

## In diesem Kapitel:

- Installation und Konfiguration
  - *Tomcat installieren und testen*
- Fehlerbehandlung in JSPs
  - *Das Konzept der Fehlerbehandlungsseite*
- Scope von Objekten
  - *Ein einfacher Zähler*
- Formulare
  - *Formulare und die dazugehörige JavaBean*
- Musterlösungen und
- Lösungshinweise

# *Java Server Pages - Übungen*

Diese Übungen zeigen Ihnen, wie Sie Tomcat - die JSP Referenz Implementation benutzen können, um JSPs zu designen, zu implementieren und zu nutzen. Nach dem Durcharbeiten der Übungen sollten Sie wissen, wie JSP basierte Web Komponenten designed, übersetzt und genutzt werden können.

### 1.1.1. Generelle Bemerkungen zu den Übungen

Eine Übung besteht aus der Aufgabenstellung und Hilfestellungen. Einige der Übungen können Sie eventuell selbständig lösen; bei anderen werden Sie Hilfe oder Lösungshinweise benötigen, beispielsweise weil die Aufgabenstellung für Sie zu unverständlich ist! Sie können sich aber auch einfach die Musterlösung ansehen, sofern eine existiert (beispielsweise bei der Installation der Software kann man kaum eine Musterlösung erwarten: die Installation geschieht auf Ihrem Rechner).

Die einzelnen Übungen bauen aufeinander auf. Aber da Sie in der Regel eine Musterlösung haben, sollte es Ihnen trotzdem möglich sein, den Übungen zu folgen.

## 1.1.2. Die Anatomie einer Übung

Jede Übung enthält eine Liste jener Übungen, welche Sie bereits gelöst haben sollten, um die aktuelle Übung verstehen zu können. Zudem finden Sie Rahmenprogramme, welche Sie als Basis für die eigene Lösung verwenden können. Falls vorhanden, werden Sie auch noch Links zu den API Seiten finden, damit Sie eventuell benötigtes Hintergrundwissen ergänzen oder nachschlagen können.

Jede Übung enthält eine Liste der Lernziele, also der Ziele, die Sie hoffentlich erreichen.

Die Hilfestellungen finden Sie gleich anschliessend an die Fragestellungen, in der Regel Schritt für Schritt.

Sie finden also pro Übung folgende Informationen:

- **Hilfen**  
mit Hinweisen zur Lösung der gestellten Aufgabe;  
eine Lösungsskizze mit Kommentaren und allem drum und dran;
- **Musterlösung**  
Die Musterlösung enthält auch die Applet Tags und alle benötigten Dateien.
- **API Dokumentation**  
Links zu den Online Dokumentation bei Sun

## 1.1.3. Design Ziele für die Übungen

Es gibt drei Arten von Übungen:

- **"Blank-Screen"** Übungen  
Sie finden einen leeren Bildschirm vor ("Blank Screen") und müssen die gesamte Lösung selber erarbeiten.
- **Erweiterungen**  
Sie erweitern eine bereits bestehende Lösung
- **Korrektur**  
Sie korrigieren das Verhalten eines bestehenden Programms.

Um Ihnen das Erlernen der JSP Techniken zu erleichtern, werden in den Übungen in der Regel nur jene Teile zu erschaffen sein, welche spezifisch für JSP sind. Alle irrelevanten und komplexen Bereiche, die Sie eventuell zur Lösung benötigen, werden Ihnen zur Verfügung gestellt.

In der Regel werden die Übungen mit Hilfe eines Web Servers / des Webs durchgeführt. In den Fällen, in denen Sicherheitsprobleme auftauchen könnten, werden die Beispiele lokal ausgeführt.

## 1.1.4. Grundlegende JavaServer Pages Übungen

Aus heutiger Sicht sind folgende Aufgaben eine Basis, auf der aufbauend Sie eigene Problemstellungen lösen können. Zum Thema JSP existieren mehrere gute bis sehr gute Bücher, mit mehr oder weniger Praxisrelevanz.

### 1. Installation und Konfiguration von Tomcat

In dieser Übung beschaffen Sie sich die Tomcat Software - die JSP Referenz Implementation. Tomcat enthält einen einfachen HTTP Server und einen Web Container, der JSP Pages und Servlets ausführen kann. Welche Version Tomcat gerade aktuell ist und welche Versionen von JSP und Servlets unterstützt werden, können Sie direkt den Web Seiten entnehmen.

Den Server benötigen wir in den nachfolgenden Übungen!

#### Lernziele

- Installation von Tomcat
- Konfiguration Ihres Rechners, damit Sie JSPs übersetzen und nutzen können.

### 2. Exception Handling in JSP

In dieser Übung lernen Sie, wie man Laufzeit Exceptions, welche in JSPs auftreten können, auf eine Fehlerbehandlungsseite umleiten kann.

#### Lernziele

- Sie lernen, wie man mit Laufzeit Exceptions innerhalb der JSPs umgehen kann und wie diese auf Fehlerbehandlungsseiten umgeleitet werden können.
- Sie verstehen, wie man auf Exceptions aus einer JSP Fehlerbehandlungsseite zugreifen kann.

### 3. JSP Objekt Scope verstehen

In dieser Übung beobachten Sie das Verhalten einer Zähler-Bean innerhalb einer JSP Page mit unterschiedlichen Scope Attributen.

#### Lernziele

- Sie kennen die wichtigsten `scope` Attribute und deren Bedeutung beim Instanzieren der Beans mit Hilfe des `useBean` Tags.
- Sie kennen den Unterschied zwischen `session` und `application` Scope.

### 4. Forms / Formulare mit JSP

In dieser Übung lernen Sie, wie HTML Forms / Formulare mit JSP realisiert werden. Zudem lernen Sie verschiedene Features der JSP Engine kennen.

#### Lernziele

- Sie können einfache HTML Formulare kreieren und verarbeiten.
- Sie kennen die Bedeutung der JavaBeans bei der Bearbeitung von Formularen.

## 1.2. *Installation und Konfiguration von Tomcat*

Diese Übung führt Sie Schritt für Schritt durch den gesamten Prozess des Herunterladens und Installierens von Tomcat - der JSP Referenzimplementation (RI). Diese Übung befasst sich also speziell mit Tomcat. Falls Sie eine andere Software als Server einsetzen möchten, werden Sie in den folgenden Übungen entsprechende Anpassungen machen müssen.

### 1.2.1. Voraussetzungen

Keine!

### 1.2.2. Aufgaben

Überprüfen Sie Ihr System und die Systemanforderungen der aktuellen Version von Tomcat.

