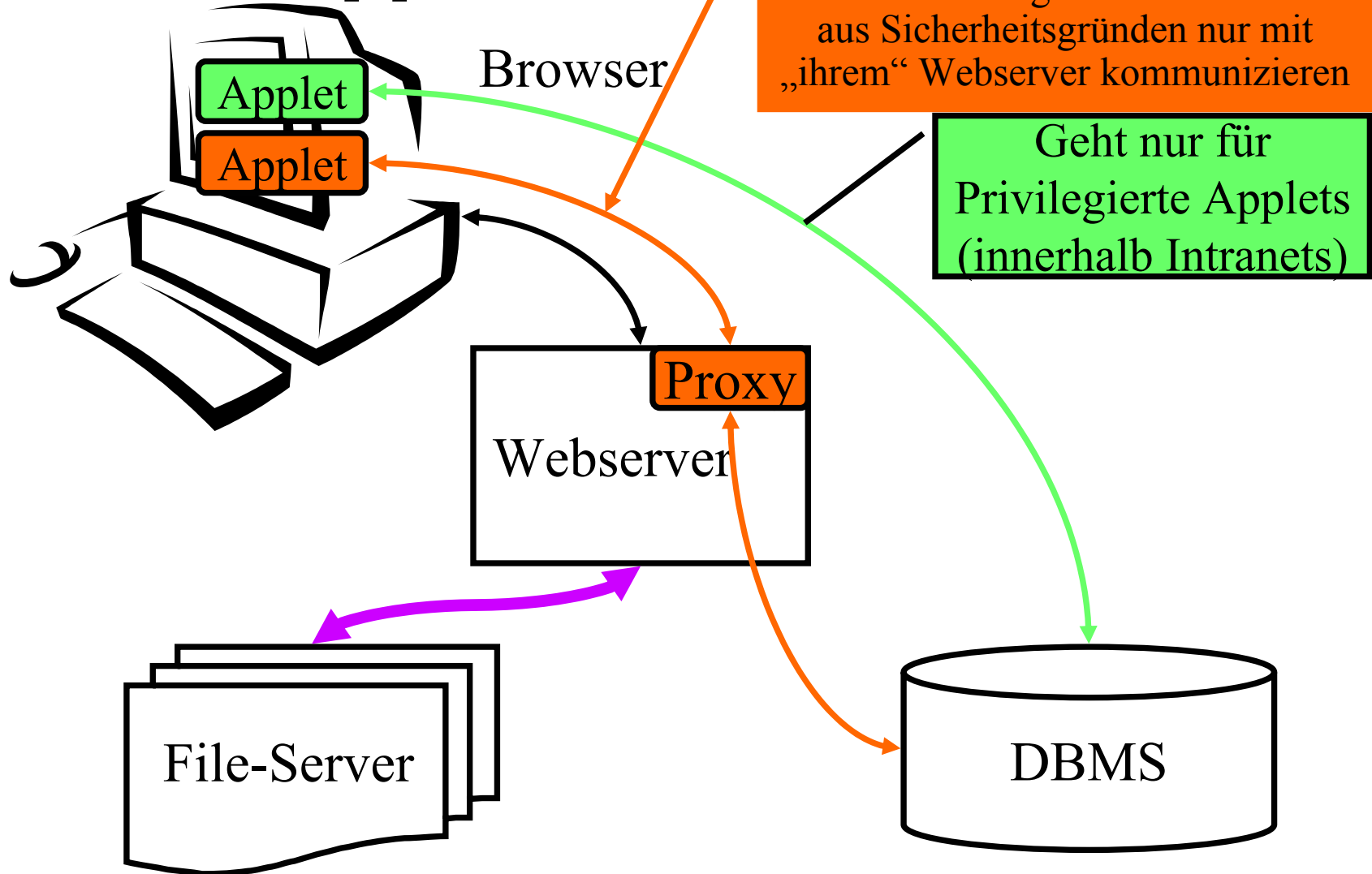
Client/Server-Architektur des WWW



Web-Anbindung von Datenbanken via Servlets



Architektur der DB-Anbindung mittels Applet



Vorlesungsverzeichnis.html

<HTML>

<HEAD> <TITLE>Vorlesungs-Verzeichnis mittels Servlet</TITLE> </HEAD>

<BODY>

<CENTER>

**<FORM ACTION="http://www.db.fmi.uni-passau.de/servlets-buch/VrIVrz"
METHOD="GET">**

Bitte geben Sie den Namen einer Professorin

**bzw. eines Professors ein:
**

**<INPUT TYPE=TEXT NAME="professor_name"></INPUT>
**

<INPUT TYPE=SUBMIT VALUE="Abfrage starten"></INPUT>

</FORM>

</CENTER>

</BODY>

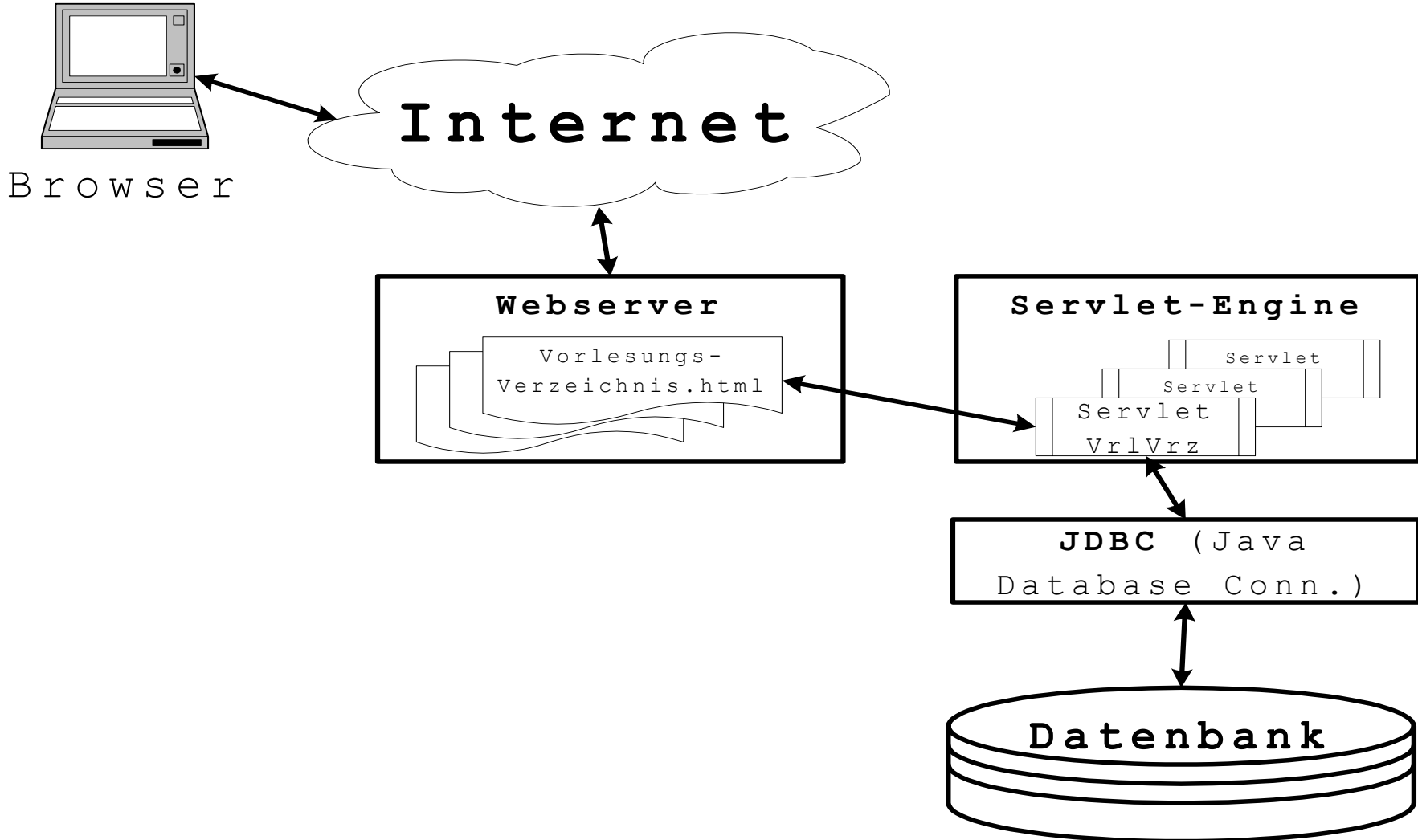
</HTML>

Browser-Darstellung

