

```
DocumentCollection dc = db.FTSearch(Abfrage);
Document d = dc.getFirstDocument();
while (d != 0) {
    String Name = db.getTitle();
    out.println(Name + "<p>");
    d = dc.getNextDocument();
}
} catch (NotesException e) {
    e.printStackTrace();
}
}
```

Der große Vorteil beim Einsatz von Servlets ist, daß sie so lange geladen werden können, wie der HTTP-Server-Task läuft, was einen deutlichen Leistungsvorteil gegenüber normalen CGI-Programmen ergibt.

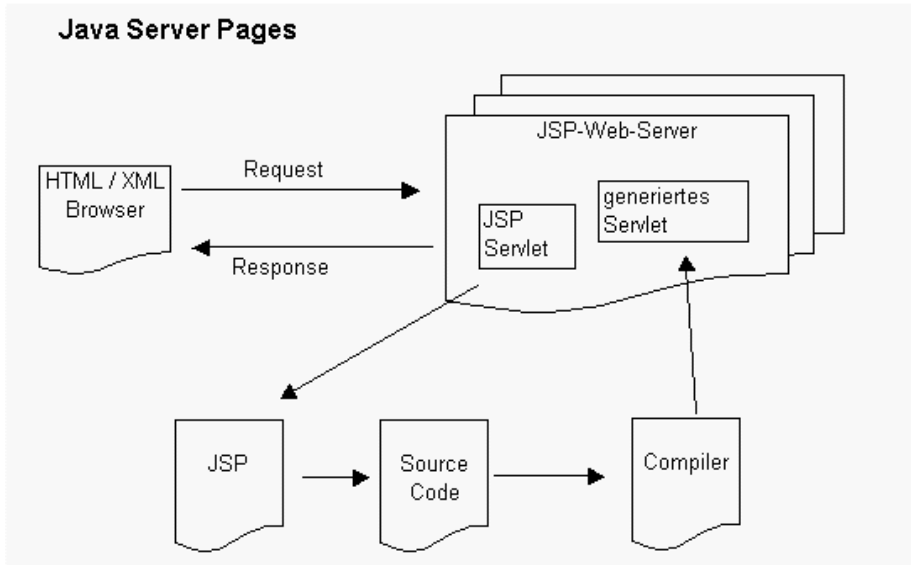
6.9 Java Server Pages

Die Technologie von Java Server Pages (JSP) ist noch recht neu. Auch hierfür benötigen Sie eine entsprechende Unterstützung. Sollten Sie nicht mit der Enterprise Edition arbeiten, benutzen Sie am besten die neueste Form des JSDK von Java. Weiterhin benötigen Sie einen lauffähigen Web-Server mit JSP-Unterstützung. Auch in diesem Fall können Sie den Java Web Server 2.0 einsetzen.

Ausgangspunkt einer Java-Server-Seite ist eine ganz normale statische HTML-Seite, die Sie mit einem beliebigen HTML-Editor erstellen.

In unserem Beispiel werden die Daten einfach als HTML-Tabelle ausgegeben. Der Datenbankzugriff wird mit der Klasse *java.sql* realisiert, die Sie schon aus Kapitel 4 kennen.

Die Abarbeitung einer Java Server Page ist ganz einfach.



Für die allgemeinen Einstellungen dient unter Java Server Pages die Direktive:

```
<% @page language = "java"
import="java.sql.*" %>
```

Mit dieser Anweisung wird das Package *java.sql* eingebunden. Die Ausgabe erfolgt über

```
<%=rs.getString("Feldbezeichnung")%>
```

Sie sehen also schon hieran, wie einfach der Aufbau einer JSP-Seite ist. Nachfolgend finden Sie die entsprechende Seite zum Auslesen und Darstellen der Tabelle »Deutschland«.

```
<HTML>
<HEAD>
<%@ page language="java" import="java.sql.*" %>
<%
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con = DriverManager.getConnection("jdbc:odbc:Warehouse");
        Statement stm = con.createStatement();
        ResultSet rs = stm.executeQuery("select * from Deutschland");
    }
%>
<TITLE>
AListe
```