Laden Sie die aktuelle Version von Tomcat herunter (ich verwende die Version 3.2). Sie finden diese bei folgender Web Adresse:

<http://jakarta.apache.org/downloads/binindex.html>

Entzippen Sie das Archiv (bei mir ist dies `jakarta-tomcat-3.2.3.zip`). Ich habe die Dateien ins Root von meinem Drive D: entpackt, da C: schon voll ist.

Vergewissern Sie sich, dass die Umgebungsvariable `JAVA_HOME` definiert ist. Bei mir besitzt diese den Wert `c:\JBuilder\jdk1.3` oder `d:\jdk1.4`, je nachdem mit welcher JDK ich arbeiten möchte. Die Umgebungsvariablen setzen Sie am Besten in der Systemsteuerung unter System, Umgebung. Zudem sollte der Java Interpreter in der `PATH` Variable enthalten sein. Sonst müssen Sie den `java` Aufruf durch `%JAVA_HOME%\bin\java` ersetzen.

Wechseln Sie ins `bin` Verzeichnis der Tomcat Installation und starten Sie Tomcat mit dem `startup.bat` Skript.

Wenn Sie alles korrekt installiert und definiert haben, sollte jetzt Tomcat starten, auf Port 8080, und eine Ausgabe in einem DOS Fenster produzieren. Schauen Sie sich die Dokumentation zu Tomcat auf dem Java Web Site etwas an, um sich mit der Software vertraut zu machen. Der Web Link für die Dokumentation ist entweder der selbe wie oben oder

<http://java.sun.com/products/jsp/tomcat/>

### 1.2.3. Musterlösung

Zu dieser Übung gibt es keine Musterlösung. Falls Sie alle Aufgaben erfolgreich gelöst haben, was wirklich nicht so schwierig ist, sollte Tomcat installiert sein und laufen. Damit haben Sie die Basis für die folgenden Übungen gelegt.

## 1.2.4. Demonstration

Nachdem Sie die obigen Aufgaben gelöst haben, ist Tomcat oder Ihre Software Alternative installiert und konfiguriert und damit für die nachfolgenden Übungen verfügbar.

Beim Starten von Tomcat sollten Sie im Konsolen Fenster typischerweise folgende Ausgabe sehen:

```
2001-07-19 21:48:53 - ContextManager: Adding context Ctx( /examples )
2001-07-19 21:48:53 - ContextManager: Adding context Ctx( /admin )
Starting tomcat. Check logs/tomcat.log for error messages
2001-07-19 21:48:53 - ContextManager: Adding context Ctx( )
2001-07-19 21:48:54 - ContextManager: Adding context Ctx( /test )
2001-07-19 21:48:55 - PoolTcpConnector: Starting HttpConnectionHandler on 8080
2001-07-19 21:48:55 - PoolTcpConnector: Starting Ajp12ConnectionHandler on 8007
```

In älteren Versionen (beispielsweise 3.0) sehen Sie eine andere Ausgabe:

```
e:\jakarta-tomcat-3.0\bin> startup
```

```
Tomcat Web Server Version 3.0
Loaded configuration from: file:E:/tomcat/server.xml
Configuring web service using "default"
Configuring web service using
  "file:E:/tomcat/examples/WEB-INF/web.xml"
default: init
jsp: init
Configuring web service using "default"
Configuring web service using
  "file:E:/tomcat/webpages/WEB-INF/web.xml"
default: init
jsp: init
Starting tcp endpoint on 8080 with
  org.apache.tomcat.service.http.HttpConnectionHandler
Starting tcp endpoint on 8007 with
  org.apache.tomcat.service.connector.Ajp12ConnectionHandler
```

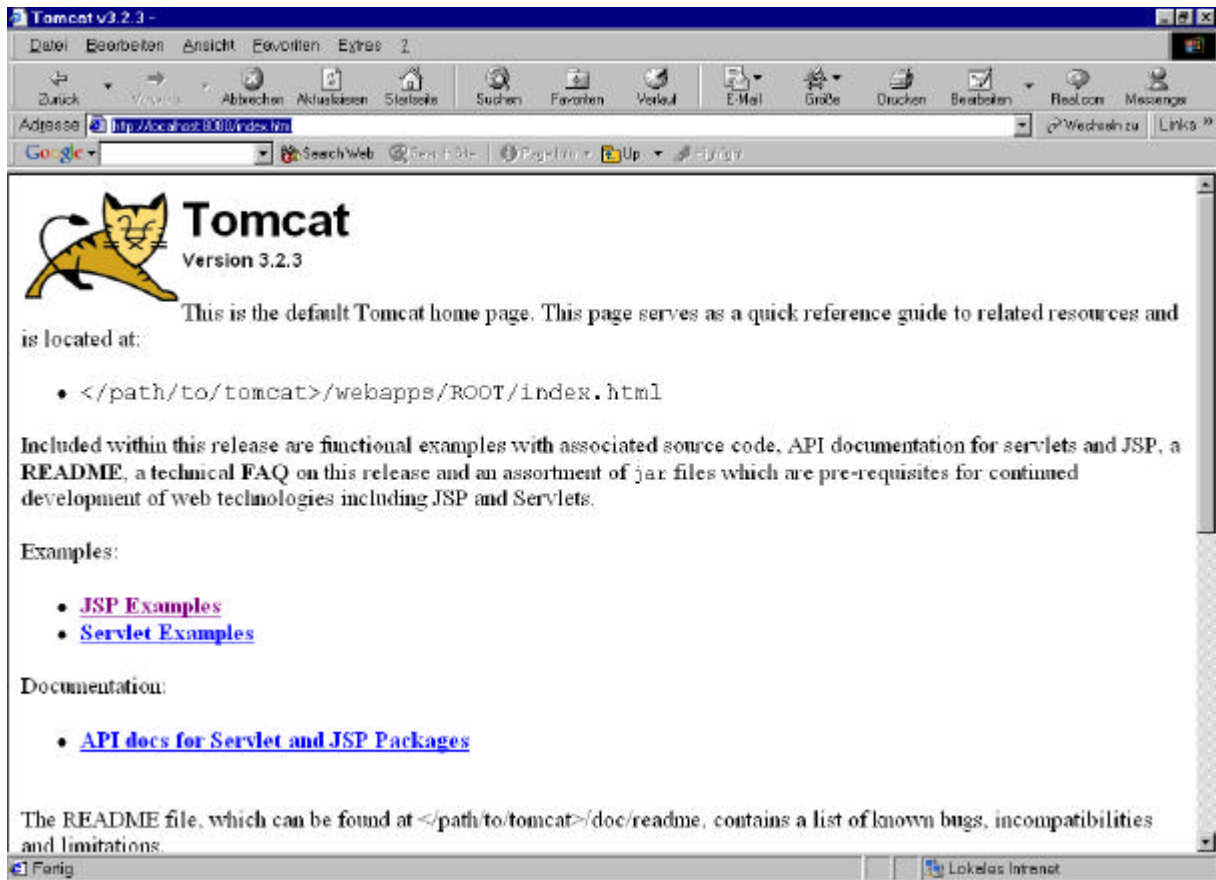
# JAVA SERVER PAGES - PRAXIS

Damit Sie sich auch gleich etwas motivieren können:

Starten Sie nun den Browser und bauen Sie eine Verbindung zum localhost auf Port 8080 auf. Voraussetzung dafür ist, dass Ihr PC TCP kennt und TCP korrekt installiert wurde und Sie Tomcat gestartet haben!

