

## Einstieg in die Programmierung von Computern – Teil XII

Im Teil 12 werden wir nun ein erstes kleines Java-Servlet-Programm als Webapplikation entwickeln und auf dem eigenen Computer starten. Wie das funktioniert und was dazu benötigt wird, möchte ich in diesem Artikel erklären.

### Installation von XAMPP mit dem Tomcat-Server

Wie bereits im letzten Teil der Serie beschrieben, wird für den Ablauf von Java-Webapplikationen ein bestimmter Server benötigt, der die Java-Servlet- und JSP-Spezifikationen implementiert hat. Eine kostenfreie Open-Source-Lösung ist der „Tomcat-Server“, der im „XAMPP-Projekt“ (ebenfalls Open-Source) enthalten ist. Gehen Sie auf folgende Website und laden Sie sich die XAMPP-Software für Windows auf Ihren Rechner ([www.apachefriends.org](http://www.apachefriends.org)).

Bei der Installation ist es wichtig, dass die XAMPP-Software direkt unter „C:\“ installiert wird, sodass nach der Installation XAMPP im Verzeichnis „C:\xampp“ installiert ist. Dabei befindet sich das Verzeichnis für „Tomcat“ unter „C:\xampp\tomcat“. Nach der Installation können Sie XAMPP mit einem Doppelklick auf das XAMPP-Symbol auf dem Desktop starten. Es wird das XAMPP-Control-Panel geöffnet, in dem der Tomcat-Server (ganz unten) separat gestartet und gestoppt werden kann. Im Normalfall braucht nichts weiter konfiguriert werden (siehe Abbildung)! Auf der Wikipedia-Website steht zu XAMPP folgende Information: „XAMPP ist eine Zu-



sammenstellung von freier Software – vorwiegend im Umfeld des LAMP-Systems. XAMPP ermöglicht das einfache Installieren und Konfigurieren des Webservers Apache mit der Datenbank MySQL bzw. SQLite und den Scriptsprachen Perl und PHP (mit PEAR). Das X steht hierbei für die verschiedenen Betriebssysteme, auf denen es eingesetzt werden kann. XAMPP enthält zusätzlich andere nützliche Werkzeuge wie den FTP-Server ProFTPd oder

FileZilla Server, den Mailserver Mercury, phpMyAdmin, Webalizer und OpenSSL. Seit Version 1.7.4 beinhaltet die Windows-Variante zusätzlich auch Apache Tomcat 7, der die Ausführung von JavaServer Pages und Java Servlets ermöglicht.“

### Die Struktur einer Java-Webapplikation

Bevor wir aber eine Java-Servlet Webapplikation schreiben werden und auf dem Tomcat-Server bereitstellen (deployen), müssen wir wissen, wie denn überhaupt eine Java-Servlet Webapplikation von der Verzeichnisstruktur aussieht. Es ist zu beachten, dass die Verzeichnisstruktur einer Java-Webapplikation immer gleich ist. Die Webapplikation befindet sich in einem ganz bestimmten Server-Verzeichnis. Zum Beispiel kann die Webapplikation „Hallo Servlet“ über die URL „http://127.0.0.1/Hallo-Servlet“ (wobei die IP-Adresse 127.0.0.1 die lokale IP-Adresse des eigenen Rechners ist) angesprochen werden. Dieser Pfad wird als „Kontext-Pfad“ bezeichnet und befindet sich auf dem Tomcat-Server im Unterverzeichnis „webapps“, also „C:\xampp\tomcat\webapps“. Im Kontext-Pfad ist in der Servlet-Spezifikation eine ganz bestimmte Verzeichnisstruktur für die Organisation einiger Ressourcen der Java-Webapplikation vorgeschrieben:

#### /WEB-INF

Verzeichnis für den sogenannten „Deployment Deskriptor“ **web.xml**. Der Deployment Deskriptor beinhaltet die Informationen über die Java-Webapplikation und beschreibt auch die Beziehungen zwischen den Komponenten.

#### /WEB-INF/lib

Hier werden die Java Archiv-Dateien (.jar-Dateien) gespeichert. Die .jar-Dateien beinhalten die für die Webapplikation notwendigen Ressourcen.

#### /WEB-INF/classes

In diesem Verzeichnis werden alle Servlets und alle dazugehörigen Java-Klassen der Webapplikation gespeichert. Die Verzeichnisstruktur für unsere Java-Servlet Beispiel-Webapplikation „Hallo-Servlet“, die wir entwickeln wollen:

```
C:\xampp\tomcat\webapps\
Hello-Servlet\
  -> WEB-INF
      -> \classes
      -> \lib
  -> zusätzliches Verzeichnis 1
      (z. B. Bilder)
  -> zusätzliches Verzeichnis 2
      (z. B. Dokumente)
```