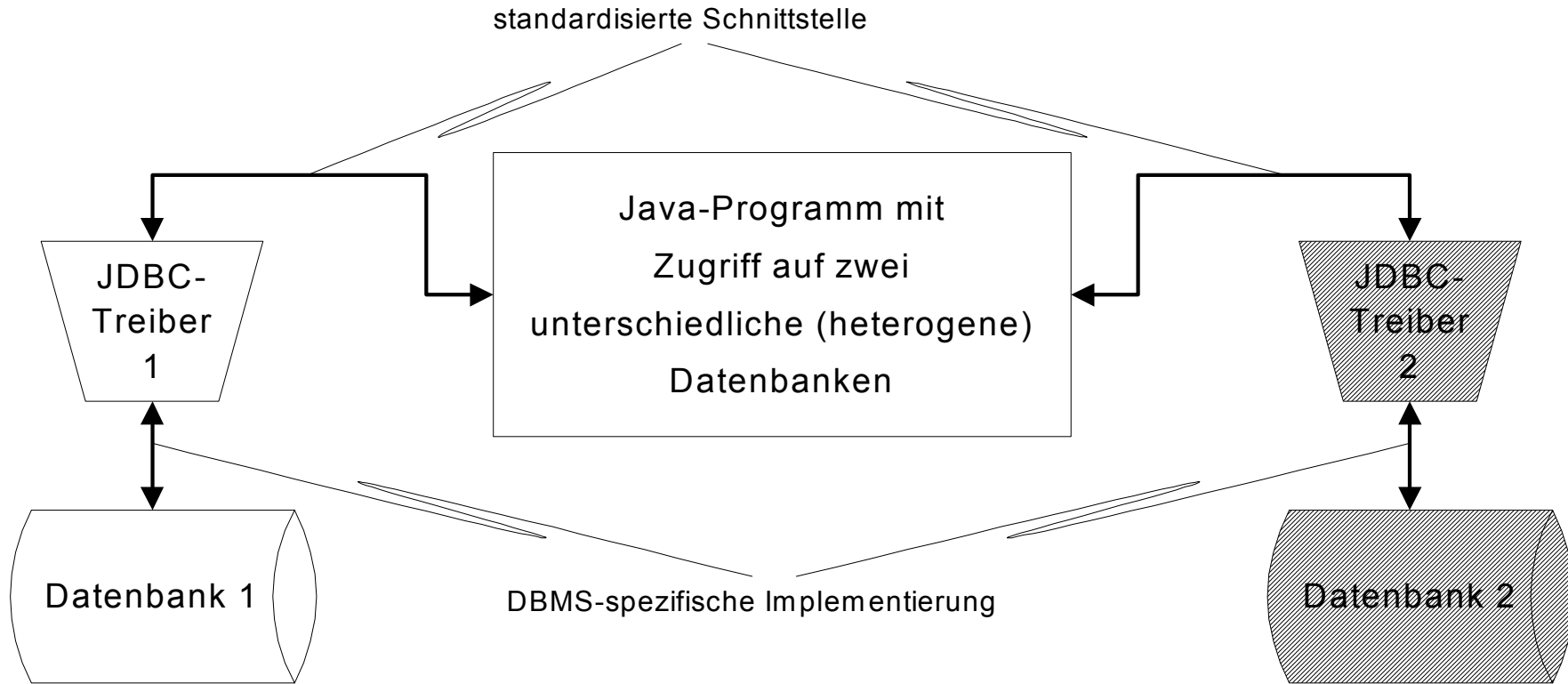


- Die generierte URL enthält einen sogenannten **Query String**
- http://www.db.fmi.uni-passau.de/servlets-buch/VrIVrz?professor_name=Sokrates
- Wenn mehrere Parameter übergeben werden müssen, werden sie mit & getrennt hintereinandergefügt

Web-Anbindung von Datenbanken via Servlets



Anbindung einer Datenbank via JDBC an das Servlet



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

public class VrlVrz extends HttpServlet {
    public void doGet (HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        Connection conn = null; Statement stmt = null;
        ResultSet rs = null;
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            conn = DriverManager.getConnection(
                "jdbc:oracle:oci8:@lsintern-db", "nobody", "Passwort");
            stmt = conn.createStatement();
            String query = "select v.VorlNr, v.Titel, v.SWS " +
                "from Vorlesungen v, Professoren p " +
                "where v.gelesenVon = p.PersNr and p.Name = '" +
                    request.getParameter("professor_name") + "'";
            rs = stmt.executeQuery(query);
            out.println("<HTML>");
            out.println("<HEAD> <TITLE> Vorlesungen von Prof. " +
                request.getParameter("professor_name") + "</TITLE></HEAD>")
        }
    }
}
```