`http://localhost:8080/index.html`

Dann sollten Sie folgendes Bild sehen (Tomcat 3.2):



Wie Sie darauf sehen, können Sie auch gleich einige JSPs testen und auf die Dokumentation des APIs zugreifen.

Tomcat enthält, wie Sie oben bereits lesen konnten, einen Web Server, den wir hier ausnutzen.

Schauen Sie sich einige Beispiele an und spielen Sie etwas mit diesen. Sie werden sehen, dass die JSP Technologie recht einfach zu erlernen ist.

## 1.3. Ausnahmebehandlung in JSP

In dieser Übung befassen wir uns mit einem einfachen Beispiel für das Exception Handling. Dazu implementieren wir eine JSP Seite (`errhandler.jsp`), welche eine POST Operation beschreiben und eine Exception werfen. Die generierte Fehlermeldung wird von der JSP Engine auf eine eigene Fehlerbehandlungsseite als 'Error Handler' umgeleitet. Sie entwickeln auch die Fehlerbehandlungsseite `errorpage.jsp`, welche die Exception mit Hilfe der impliziten Variable `exception` empfängt und deren Informationen auswertet.

### 1.3.1. Voraussetzungen

Sie benötigen eine funktionsfähige Installation von Tomcat.

### 1.3.2. Rahmenprogramme

Sie erhalten zwei Lösungshinweise in Form von jsp Dateien:

#### 1.3.2.1. ERRHANDLER.JSP

```
<!-- Angaben zum Error Handler --%>
<html>
<body>
<form method=post action="errhandler.jsp">
Welches ist die coole'ste Programmiersprache im Universum?<p>
Java<input type=radio name=language value="JAVA" checked>
C++<input type=radio name=language value="CPP">
Visual Basic<input type=radio name=language value="VB">
<p>
<input type=submit>
</form>

<%
    if (request.getMethod().equals("POST")) {
        if (request.getParameter("language").equals("JAVA")) {
            out.println("<hr><font color=red>Sie haben eine gute Wahl
getroffen!</font>");
        } else {
            //werfen Sie eine Exception
        }
    }
%>

</body>
</html>
```



## 1.3.2.2. ERRORPAGE.JSP

```
<%-- Diese Seite ist eine Error Page --%>
<html>
<body>
<h1>
Fehlerbehandlungsseite
</h1>
<hr>
<h2>
Es wurde folgende Exception geworfen:<br>
<font color=red>
<%= exception.toString() %>
</font>
</h2>
</body>
</html>
```

## 1.3.3. Aufgaben

Schreiben Sie eine JSP Seite, `errhandler.jsp`, welche eine POST Anweisung bearbeiten kann.

Geben Sie als Fehlerbehandlungsseite die JSP Seite `errhandler.jsp` an, mit Hilfe der `page` Anweisung für die JSP Seite.

Verarbeiten Sie die Elemente der POST Anweisung:

- werfen Sie eine Exception, falls der ausgewählte Wert nicht Ihren Vorstellungen entspricht.
- falls die Antwort korrekt ist, dann geben Sie eine nette Meldung auf dem Bildschirm aus.

Entwickeln Sie eine Fehlerbehandlungsseite `errorpage.jsp`, welche die Laufzeit Exception anzeigen kann, also jene Exception, die bei falscher Auswahl geworfen wird.

Testen Sie die neuen Seiten mit Tomcat:

- kopieren Sie die Seiten in das Tomcat Verzeichnis, falls Sie Tomcat nicht so konfiguriert haben, dass Sie aus Tomcat auf das Übungsverzeichnis zugreifen können.
- starten Sie Tomcat
- schauen Sie sich die JSP Seite an und wählen Sie einmal korrekt und einmal falsch.
- prüfen Sie, ob die Fehlerbehandlung korrekt abläuft.

# JAVA SERVER PAGES - PRAXIS

## 1.3.4. Musterlösung

Sie finden auf dem Server / der CD Musterlösungen zu dieser Aufgabe. Da die Dateien nicht sehr gross sind, haben wir diese hier gleich abgebildet:

### 1.3.4.1. errhandler.jsp

```
<%@ page errorPage="errorpage.jsp" %>
<html>
<body>
<form method=post action="errhandler.jsp">
Welches ist die coolste Programmiersprache im Universum?<p>
Java<input type=radio name=language value="JAVA" checked>
C++<input type=radio name=language value="CPP">
Visual Basic<input type=radio name=language value="VB">
<p>
<input type=submit>
</form>

<%
    if (request.getMethod().equals("POST")) {
        if (request.getParameter("language").equals("JAVA")) {
            out.println("<hr><font color=red>Sie haben's erfasst!</font>");
        } else {
            throw new Exception("Schlechte Wahl!");
        }
    }
%>

</body>
</html>
```