### Das erste kleine Java-Servlet-Programm

Wenn wir nun unsere erste Java-Webapplikation entwickeln, werden wir aber die Java-Quelldateien nicht im Tomcat-Webverzeichnis speichern (das ist die übliche Vorgehensweise), sondern in einem eigenen Projekt-Arbeitsverzeichnis außerhalb des Tomcat-Servers, z. B. „C:\Java-Webapps\Halo-Servlet“:

```
C:\Java-Webapps\Halo-Servlet\
  -> WEB-INF
      -> \classes
  -> \src
```

Unter „src“ liegen prinzipiell alle Java-Quelldateien einer Webapplikation. Eine Java-Webapplikation besteht mindestens aus einem **Java-Servlet** und dem **Deployment-Deskriptor**

„web.xml“. In unserer ersten kleinen Java-Webapplikation haben wir nur eine Datei mit dem Namen „**HelloServlet.java**“. Es werden zunächst keine weiteren Ressourcen wie z. B. Bilddateien benötigt. Im Kasten 1 sehen Sie den Quellcode der Datei „HelloServlet.java“.

### Erklärung der Java Quellcode-Datei „HelloServlet.java“

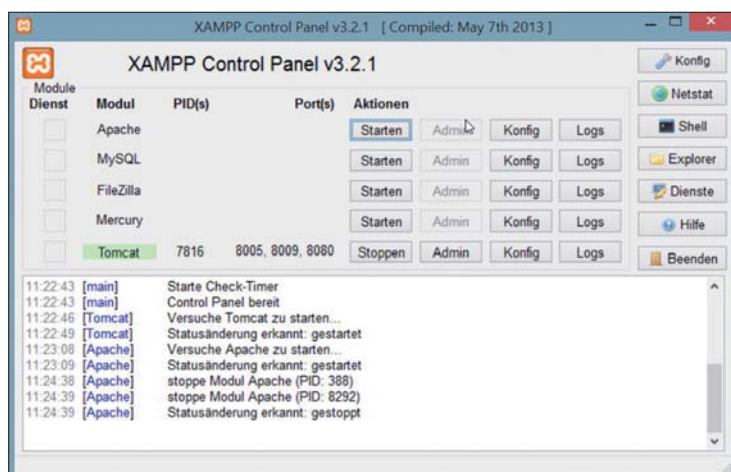
Unsere Java-Klasse „HelloServlet“ ist von der Klasse „**GenericServlet**“ abgeleitet. Diese Klasse ist im Paket „**javax.servlet**“ enthalten (muss deswegen in unsere Webapplikation importiert werden). **GenericServlet** ist ein generisches und protokollunabhängiges Servlet. Da unsere Klasse **HelloServlet** von **GenericServlet** abgeleitet ist, kann die Methode „**service()**“ von uns überschrieben werden.

Was passiert da eigentlich? Ganz einfach: Bei jeder Anfrage an das Servlet wird die Methode „**service()**“ vom Servlet-Container aufgerufen d. h. die Methode „**service()**“ hat somit die *Funktionalität des Servlets implementiert*. Als Übergabeparameter bekommt „**service()**“ vom Servlet-Container die Objekt-Referenzen von **ServletRequest** und **ServletResponse** übergeben, mit denen einerseits Daten von der Client-Seite gelesen werden können (**ServletRequest resp**) und andererseits Antworten wieder zum Client zurückgesendet werden können (**ServletResponse resp**). In unserem

ANZEIGE



Beispiel wird eine HTML-Seite wieder zum Client-Browser (über ein **PrintWriter**-Objekt) zurückgesendet. Mittels der Anweisung „**resp.setContentType(“text/html”);**“ (auch als **MIME-Bezeichner** bezeichnet) wird dem Client-Browser mitgeteilt, welche Art von Daten die Ant-



„XAMPP Control Panel“ zum Starten und Stoppen des Tomcat-Servers.

#### // Datei: HelloServlet.java

```
import java.io.*;
import javax.servlet.*; // Enthält die Klasse GenericServlet!

public class HelloServlet extends GenericServlet {

    public void service(ServletRequest req, ServletResponse resp) throws ServletException,
        IOException {

        resp.setContentType("text/html");
        PrintWriter output = resp.getWriter();

        // Jetzt wird eine HTML-Seite wieder an den Internet-Browser zurück-
        // gesendet!
        output.println("<html><body><h1>Hello Servlet!</h1></body></html>");
    }
}
```

Kasten 1: Java-Servlet HelloServlet.java.



wort beinhaltet. In unserem Fall also „HTML“.

Mit der Anweisung `output.println("<html><body><h1>HelloServlet!</h1></body></html>");` wird zum Schluss die HTML-Seite als String zurückgesendet.

#### Erklärung des „Deployment Deskriptors web.xml“

Hier steht der Text ...

ANZEIGE

**ZAHNWERK**  
Frästechnik GmbH

Ihr Fräs-zentrum

Testen Sie uns!