```
out.println("<BODY>");
out.println("<H1> Vorlesungen von Prof. " +
    request.getParameter("professor_name") + ": </H1>");
out.println("<UL>");
while (rs.next())
    out.println("<LI>" + rs.getInt("VorlNr") +
        ": " + rs.getString("Titel") +
        " (mit " + rs.getInt("SWS") + " SWS)");
out.println("</UL>"); out.println("</Body> </HTML>");
}
catch(ClassNotFoundException e) {
    out.println("Datenbanktreiber nicht gefunden:" + e.getMessage())
}
catch (SQLException e) {
    out.println("SQLException: " + e.getMessage());
}
finally {
    try {
        if (conn != null) conn.close();
    } catch (SQLException ignorieren) {}
}
}
}
```

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

```
public class VrlVrz extends HttpServlet {
    public void doGet (HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException {
        Connection conn = null; Statement stmt = null;
        ResultSet rs = null;
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            conn = DriverManager.getConnection(
                "jdbc:oracle:oci8:@lsintern-db", "nobody", "Passwort");
            stmt = conn.createStatement();
```

JDBC-Datenbank-
Verbindungsaufbau



String query =

```
"select v.VorlNr, v.Titel, v.SWS " +  
"from Vorlesungen v, Professoren p " +  
"where v.gelesenVon = p.PersNr and p.Name = '" +  
    request.getParameter("professor_name") + "'";
```

```
rs = stmt.executeQuery(query);
```

```
out.println("<HTML>");
```

```
out.println("<HEAD> <TITLE> Vorlesungen von Prof. " +  
    request.getParameter("professor_name") +  
    "</TITLE></HEAD>");
```

```
out.println("<BODY>");
```

```
out.println("<H1> Vorlesungen von Prof. " +  
    request.getParameter("professor_name") + ": </H1>");
```

```
out.println("<UL>");
```

**Generierung der HTML-
Seite aus dem Anfrageergebnis**

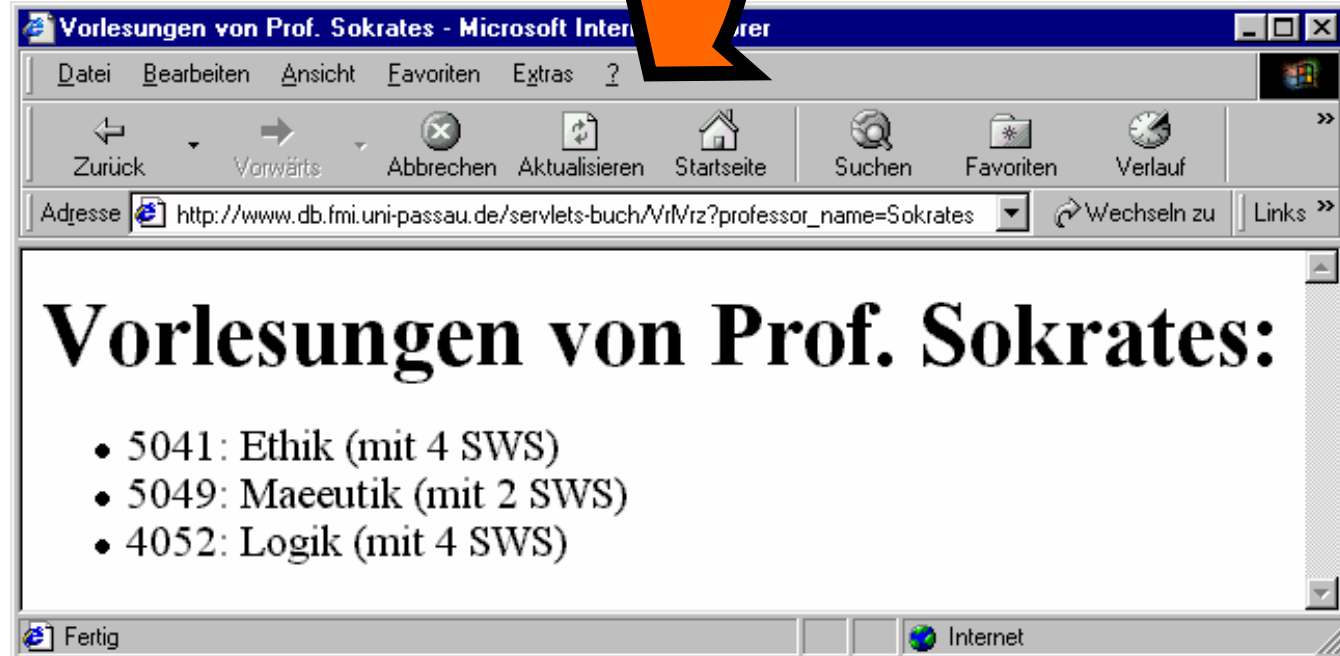
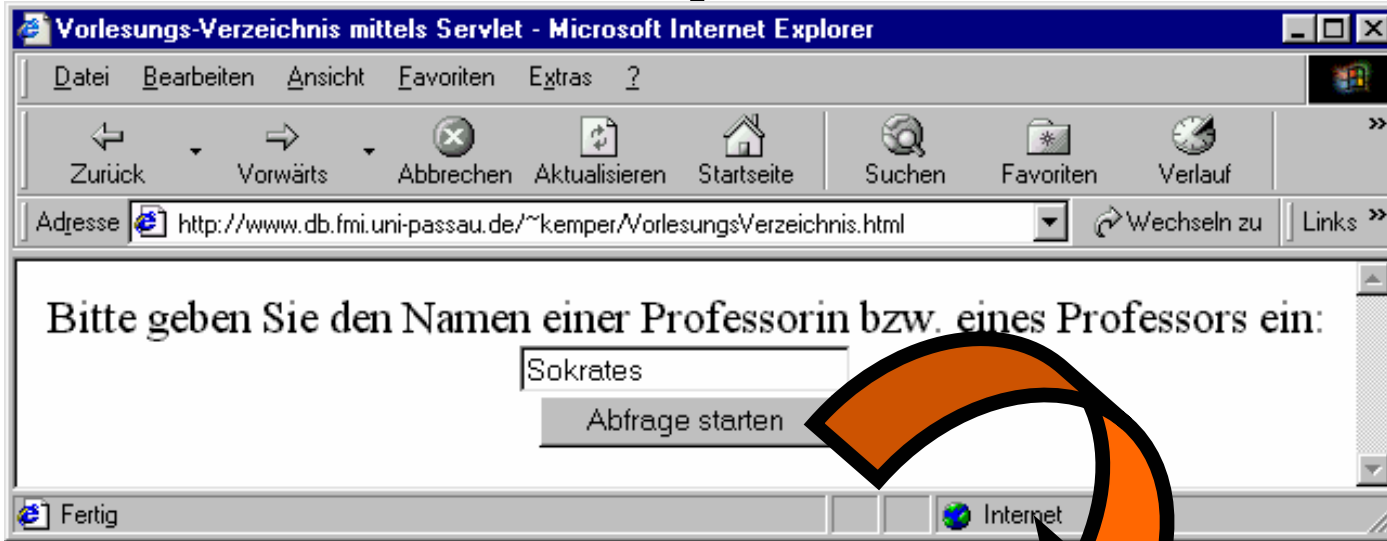