### 1.3.4.2. errorpage.jsp

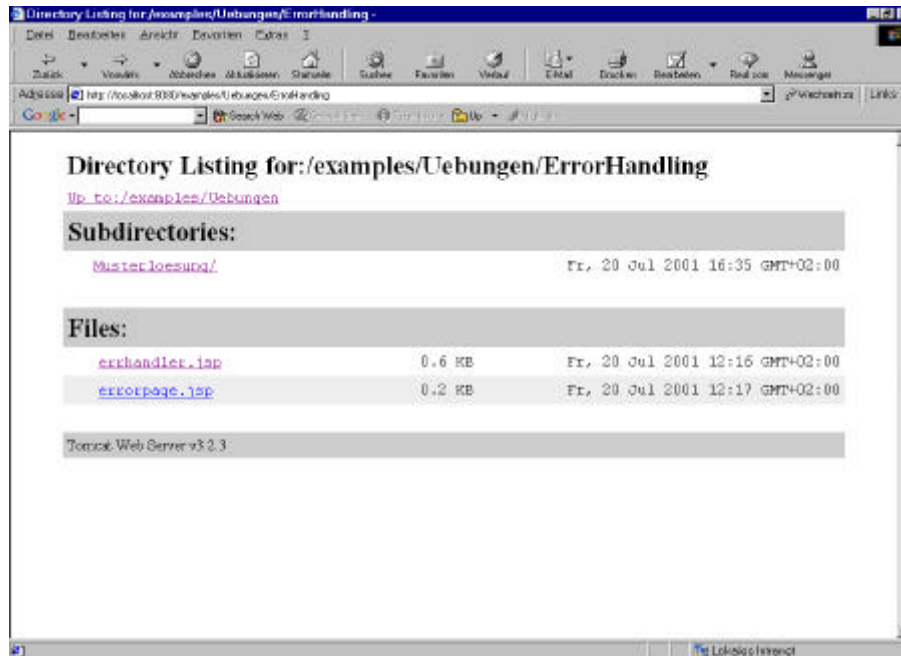
```
<%@ page isErrorPage="true" %>
<html>
<body>
<h4>
Fehlerbehandlungsseite
</h4>
<hr>
<b>
Die folgende Exception wurde geworfen:<br>
<font color=red>
<%= exception.toString() %>
</font>
</b>
</body>
</html>
```

# JAVA SERVER PAGES - PRAXIS

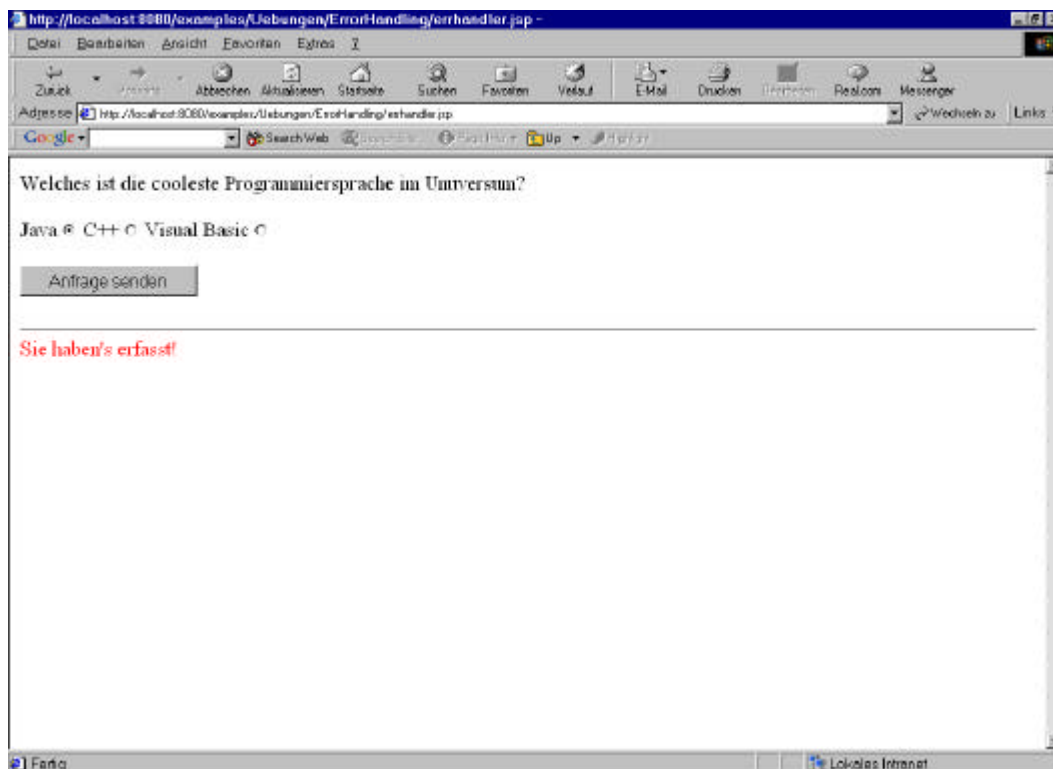
## 1.3.5. Demonstration

Hier meine Screen Snapshots zu dieser Übung:

### 1) Verzeichnis (localhost:8080/examples/Uebungen/ErrorHandling

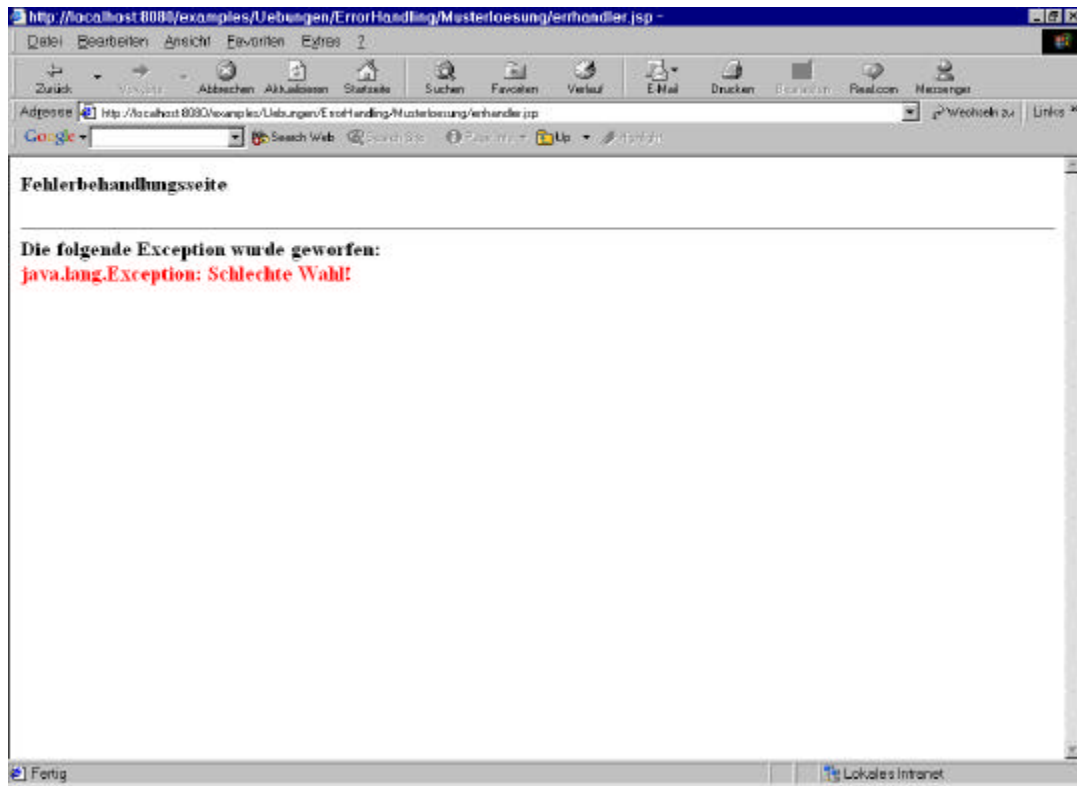


### 2) ein positives Erlebnis:



# JAVA SERVER PAGES - PRAXIS

3) ein negatives Erlebnis



Sie müssen dabei aber zuerst ins Verzeichnis "Musterloesung" wechseln, sonst sehen Sie diese Seite nie!