6 Anwendungsentwicklung im Intranet

```
</TITLE>
</HEAD>
<BODY>
<H3><U>Datenauswertung mit JSP</U></H3>
<TABLE BORDER="1" WIDTH="80%" BGCOLOR="silver">
  <TR>
    <TD WIDTH="16%"><P ALIGN="CENTER"><B>Zeitraum</B></TD>
    <TD WIDTH="16%"><P ALIGN="CENTER"><B>Produkt 1</B></TD>
    <TD WIDTH="16%"><P ALIGN="CENTER"><B>Produkt 2</B></TD>
    <TD WIDTH="16%"><P ALIGN="CENTER"><B>Produkt 3</B></TD>
    <TD WIDTH="16%"><P ALIGN="CENTER"><B>Produkt 4</B></TD>
  </TR>
  <% rs.next(); %>
  <TR>
    <TD WIDTH="16%">&nbsp;<%= rs.getString("Zeitraum")%></TD>
    <TD WIDTH="16%">&nbsp;<%= rs.getString("Produkt1")%></TD>
    <TD WIDTH="16%">&nbsp;<%= rs.getString("Produkt2")%></TD>
    <TD WIDTH="16%">&nbsp;<%= rs.getString("Produkt3")%></TD>
    <TD WIDTH="16%">&nbsp;<%= rs.getString("Produkt4")%></TD>
  </TR>
  <% while(rs.next()) { %>
  <TR>
    <TD WIDTH="16%">&nbsp;<%= rs.getString("Zeitraum")%></TD>
    <TD WIDTH="16%">&nbsp;<%= rs.getString("Produkt1")%></TD>
    <TD WIDTH="16%">&nbsp;<%= rs.getString("Produkt2")%></TD>
    <TD WIDTH="16%">&nbsp;<%= rs.getString("Produkt3")%></TD>
    <TD WIDTH="16%">&nbsp;<%= rs.getString("Produkt4")%></TD>
  </TR>
  <% }
  con.close();
} catch(Exception ex) {
  ex.printStackTrace();
} %>
</TABLE>
<p><a HREF="DataWarehouse.html">Zurück zum Applet</a></p>
</BODY>
</HTML>
```

Beachten Sie, daß Sie die Datei mit der Kennung »*.jsp« speichern müssen.

Es dürfte Ihnen jetzt nicht schwerfallen, die Datei für die Datenbanktabelle »Frankreich« zu erstellen beziehungsweise zu modifizieren.

Sie können hierbei auch mit anderen SQL-Klauseln arbeiten. So ist es zum Beispiel über die Abfrage

```
SELECT Umsätze.Land, Umsätze.Produkt, Umsätze.Geldeinheit
FROM Umsätze
WHERE (((Umsätze.Produkt)="Produkt 4"));
```

möglich, alle Umsätze für Produkt 4 in Deutschland und Frankreich anzuzeigen. Achten Sie nur hierbei auf die Ausgabe in der Tabelle.

Im Beispielprogramm wurde die Datenbank Access verwendet. Access bietet die Möglichkeit, sogenannte Kreuztabellenabfragen zu erzeugen. Dahinter verbirgt sich eine Pivot-Abfrage zur Auswertung der Datenbank. Durch die nachfolgende SQL-Abfrage können Sie eine entsprechende Abfrage durchführen.

```
TRANSFORM Sum([Geldeinheit]) AS [Der Wert]
SELECT [Produkt], Sum([Geldeinheit]) AS [Gesamtsumme von Geldeinheit]
FROM Umsätze
GROUP BY [Produkt]
PIVOT [Land];
```

Experimentieren Sie mit den Datenbanktabellen und nutzen Sie sie in Ihren Web-Anwendungen zum schnellen Zugriff auf Java Server Pages.

Es ist zum Beispiel mit dem nachfolgenden HTML-Code möglich, dynamisch eine Auswahlbox zu füllen.

```
<HTML>
<HEAD>
<%@ page language="java" import="java.sql.*" %>
<%
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con = DriverManager.getConnection("jdbc:odbc:Warehouse");
        Statement stm = con.createStatement();
        ResultSet rs = stm.executeQuery("select Zeitraum from Deutschland");
        rs.next();
    }
%>
<TITLE>
```