[www.zahnwerk.eu](http://www.zahnwerk.eu)

#### Kompilieren und Deployment unserer ersten Java-Webapplikation

Da ja unsere kleine Webapplikation eine ganz normale Java-Quelldatei ist, kann diese natürlich auch ganz einfach mit dem Java-Compiler kompiliert werden. Der Aufruf des Java-Compilers geschieht aus dem „src-Verzeichnis“ heraus. Öffnen Sie nun eine „Windows Eingabeaufforderung“

```
javac -cp C:\xampp\tomcat\lib\servlet-api.jar
-d C:\Java-Webapps\HelloServlet\WEB-INF\classes
HelloServlet.java
```

Bitte beachten Sie, dass Sie in der **Windows Eingabeaufforderung** den ganzen Befehl ohne Returns eingeben müssen. Der Schalter „-d“ besagt, dass die erzeugte .class-Datei im Verzeichnis „classes“ gespeichert werden soll.

#### Der Deployment Deskriptor 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 Servlet
</display-name>
<description>Das erste Java-Servlet.</description>
<servlet>
<servlet-name>HelloServlet
</servlet-name>
<servlet-class>HelloServlet
</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>HelloServlet
</servlet-name>
<url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
</web-app>
```

Im Deployment Deskriptor werden die umgebungsspezifischen Parameter der Webapplikation eingetragen bzw. festgelegt. Man kann auch sagen, dass damit die Webapplikation konfiguriert wird. Die Deklaration der Daten erfolgt in **XML-Notation** und ist relativ einfach zu verstehen (fast selbsterklärend!).

Die wichtigsten Parameter sind:

- **<servlet>**: Hier müssen der Name des Servlets und die Java-Klasse angegeben werden
- **<servlet-mapping>**: Hier wird die Zuordnung „Servlet – URL“ festgelegt, also welches Servlet soll unter welcher URL erreichbar sein?

Nun müssen wir noch eine **Java-Archivdatei** (Endung .war) erzeugen, mit der unsere Java-Webapplikation auf dem Tomcat-Server deployed bzw. installiert werden kann. Wir öffnen wieder die „Windows Eingabeaufforderung“ und wechseln in das Verzeichnis „C:\Java-Webapps\HelloServlet“. Hier geben wir den folgenden Befehl ein, um die Java-Archivdatei zu erzeugen:

```
jar -cf HelloServlet.war *
```

Die Schalter „c“ und „f“ geben an, dass ein neues Java-Archiv mit dem Namen „HelloServlet.war“ generiert werden soll. Der nachfolgende „\*“ besagt, dass der gesamte Inhalt im aktuellen Verzeichnis dem Archiv hinzugefügt werden soll.

Zum Schluss wird die Datei „HelloServlet.war“ in das Unterverzeichnis „webapps“ des Tomcat-Servers kopiert. Auf meinem Rechner ist das: **C:\xampp\tomcat\webapps**

Wenn der Tomcat-Server gestartet ist, wird automatisch das Verzeichnis „HelloServlet“ mit dem Inhalt der war-Datei erzeugt. Das automatische Erzeugen des Verzeichnisses nennt man auch „Hot Deployment“.

Nun kann man das Servlet im Internet-Browser aufrufen: <http://localhost:8080/HelloServlet/servlet>

#### Ausblick zum Teil 13

Im nächsten Teil werden wir das Servlet-Beispiel ausbauen und um neue Funktionen erweitern. Bleiben Sie also weiterhin dran! 



#### 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

Precision  
Hightech  
Speed  
Innovation



**Komplexe Geometrien?**  
Nutzen auch Sie die Zfx Fräszentren als kompetente Outsourcing-Partner!

## Zfx™ Inlab-Pakete: Industrie-Standards auf kleinstem Raum

Der **Zfx™ Evolution** ist mit einer absoluten Merkmalgenauigkeit von 9 µm im Volumenkörper (120 x 80 mm) laut VDI\* einer der exaktesten Dentalscanner auf dem Markt und ermöglicht beste Passung bei direkt verschraubten Arbeiten auf Implantatniveau! In der 5-Achs-Simultanbearbeitung garantiert die aus einem 200 kg Gusskörper bestehende **Zfx™ Inhouse5x** maximale Stabilität und höchste Genauigkeit auf kleinstem Raum. Die Fertigungseinheit eignet sich dabei sowohl zum Fräsen und Nass-Schleifen, was das Bearbeiten auch von harten Materialien wie Titan und Kobalt-Chrom ermöglicht. Die CAD/CAM Inlab-Systeme von Zfx werden nach höchster Industrie-Qualität in Deutschland gefertigt und finden auf Grund ihrer kompakten Maße Platz in jedem Labor. Überzeugen Sie sich selbst und erleben Sie die Zfx Systeme live auf der Zfx Dental Roadshow!

Detaillierte Informationen zu den Zfx Inlab-Systemen und alle aktuellen Termine der Zfx Dental Roadshow finden Sie auf [www.zfx-dental.com](http://www.zfx-dental.com)