## 1.3.6. Lösungshinweise

*Schreiben Sie eine JSP Seite, `errhandler.jsp`, welche eine POST Anweisung bearbeiten kann.*

Der Einfachheit halber schreiben wir das HTML Formular gleich in die JSP. Beachten Sie, dass als Aktion / ACTION der POST Form die JSP selber angegeben werden muss.

Als Eingabe verwenden wir Radio Buttons zur Auswahl der Programmiersprache.

*Geben Sie als Fehlerbehandlungsseite die JSP Seite `errhandler.jsp` an, mit Hilfe der `page` Anweisung für die JSP Seite.*

Laufzeitfehler, welche nicht andersweitig abgefangen werden, werden automatisch an die Fehlerseite weitergeleitet. Wo sich diese Fehlerseite befindet, kann man im `page` Tag angeben.

*Verarbeiten Sie die Elemente der POST Anweisung:*

- *werfen Sie eine Exception, falls der ausgewählte Wert nicht Ihren Vorstellungen entspricht.*
- *falls die Antwort korrekt ist, dann geben Sie eine nette Meldung auf dem Bildschirm aus.*

Auf die Form Elemente kann man mit der Methode `request.getParameter()` zugreifen. Sie können auch Meldungen an den Browser senden: `response.println()`. Ausnahmen können Sie mit der Anweisung `new Exception("Eine Meldung")` absetzen.

*Entwickeln Sie eine Fehlerbehandlungsseite `errorpage.jsp`, welche die Laufzeit Exception anzeigen kann, also jene Exception, die bei falscher Auswahl geworfen wird.*

Eine JSP Fehlerseite kann mit dem `isErrorPage` Attribut des `page` Tags (`isErrorPage=true`) gekennzeichnet werden. Auf Laufzeitexceptions kann man über die Variable `<%= exception.toString() %>` zugreifen.

*Testen Sie die neuen Seiten mit Tomcat:*

- *kopieren Sie die Seiten in das Tomcat Verzeichnis, falls Sie Tomcat nicht so konfiguriert haben, dass Sie aus Tomcat auf das Übungsverzeichnis zugreifen können.*
- *starten Sie Tomcat*
- *schauen Sie sich die JSP Seite an und wählen Sie einmal korrekt und einmal falsch.*
- *prüfen Sie, ob die Fehlerbehandlung korrekt abläuft.*

Dazu finden Sie Batch Skripts, welche hoffentlich alles korrekt aufsetzen und Ihnen gestatten, Ihre Seiten zu testen.

## 1.4. JSP Objekt Scopes verstehen

In dieser Übung implementieren wir einen einfachen Zähler, `Counter.jsp`. Diese instanziert eine JavaBean zweimal, um den Zähler zu implementieren (`CounterBean.java`). Die eine JavaBean wird mit dem Scope `session`, die andere mit dem Scope `application` versehen. Bei jedem Aufruf der JavaBean wird der Zähler um eins weitergezählt. Sie können den Unterschied zwischen `session` und `application` Scope beobachten, indem Sie mehrere Browser Sessions eröffnen.

Jeder Browser unterhält einen eigenen Zähler für die Session. Aber der Zähler mit `application` Scope wird als globale Variable von beiden geteilt.

### 1.4.1. Voraussetzungen

Sie müssen Tomcat installiert haben und zwar so, dass die Software auch problemlos funktioniert.

### 1.4.2. Rahmenprogramme

#### 1.4.2.1. Counter.jsp

```
<%@ page import="counterbean.CounterBean" %>

<!-- definieren Sie passende Werte für die Klasse und die Scope Attribute -
-->

<jsp:useBean id="session_counter" class="" scope="" />
<jsp:useBean id="app_counter" class="" scope="" />

<% session_counter.increaseCount();
   synchronized(page) {
       app_counter.increaseCount();
   }
%>
<h4>
Anzahl gleichzeitiger Zugriffe :

<!-- definieren Sie passende Werte für das Namensattribut -->

<jsp:getProperty name="" property="count" />
</h4>
<p>
<h4>
Gesamtzahl Zugriffe:

<!-- definieren Sie passende Werte für das Namensattribut -->

<% synchronized(page) { %>
<jsp:getProperty name="" property="count" />
<% } %>
</h4>
```

## 1.4.2.2. CounterBean.java

```
package counterbean;
public class CounterBean {
    //deklarieren Sie eine Integer Variable (Zähler)

    public int getCount() {
        //liefern Sie den Zähler
        package counterbean;}

    public void increaseCount() {
        //inkrementieren Sie den Zähler
    }
}
```

## 1.4.3. Aufgaben

Entwickeln Sie eine einfache JavaBean als Zähler, CounterBean.java.

Übersetzen Sie diese JavaBean.

Nutzen Sie diese JavaBean in Tomcat: tragen Sie diese dort ein.

Entwickeln Sie eine JSP Seite, Counter.jsp, welche zwei Instanzen der Counter JavaBean kreiert, eine mit session Scope, eine mit application Scope.

Nutzen Sie diese JSP Datei beispielsweise mit Tomcat: kopieren Sie die Seite (und die JavaBean) in passende Verzeichnisse.

Starten Sie das Beispiel.