```
while (rs.next())
    out.println("<LI>" + rs.getInt("VorlNr") +
        ": " + rs.getString("Titel") +
        " (mit " + rs.getInt("SWS") + " SWS)");
out.println("</UL>"); out.println("</Body> </HTML>");
}
catch(ClassNotFoundException e) {
    out.println("Datenbanktreiber nicht gefunden:" +
        e.getMessage());
}
catch (SQLException e) {
    out.println("SQLException: " + e.getMessage());
}
finally {
    try {
        if (conn != null) conn.close();
    } catch (SQLException ignorieren) {}
}
}
```



Ausnahme-
Behandlung

Antwortseite (Browser-Darstellung)



Verwendung von POST statt GET

<FORM

ACTION="http://www.db.fmi.uni-passau.de/servlets-buch/VrlVrz"

METHOD="POST">

Bitte geben Sie den Namen einer Professorin

**bzw. eines Professors ein:
**

<INPUT TYPE=TEXT NAME="professor_name"> </INPUT>

**
**

<INPUT TYPE=SUBMIT VALUE="Abfrage starten"> </INPUT>

</FORM>

Verwendung von POST statt GET

- Weiterhin muß das Servlet natürlich die Methode doPost implementieren.
- Eine Möglichkeit besteht darin, daß das Servlet beide Methoden, also doGet und doPost, auf die gleiche Weise realisiert.
- Dann könnte man einfach zusätzlich folgende Methode der Klasse VrlVrz hinzufügen:

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

Wiederverwendung des Servlets

<HTML>

<HEAD> <TITLE>Gesamtes Vorlesungs-Verzeichnis mittels Servlet</TITLE>

</HEAD>

<BODY>

<H1> Die Professoren der Universität </H1>

** Prof. Augustinus**

** **

**Vorlesungen **

** Prof. Curie**

** **

**Vorlesungen **

...