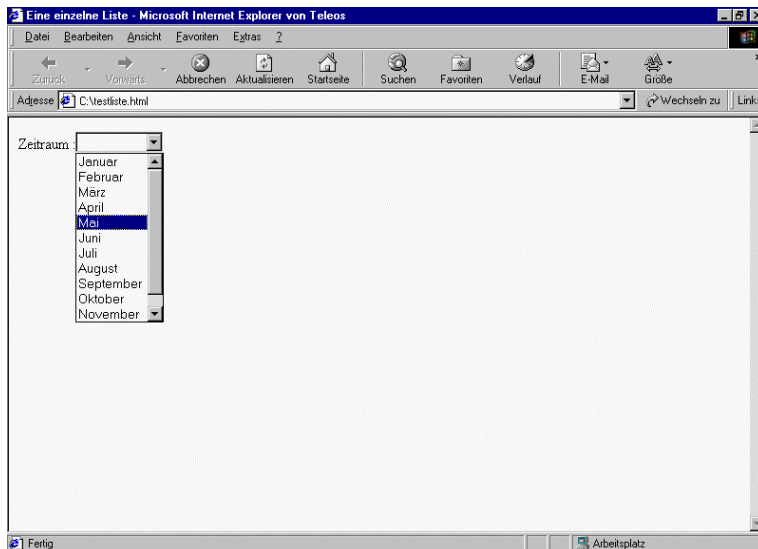
6 Anwendungsentwicklung im Intranet

Eine einzelne Liste

```
</TITLE>
</HEAD>
<BODY>

<P>Zeitraum :<SELECT NAME="Selection">
<OPTION><%= rs.getString("Zeitraum") %></OPTION>
<%
    while(rs.next()) {
%>
<OPTION><%= rs.getString("Zeitraum") %></OPTION>
<%
    }
    con.close();
} catch(Exception ex) {
ex.printStackTrace();
}
%>
<P>
</BODY>
</HTML>
```

Die JSP-Seite liefert hierbei folgendes Ergebnis.



6.9.1 Allgemeiner Datenbankzugriff

Sie können also auch mit JSP eine allgemeine Klasse für den Datenzugriff erstellen. Diese könnte folgendes Aussehen haben:

```

<%@ page import="java.sql.*"%>
<Html>
<Head>
<Title>Verkaufsanalyse mit JSP</Titel>
</Head>

<Body>

    Auswertung der Verkaufsanalyse

    <Table Border="0" Cellpadding="2" Cellspacing="2" Width="600">
    <Tr>
        <Td Align="Center">Spaltenüberschrift1</Td>
        <Td Align="Center">Spaltenüberschrift2</Td>
    </Tr>

    <%
    String url    = "jdbc:odbc:SQL_Zugriff";
    String sqlSt = "Select ProductID from Products";
    String dr     = "sun.jdbc.odbc.JdbcOdbcDriver";

    Class.forName(dr);
    Connection con = DriverManager.getConnection(url, "", "");
    Statement st = con.createStatement();

    ResultSet rs = st.executeQuery(sqlSt);
    while(rs.next()) {
    <Tr>
        <Td><%=rs.getString("ProductID")%></td>
        .../weitere...
    </Tr>

    <%
    }
    rs.close();
    con.close();
    %>
    </Table>
</Body>
<Html>

```

Da Java Server Pages unter Lotus Notes grundsätzlich als Servlets behandelt werden, ist auch der Einsatz von JSP unter Lotus Notes möglich. Hierfür müssen Sie unter dem Lotus Domino Server die *Servlet file extensions*: auf **.jsp* einstellen. Danach brauchen Sie nur noch zusätzlich die Packages von Lotus Notes einzubinden:

```
<% page import="lotus.domino.*"%>
```

Somit stehen Ihnen JSPs auch unter Lotus Notes zur Verfügung.

6.10 Java Beans

Mit Java Beans liefert Sun eine Technologie für Softwarekomponenten. Hierbei handelt es sich um Komponenten, die mit bestimmten Methoden und Helferklassen zur Entwurfszeit aktiv sind. Dabei können die Beans visuell zur Entwurfszeit manipuliert werden. Zur Laufzeit verhält sich ein Bean wie eine normale Java-Klasse.

6.10.1 Enterprise Java Beans

Enterprise Java Beans, kurz EJBs genannt, stellen ein spezifiziertes Komponentenmodell für Java Beans dar, das serverseitig läuft und in Verbindung mit anderen Beans zur Erstellung neuer Applikationen dient. Dadurch ist es möglich, Anwendungen multiuserfähig, skalierbar, plattformunabhängig und transaktional zu entwickeln. Für die Entwicklung von EJBs benötigen Sie auf jeden Fall die Java 2 Enterprise Edition (J2EE) und einen entsprechenden Applikations-Server.

Der nachfolgende Abschnitt gibt eine grundlegende Einführung in die EJB-Technologie.

6.10.2 Architektur von EJB

Die EJB-Architektur von Java stellt ein Framework für serverseitige Objekte im Applikations-Server-Umfeld dar. Es stellt dem Entwickler die Multi-Tier-Architektur zur Verfügung.