ZT IT-KOLUMNE

Einstieg in die Programmierung von Computern – Teil XIII

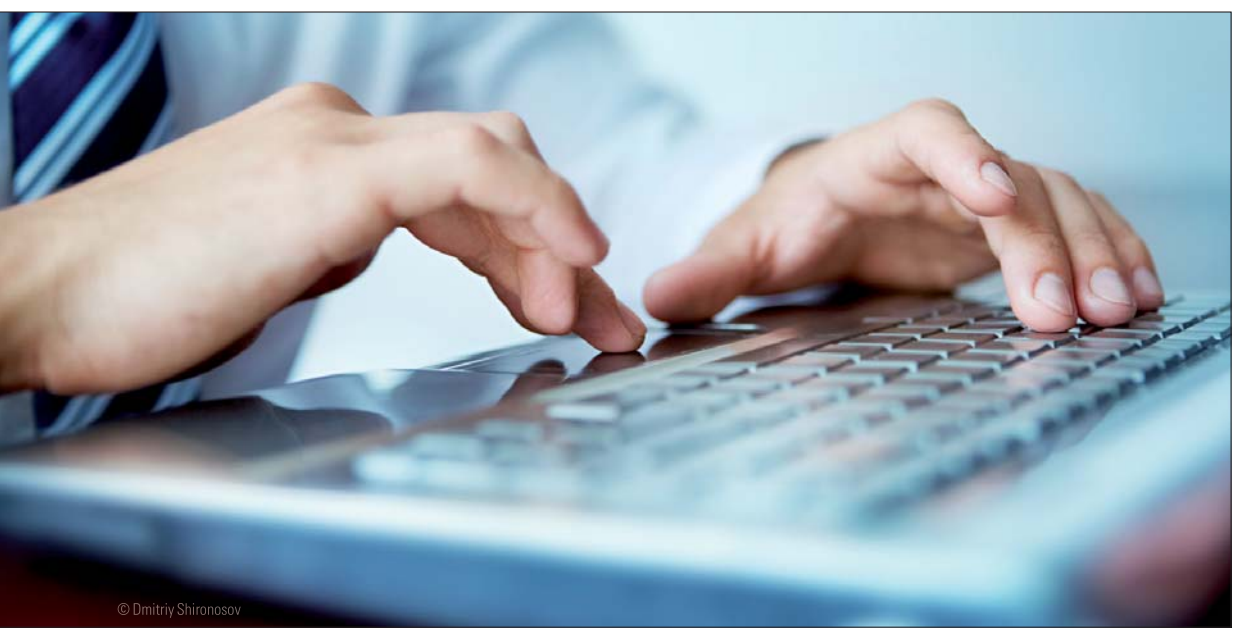
In diesem Teil wollen wir nun unser erstes Servlet-Programm zu einem richtigen JSF-Programm ausbauen. Wir werden sehen, wie die Client-Software mit der Server-Software kommuniziert und was dazu benötigt wird.

Kurze Wiederholung

Im letzten Teil haben wir ein einfaches erstes Servlet geschrieben, das eine vollständige HTML-Seite mit dem Begrüßungssatz „Hello Servlet!“ zum Browser zurückgesendet hat. Dazu verwendeten wir die Methode **„service(ServletRequest, ServletResponse)“** und haben diese überschrieben, d.h. wir haben in dieser Methode unseren eigenen Code implementiert. Das Überschreiben von Methoden in Java ist ein elementares Feature in der objektorientierten Softwareentwicklung. Der Servlet-Container hält in der Regel eine Instanz pro Servlet. Es wird somit für jeden Request ein eigener Thread gestartet.
In eigenen Servlets für HTTP-Anfragen (bearbeiten des Request) sollte man aber keine Service-Methode verwenden und diese dann überschreiben. Besser ist es, die Klasse von **„HttpServlet“** abzuleiten (die eigene Klasse erbt also von HttpServlet) und die beiden Methoden **„doGet()“** und **„doPost()“** zu verwenden und zu überschreiben. Damit übernimmt der Servlet-Container das korrekte Handling der HTTP-Methoden.

Die HTTP-Methoden

HTTP unterstützt in der Version 1.1 folgende acht Methoden: **GET, POST, HEAD, TRACE, PUT, DELETE, OPTIONS, CONNECT.** In unserer Java-Serie wollen wir aber nur die beiden Methoden **„GET“** und **„POST“** behandeln. Die anderen Methoden spielen oft keine Rolle, da auch viele Server diese gar nicht unterstützen. In der HTTP 1.1-Spezifikation werden sogar nur die Methoden **GET** und **HEAD** als verpflichtend beschrieben, doch unterstützt in der Regel jeder Server auch die **POST**-Methode


© Dmitriy Shironosov

HTTP-GET
Die HTTP-Methode GET ist wohl die meist verwendete Methode. Wird eine URL im Adresseingabefeld des Browsers eingegeben, so ist dies standardmäßig eine GET-Methode. Trägt der Benutzer Werte in Formularfelder einer Webseite ein, werden die Werte standardmäßig mit der GET-Methode zum Server übertragen. Die Parameter-Wertepaare werden dabei einfach durch ein Fragezeichen in der URL eingeleitet.