** Prof. Sokrates**

** **

**Vorlesungen **

</BODY> </HTML>

Browser-Darstellung



The screenshot shows a Microsoft Internet Explorer browser window. The title bar reads "Gesamtes Vorlesungs-Verzeichnis mittels Servlet - Microsoft Internet Explorer". The address bar contains the URL "http://www.db.fmi.uni-passau.de/~kemper/GesamtVorlesungsVerzeichnis.html". The main content area displays the heading "Die Professoren der Universität" followed by a list of professors and links to their lecture pages.

Gesamtes Vorlesungs-Verzeichnis mittels Servlet - Microsoft Internet Explorer

Adresse <http://www.db.fmi.uni-passau.de/~kemper/GesamtVorlesungsVerzeichnis.html>

Die Professoren der Universität

- Prof. Augustinus
 - [Vorlesungen](#)
- Prof. Curie
 - [Vorlesungen](#)
- Prof. Kant
 - [Vorlesungen](#)
- Prof. Kopernikus
 - [Vorlesungen](#)
- Prof. Popper
 - [Vorlesungen](#)
- Prof. Russel
 - [Vorlesungen](#)
- Prof. Sokrates
 - [Vorlesungen](#)

Fertig Internet

Wenn man auf Vorlesungen von Kant geklickt hat ...



Diskussion der Servlets

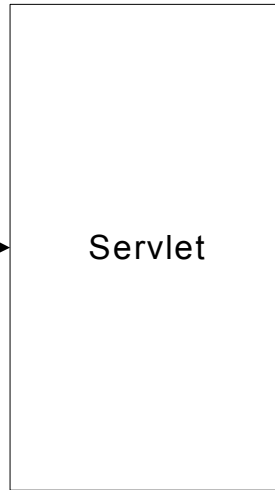
Java Server Pages (JSPs)

Active Server Pages (ASPs)

JSP-Seite (***.jsp)

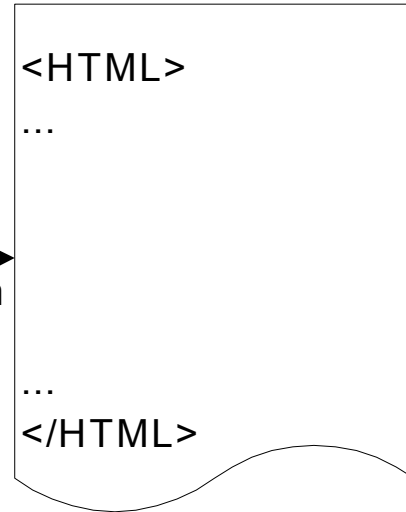


Übersetzung
erfolgt bei jeder
Änderung der
JSP-Seite



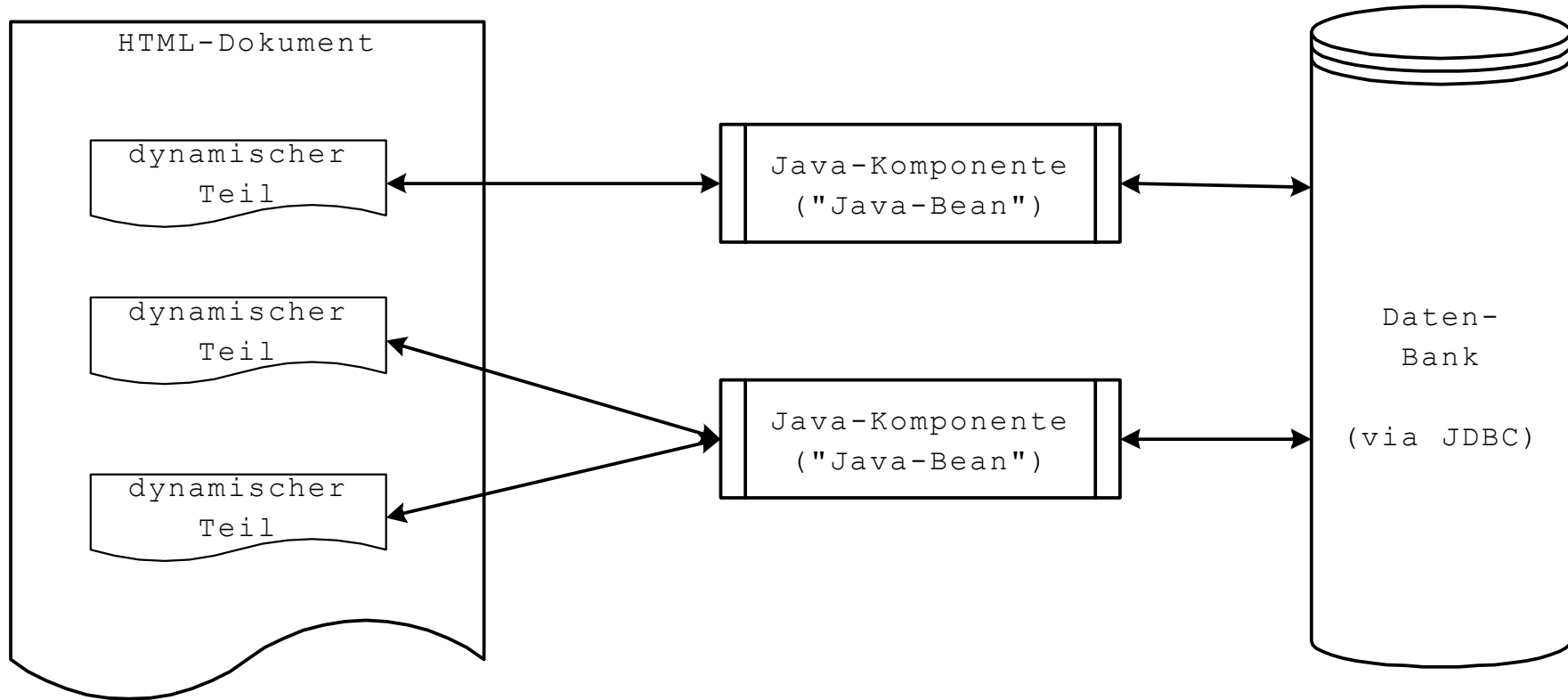
Ausführung
erfolgt bei jedem
Zugriff auf die
Seite

"reine" HTML-Seite



Java-Beans zur Generierung der dynamischen Inhalte

Java Server Page



JSP-Seite