## 1.4.4. Musterlösungen

### 1.4.4.1. Counter.jsp

```
<%@ page import="counterbean.CounterBean" %>
<jsp:useBean id="session_counter" class="counterbean.CounterBean" scope="session" />
<jsp:useBean id="app_counter" class="counterbean.CounterBean" scope="application" />

<% session_counter.increaseCount();
    synchronized(page) {
        app_counter.increaseCount();
    }
%>
<h4>
Anzahl Zugriffe auf den Zähler (Session):
<jsp:getProperty name="session_counter" property="count" />
</h4>
<p>
<h4>
Anzahl Zugriffe (Applikation):
<% synchronized(page) { %>
<jsp:getProperty name="app_counter" property="count" />
<% } %>
</h4>
```

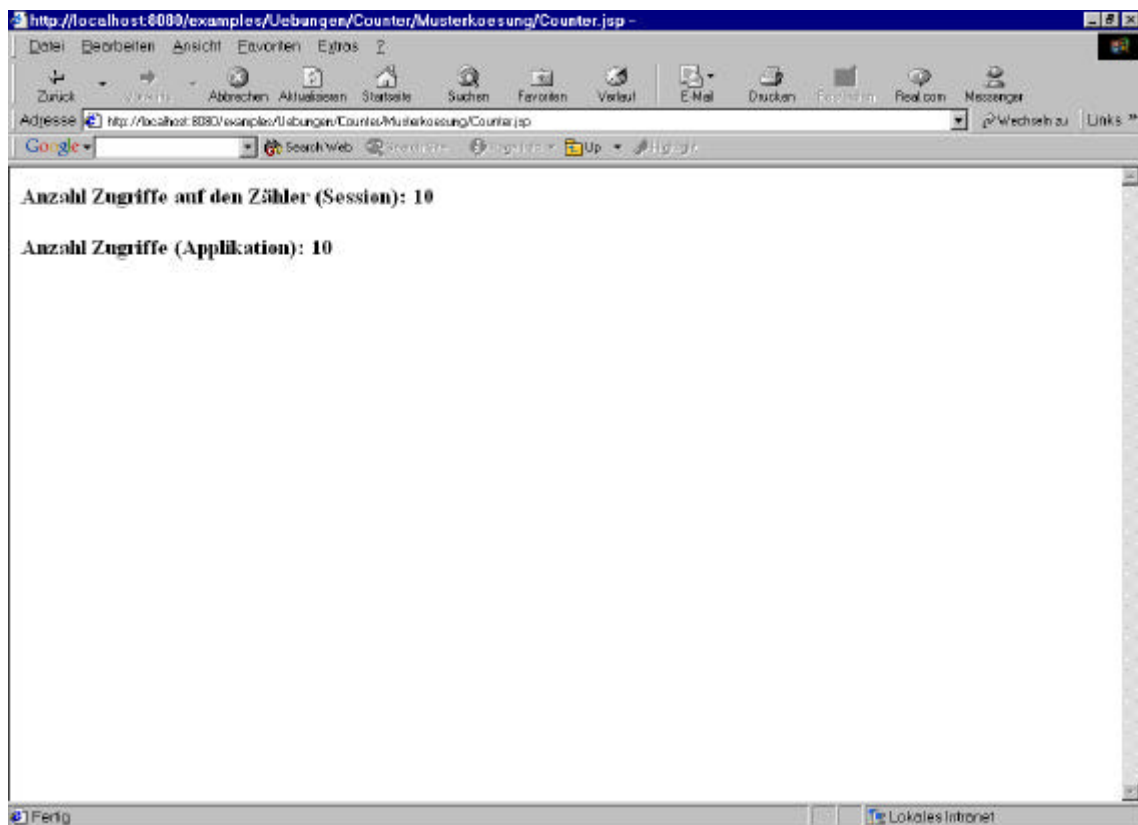
# JAVA SERVER PAGES - PRAXIS

## 1.4.4.2. CounterBean.Java

```
package counterbean;  
  
/**  
 * Title:          CounterBean  
 * Description:  
 * Copyright:     Copyright (c) J.M.Joller  
 * @author J.M.Joller  
 * @version 1.0  
 */  
  
public class CounterBean {  
    int count;  
  
    public int getCount() {  
        return count;  
    }  
  
    public void increaseCount() {  
        count++;  
    }  
}
```

## 1.4.5. Demonstration

Da ich lediglich einen Browser habe, konnte ich die Tests nicht umfassend genug ausführen.





## 1.4.6. Lösungshinweise

*Entwickeln Sie eine einfache JavaBean als Zähler, CounterBean.java.*

Sie können das Rahmenprogramm einfach durch Methoden ergänzen, welche die gewünschten Informationen liefern (hochzählen und ausgeben). Ich verwende dazu einfach JBuilder. Sie können aber auch das Bean Development Kit von Java verwenden oder irgend ein anderes Tool.

JBuilder gestattet es Ihnen auch JSPs zu kreieren!

*Übersetzen Sie diese JavaBean.*

Auch diesen Schritt habe ich im JBuilder erledigt. Sie finden dieses JBuilder Projekt auf dem Server / der CD.

*Nutzen Sie diese JavaBean in Tomcat: tragen Sie diese dort ein.*

Kopieren Sie die CounterBean.class Datei in das Verzeichnis

```
d:\jakarta-tomcat-3.2.3\webapps\examples\Web-inf\classes\counterbean
```

Dieses Verzeichnis entspricht unserem Package (counterbean).

*Entwickeln Sie eine JSP Seite, Counter.jsp, welche zwei Instanzen der Counter JavaBean kreiert, eine mit session Scope, eine mit application Scope.*

Die entsprechende Lösung finden Sie oben unter "Musterlösung".

*Nutzen Sie diese JSP Datei beispielsweise mit Tomcat: kopieren Sie die Seite (und die JavaBean) in passende Verzeichnisse.*

Die JSP wird mit der Batch Datei automatisch ins korrekte Verzeichnis verschoben.

*Starten Sie das Beispiel.*

Dazu greifen Sie einfach nach dem Starten von Tomcat auf

```
http://localhost:8080/examples/Uebungen/Counter/Musterkoesung/Counter.jsp
```

Eine Beispielausgabe sehen Sie oben unter "Musterlösung".

## 1.5. *Formulare mit JSP bearbeiten*

In dieser Übung entwickeln Sie eine einfache JSP Seite (`form.jsp`), welche ein HTML Formular mit typischen Eingabeelementen, wie Textboxen, Radio Buttons und Checkboxes verarbeiten kann.

Zudem entwickeln Sie wieder eine einfache JavaBean (`FormBean.java`) deren Property Namen mit jenen des Formulars übereinstimmen.

Testen Sie die JavaBean und das Formular mit einer einfachen POST Anweisung. Die Felder können Sie mit Introspektive des JSP Engine / Java Reflection verwalten.

### 1.5.1. Voraussetzungen

Sie sollten eine funktionsfähige Konfiguration von Tomcat zur Verfügung haben.