Beispiel-URL (Quelle: Wikipedia http://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol#HTTP_GET):
Auf der Startseite von Wikipedia wird in das Eingabefeld der Suche „Katzen“ eingegeben und auf die Schaltfläche „Artikel“ geklickt. Der Browser sendet folgende oder ähnliche Anfrage an den Server:

```
GET /wiki/Spezial:Search?search=Katzen&go=Artikel HTTP/1.1
Host: de.wikipedia.org
...
```

HTTP-POST
Soll jedoch die POST-Methode verwendet werden, muss dies im HTML-Code angegeben werden. Mit der POST-Methode können umfangreiche Informationen wie z. B. Dateien oder Meldungen an den Server übertragen werden. Des Weiteren wird die POST-Methode für größere Formulare verwendet. Der Hauptunterschied zwischen der

```
<!-- Datei: index.html -->

<!DOCTYPE html>
<html lang="en">
    <head>
      <meta charset="utf-8">
      <title>Unser erweitertes Servlet</title>
      <meta name="author" content="Thomas Burgard" />
    </head>

    <body>
      <h1>Kommunikation mit unserem HelloServlet</h1>
      <p>
          Beim Betätigen der Schaltfläche wird der eingegebene Vorname zum Hello-
          Servlet gesendet.
      </p>
      <p> </p>

      <form method=post action= "http://127.0.0.1/HelloServletErweitert">
          Mein Vorname: <input type=text name="yourname" />
          <br />
          <input type="submit" value="jetzt zum Server senden">
      </form>
    </body>
</html>
```

Kasten 1: Der HTML-Code mit dem Formular zur Dateneingabe.

```
// Datei: HelloServletErweitert.java

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

/**
 * Class: HelloServletErweitert
 *
 * HelloServletErweitert ist von HttpServlet abgeleitet und muss
 * deshalb die Methoden doGet und doPost überschreiben.
 */
public class HelloServletErweitert extends HttpServlet {

    /**
     * Nimmt die Nachricht vom Client entgegen.
     */
    protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws IOException, ServletException {

    // Die doGet-Methode verrichtet die Abeit
    doGet(request, response);
    }

    /**
     * Verarbeitet die vom Benutzer (Client-Seite) eingegebenen Daten.
     */
    public void doGet(HttpServletRequest request,
    HttpServletResponse response) throws IOException, ServletException{

      // 1. Nimmt die Daten vom Benutzer entgegen
      //    und verarbeitet die Daten.
      String vorname = request.getParameter("yourname");

      String client = request.getHeader("User-Agent");

      // 2. Die Antwortnachricht ist vom Typ "HTML"
      response.setContentType("text/html");

      // 3. Baut die Antwort in HTML zusammen
      PrintWriter out = response.getWriter();
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Antwort vom Servlet</title>");
      out.println("</head>");
      out.println("<body>");

        if (storeToDatabase(vorname) == true) {
                out.println("<p>Hello " + vorName + "</p>");
          out.println("<p>Der Vorname wurde in der Server-Datenbank
          gespeichert!</p>");
        } else {
          out.println("<p>Fehler: Du hast keinen Vornamen eingegeben!</p>");
        }

      out.println("</body></html>");
      out.close();
    }

    /**
     * Speichert den Vorname in der Server-Datenbank.
     * In diesem Beispiel soll eine ArrayList die Datenbank simulieren.
     */
    private boolean storeToDatabase(String vorname) {

        if (vorname == "") {
          return false;
        } else {
          List<String> database = new ArrayList<String>();

          // Speichert den Vornamen in der Datenbank
          database.add(vorName);
          return true;
        }
    }
}
```

Kasten 2: Das modifizierte Servlet „HelloServletErweitert“.

GET- und der POST-Methode ist, dass bei der POST-Methode die Parameter-Wertepaare nicht an die URL angehängt werden, sondern im Nachrichten-Body des Request. Die Parameter folgen, nachdem der Header-Bereich mit einer Leerzeile abgetrennt wurde.