```
<%@ page import="jspdemo.VorlesungenBean" %>
```

```
<jsp:useBean id="mybean" class="jspdemo.VorlesungenBean"  
             scope="application" />
```

```
<HTML> <HEAD>
```

```
  <TITLE>Gesamtes Vorlesungs-Verzeichnis mittels Java Server Page </TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
  <H1>Die Professoren der Universität</H1>
```

```
  <UL>
```

```
    <LI> Prof. Augustinus
```

```
      <%= mybean.generiereVorlListe("Augustinus") %>
```

```
    <LI> Prof. Curie
```

```
      <%= mybean.generiereVorlListe("Curie") %>
```

```
  </UL>
```

```
</BODY>
```

```
</HTML>
```

JSP-Tags

- `<% @page Attribute der Direktive >`
 - Mit der page-Direktive kann der Übersetzungsvorgang der JSP-Seiten mittels unterschiedlicher Attribute gesteuert werden.
 - Hierunter fällt auch das Attribut `import` für die Inkludierung von Java-Packages, die für die Übersetzung des in der JSP-Seite enthaltenen Java-Codes benötigt werden.
 - In unserem Beispiel wird das Package für die JDBC-Funktionalität `java.sql.*` inkludiert.
- `<%! Deklaration %>`
 - Von diesem Tag eingeschlossen kann man z.B. eine Java-Operation definieren, die dann später in der Seite (mehrfach) verwendet wird.
 - Dies ist in unserem Beispiel geschehen: Dort wurde die String-Funktion `generiereVorListe` deklariert und später mehrfach aufgerufen.

JSP-Tags

- `<%= Ausdruck %>`
 - An die Stelle des Tags wird die textuelle Ausgabe des Ausdrucks substituiert.
 - Es handelt sich hierbei um eine vereinfachende Konvention, da das obige Konstrukt äquivalent zu folgendem ist:
`<% out.print(Ausdruck) %>`.
- `<% Java-Code-Fragment %>`
 - umschließt Java-Code, der ausgeführt wird.
 - Durch Schreiben auf das implizit zur Verfügung stehende Objekt `out` kann der Code auch Ausgaben erzeugen, die in das HTML-Dokument an dieser Stelle eingefügt werden.
- `<%-- Kommentar --%>`
 - Hiermit wird ein Kommentar abgegrenzt.
 - Dieser Kommentar wird nicht in das erzeugte HTML-Dokument, das letztendlich an den Klienten (Web-Browser) ausgeliefert wird, übernommen.

Die Java VorlesungenBean