### 1.5.2. Rahmenprogramme

#### 1.5.2.1. FormBean.java

```
package formbean;

public class FormBean {
    // Properties für
    // Vorname, Nachname, Notify und Email und
    // Namen gemäss den Eingabeelementen
    private String[] languages;

    public FormBean() {
        firstName="";
        lastName="";
        email="";
        languages = new String[] { "1" };
        notify="";
    }

    // getter Methoden für firstname, lastname, notify, email und
    languages

    // setter Methoden für firstname, lastname, notify, email und
    languages
}
```



# JAVA SERVER PAGES - PRAXIS

```
</tr>

</table>
</center>
</form>

<%-- kreierte die JavaBean, falls das Formular submitted wird --%>
<%
if (request.getMethod().equals("POST")) {
%>

<jsp:useBean id="formHandler" class="formbean.FormBean">
<%-- wir benötigen einen setProperty Tag und müssen sicher sein, dass
Introspective eingesetzt wird --%>
</jsp:useBean>
<p>
<hr>
<font color=red>
<b>Ihre Eingabe:<P>
Vorname:</b><br>
<%-- Einsatz der get...() Methode und des getProperty Tag --%>
<br><b>Nachname:</b><br>
<%-- Einsatz der get...() Methode und des getProperty Tag --%>
<br><b>Email:</b><br>
<%-- Einsatz der get...() Methode um die email Adresse zu bestimmen,
zusammen mit dem getProperty Tag --%>
<b>Programmiersprachen:</b><br>
<%
    String[] lang = formHandler.getLanguages();
    if (!lang[0].equals("1")) {
        out.println("<ul>");
        for (int i=0; i<lang.length; i++)
            out.println("<li>"+lang[i]);
        out.println("</ul>");
    } else out.println("Es wurde keine Sprache ausgew&auml;hlt!<br>");
%>
<b>Kontaktaufnahme:</b><br>
<%-- Einsetz der get...() Methode und des getProperty Tag --%>
<br>
<%
}
%>
</font>
</body>
</html>
```

## 1.5.3. Aufgaben

Schauen Sie sich die JSP Vorgabe an. Dieses Formular senden den POST Request an sich selbst.:

```
<form action="/examples/Uebungen/Forms/Musterloesung/form.jsp" method=post>
```

Die JavaBean soll nur instanziiert werden, falls das Formular mit dem POST Befehl aktiviert wird. Die get...() Methoden sollen Introspection ausnutzen.

Verschieben Sie die JSP Seite zu Tomcat.

Entwickeln Sie eine passende JavaBean, mit Properties, die den Namen im Formular entsprechen.

Übersetzen Sie die JavaBean.

Verschieben Sie die JavaBean Klasse zum Tomcat Server.

Starten Sie das Beispiel.



# JAVA SERVER PAGES - PRAXIS

```
<input type="submit" value="Submit"> <input type="reset" value="Reset">
</td>
</tr>

</table>
</center>
</form>

<%-- Create the bean only when the form is posted --%>
<%
if (request.getMethod().equals("POST")) {
%>

<jsp:useBean id="formHandler" class="formbean.FormBean">
<jsp:setProperty name="formHandler" property="*" />
</jsp:useBean>
<p>
<hr>
<font color=red>
<b>Ihre Eingaben:<p>
Vorname:</b><br>
<jsp:getProperty name="formHandler" property="firstName" /><br>
<b>Nachname:</b><br>
<jsp:getProperty name="formHandler" property="lastName" /><br>
<b>Email:</b><br>
<jsp:getProperty name="formHandler" property="email" /><br>
<b>Programmiersprachen:</b><br>
<%
    String[] lang = formHandler.getLanguages();
    if (!lang[0].equals("1")) {
        out.println("<ul>");
        for (int i=0; i<lang.length; i++)
            out.println("<li>"+lang[i]);
        out.println("</ul>");
    } else out.println("Sie haben keine Programmiersprache
ausgew&auml;hlt<br>");
%>
<b>Befragungen:</b><br>
<jsp:getProperty name="formHandler" property="notify" /><br>
<%
}
%>
</font>
</body>
</html>
```

# JAVA SERVER PAGES - PRAXIS

## 1.5.4.2. FormBean.Java

```
package formbean;

/**
 * Title:          FormBean
 * Description:
 * Copyright:      Copyright (c) J.M.Joller
 * @author J.M.Joller
 * @version 1.0
 */

public class FormBean {
    private String firstName;
    private String lastName;
    private String email;
    private String languages[];
    private String notify;

    public FormBean() {
        firstName="";
        lastName="";
        email="";
        languages = new String[] { "1" };
        notify="";
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public String getEmail() {
        return email;
    }

    public String[] getLanguages() {
        return languages;
    }

    public String getLanguages(int index) {
        return languages[index];
    }

    public String getNotify() {
        return notify;
    }

    public void setFirstName(String x) {
        firstName = x;
    }

    public void setLastName(String x) {
        lastName = x;
    }

    public void setEmail(String x) {
```



# JAVA SERVER PAGES - PRAXIS

```
    email = x;
}

public void setLanguages(String[] x) {
    languages = x;
}

public void setLanguages(String x, int index) {
    languages[index] = x;
}

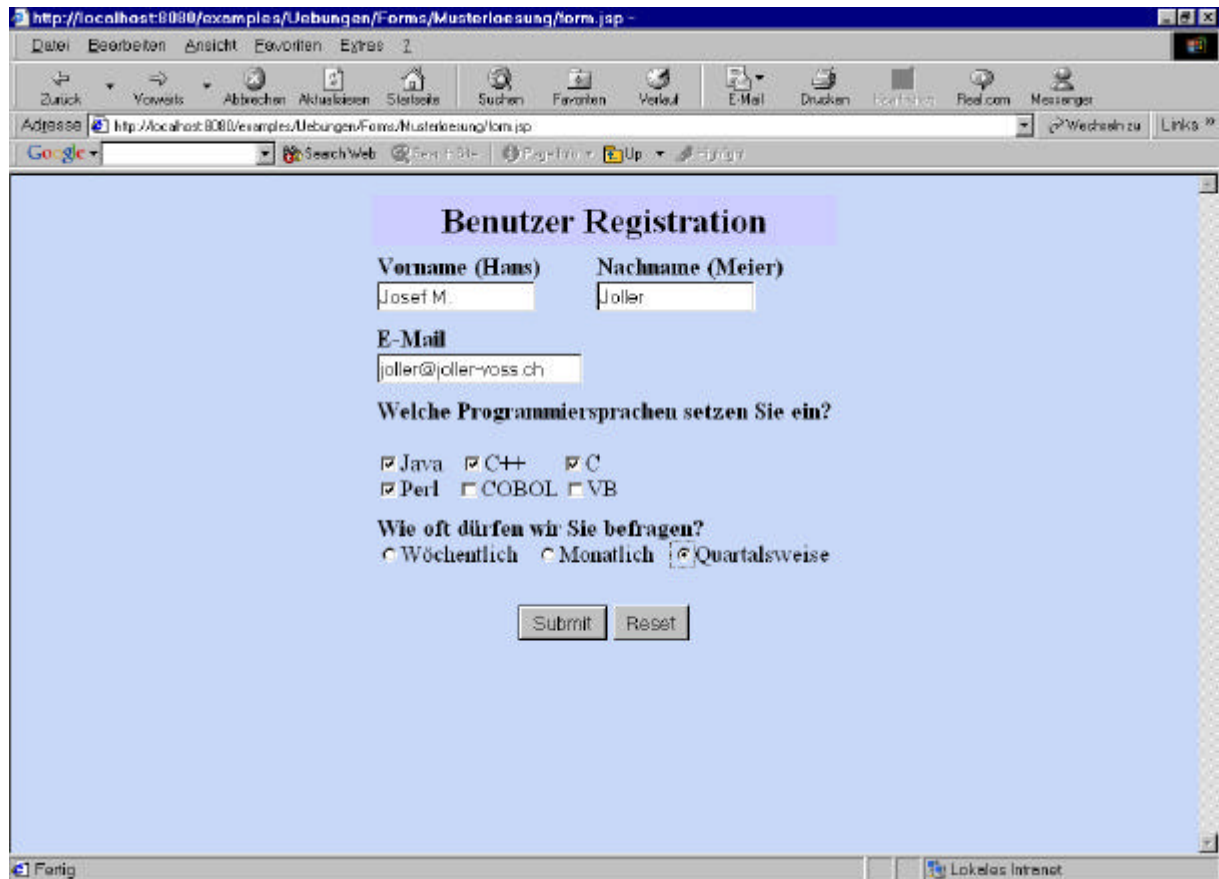
public void setNotify(String x) {
    notify = x;
}
}
```