Beispiel-URL (Quelle: Wikipedia http://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol#HTTP_GET):
Im folgenden Beispiel wird wieder der Artikel Katzen angefordert, doch diesmal verwendet der Browser aufgrund eines modifizierten HTML-Codes (method="POST") eine POST-Anfrage. Die Variablen stehen dabei nicht in der URL, sondern gesondert im Body-Teil, etwa:

```
POST /wiki/Spezial:Search HTTP/1.1
Host: de.wikipedia.org
Content-Type: application/x-www-form-urlencoded
Content-Length: 24

search=Katzen&go=Artikel
```

Möchte man z. B. geheime Daten (z. B. Nutzer-Passwort, …) übertragen, so ist die POST-Methode sicherlich die vorzuziehende Variante.

Zusammenspiel von Client und Server

Wir werden nun unser kleines Servlet-Beispiel vom letzten Teil so erweitern, dass unser Servlet vom Client (HTML Web-Client im Browser) vom Client eingegebene Daten entgegennehmen und anschließend verarbeiten kann. Das Servlet wird in unserem Beispiel den eingegebenen Vornamen zuerst in einer Server-Datenbank speichern (wird im Code nur angedeutet) und anschließend eine Antwort zum Client zurücksenden. In der Antwort ist dann der eingegebene und übertragene Vorname mit enthalten. Das Servlet baut sozusagen eine HTML-Seite zusammen und sendet diese als Response zum Client. In unserem Beispiel wird der empfangene Vorname im Servlet in das HTML eingebaut:

```
out.println("<p>Hello" + vorName
+ "<br />Du verwendest: " + client
+ "</p>");
```

Damit das Servlet überhaupt HTTP-Nachrichten vom Client empfangen und verarbeiten kann, muss im HTML-Formular-Tag in der HTML-Datei eine sogenannte „action“ eingetragen werden. In unserem Fall also der Name des Servlets, und zwar in dieser Form:

```
<form method=post action=
"http://127.0.0.1/HelloServlet">
```

Im Kasten 1 sehen Sie den Quellcode der einfachen HTML-Datei „index.html“. Diese Datei wird ebenfalls im selben Verzeichnis gespeichert, in dem auch die Datei HelloServletErweitert.java gespeichert ist.
Ansonsten kann im Servlet normaler Java-Code für die Datenverarbeitung verwendet werden. Man könnte z. B. eine eigene Java-Klasse für die Datenbank-Kommunikation entwickeln und im Servlet dann verwenden. In unserem Beispiel jedoch möchte ich für das Speichern des Vornamen in die Server-Datenbank

die Methode **„storeToDatabase (String vorname)“** mit einem einfachen „Vector“ als Datenbank verwenden.
Im Kasten 2 sehen Sie den Quellcode des Java-Servlets **„HelloServletErweitert“**.
Gehen Sie beim Kompilieren der Servlet-Klasse und dem Erzeugen der „HelloServletErweitert.war“ wie im letzten Teil der Serie beschrieben vor!
Der Deployment-Deskriptor, also die Datei web.xml, ändern wir wie im Kasten 3 gezeigt ab.

Nun starten Sie wieder den Tomcat-Server im XAMPP Control Panel und kopieren die vorher erzeugte Datei „HelloServlet Erweitert.war“ in das Tomcat-Verzeichnis „C:\xampp\tomcat\webapps“. Nachdem Sie die Datei erfolgreich kopiert haben, erzeugt der Tomcat-Server automatisch das Verzeichnis „HelloServletErweitert“. In diesem Verzeichnis ist auch die HTML-Datei „index.html“ gespeichert, die Sie nun wie folgt in Ihrem Browser aufrufen können:

http://127.0.0.1/ HelloServletErweitert

Geben Sie Ihren Namen ein und warten dann auf die Antwort vom Servlet!

Im nächsten Teil bleiben wir beim Thema Java-Webapplikationen und lernen die „Java Server Faces (JSF)“ kennen, die bei komplexeren Web-Oberflächen verwendet werden. Bleiben Sie also weiterhin dran! ZT

Infos zum Autor

ZT Adresse

Thomas Burgard Dipl.-Ing. (FH)
Softwareentwicklung
& Webdesign
Bavariastraße 18b
80336 München
Tel.: 089 540707-10
info@burgardsoft.de
www.burgardsoft.de
burgardsoft.blogspot.com
twitter.com/burgardsoft

Kasten 3: Der Deployment-Deskriptor „web.xml“.

```
<!-- Datei: web.xml -->

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                     http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
 version="3.0"
 metadata-complete="true">

    <display-name>Hello World erweitert</display-name>
    <description>Das zweite Java-Servlet.</description>

    <servlet>
        <servlet-name>HelloServletErweitert</servlet-name>
        <servlet-class>HelloServletErweitert</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloServletErweitert</servlet-name>
        <url-pattern>/HelloServletErweitert</url-pattern>
    </servlet-mapping>
</web-app>
```