```
package jspdemo; import java.sql.*;

public class VorlesungenBean {
    Connection conn = null;
    String conn_error = null;
    public VorlesungenBean() {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            conn = DriverManager.getConnection(
                "jdbc:oracle:oci8:@lsintern-db", "nobody", "Passwort");
        }
        catch(Exception e) {
            conn_error = e.toString(); }
    }
    public String generiereVorlListe(String name) {
```



```

public String generiereVorlListe(String name) {
    Statement stmt = null;
    ResultSet rs = null;
    if (conn == null)
        return("Probleme mit Datenbank:" + conn_error + " </br>");
    StringBuffer result = new StringBuffer();
    try {
        stmt = conn.createStatement();
        String query = "select v.VorlNr, v.Titel, v.SWS " +
            "from Vorlesungen v, Professoren p " +
            "where v.gelesenVon = p.PersNr and " +
            "p.Name = '" + name + "'";
        rs = stmt.executeQuery(query);
        result.append("<UL>");
        while (rs.next())
            result.append("<LI>" + rs.getInt("VorlNr") +
                ": " + rs.getString("Titel") +
                " (mit " + rs.getInt("SWS") + " SWS)");
        result.append("</UL>");
    } catch (SQLException e) {

```


Wiederverwendung der **VorlesungenBean** für Sokrates' Home-Page

```
<%@ page import="jspdemo.VorlesungenBean" %>
<jsp:useBean id="prg" class="jspdemo.VorlesungenBean"
            scope="application"/>

<HTML>

  <HEAD> <TITLE>Sokrates' Home-Page mit JSP</TITLE> </HEAD>

  <BODY>

    <H1><IMG ALIGN=TOP ALT="Bild" SRC="Sokrates.gif">
      Prof. Sokrates</H1>

    <H1> Vorlesungen </H1>

    <%= prg.generiereVorlListe("Sokrates") %>

  </BODY>
</HTML>
```

Wiederverwendung der **VorlesungenBean**

```
<%@ page import="jspdemo.getVorlesungenBean" %>
```

```
<jsp:useBean id="mybean" class="jspdemo.VorlesungenBean"  
            scope="application" />
```

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>Gesamtes Vorlesungs-Verzeichnis mittels Java Server Page</TITLE>
```

```
</HEAD>
```

```
<BODY>
```

```
<H1>Die Professoren der Universität</H1>
```

```
<UL>
```

```
<LI> Prof. Augustinus
```

```
<%= mybean.generiereVorListe("Augustinus") %>
```

```
<LI> Prof. Curie
```

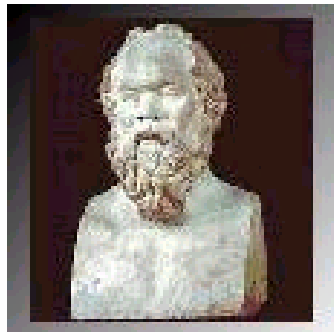
```
<%= mybean.generiereVorListe("Curie") %>
```

```
.....
```

```
</UL>
```

```
</BODY>
```

```
</HTML>
```



Prof. Sokrates

Vorlesungen

- 5041: Ethik (mit 4SWS)
- 5049: Maeutik (mit 2SWS)
- 4052: Logik (mit 4SWS)

Die Professoren der Universität

- Prof. Augustinus
 - 5022: Glaube und Wissen (mit 2SWS)
- Prof. Curie
- Prof. Kant
 - 5001: Grundzuege (mit 4SWS)
 - 4630: Die 3 Kritiken (mit 4SWS)
- Prof. Kopernikus
- Prof. Popper
 - 5259: Der Wiener Kreis (mit 2SWS)
- Prof. Russel
 - 5043: Erkenntnistheorie (mit 3SWS)
 - 5052: Wissenschaftstheorie (mit 3SWS)
 - 5216: Bioethik (mit 2SWS)
- Prof. Sokrates
 - 5041: Ethik (mit 4SWS)
 - 5049: Maeutik (mit 2SWS)
 - 4052: Logik (mit 4SWS)

Projektarbeit

- **Entwickeln Sie ein umfassendes Web-Informationssystem für unsere Universität.**
- **Achten Sie bei Ihrer Entwicklung insbesondere auch auf Sicherheit und Datenschutz.**
- **Z.B. sollte sichergestellt werden, daß Studenten nur Ihre eigenen Prüfungsergebnisse abrufen können und daß diese Information abhörsicher (also via HTTPS/SSL) übertragen wird.**