# JAVA SERVER PAGES - PRAXIS

## 1.5.5. Demonstration

Hier einige Screen Shots:

1) das ausgefüllte Formular:



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/examples/uebungen/Forms/Musterlausung/form.jsp`. The browser's menu bar includes 'Datei', 'Bearbeiten', 'Ansicht', 'Favoriten', and 'Extras'. The address bar contains the URL and a search engine dropdown set to 'Google'. The main content area displays a registration form titled 'Benutzer Registration' on a light blue background. The form fields are filled with the following information:

- Vorname (Haas):** Josef M.
- Nachname (Meier):** Joller
- E-Mail:** joller@joller-yoss.ch

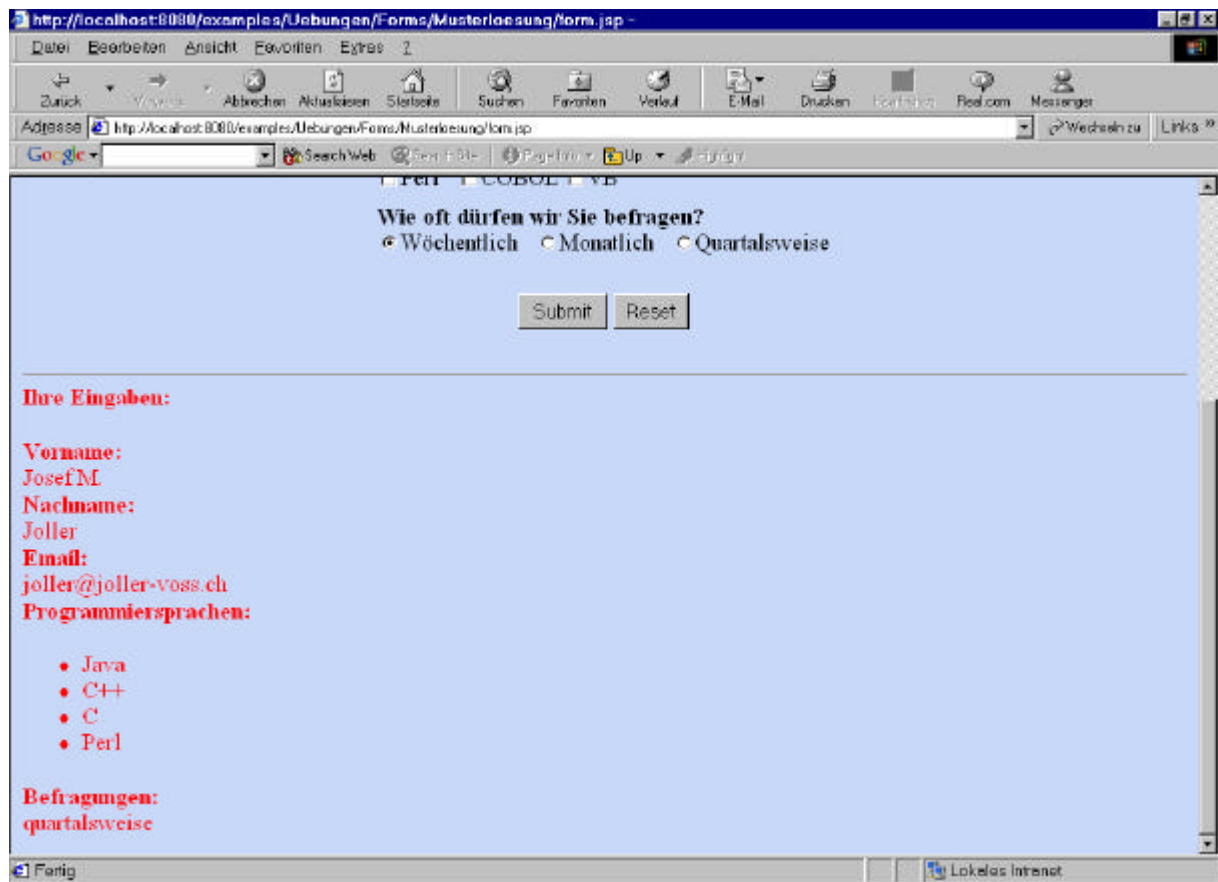
Below the form, there are two sections with radio button options:

- Welche Programmiersprachen setzen Sie ein?**
  - Java
  - C++
  - C
  - Perl
  - COBOL
  - VB
- Wie oft dürfen wir Sie befragen?**
  - Wöchentlich
  - Monatlich
  - Quartalsweise

At the bottom of the form are two buttons: 'Submit' and 'Reset'. The browser's status bar at the bottom shows 'Fertig' and 'Lokales Internet'.

# JAVA SERVER PAGES - PRAXIS

## 2) Der Feedback



Toll oder nicht?

## 1.5.6. Lösungshinweise

*Schauen Sie sich die JSP Vorgabe an. Dieses Formular senden den POST Request an sich selbst.:*

```
<form action="/examples/Uebungen/Forms/Musterloesung/form.jsp" method=post>
```

*Die JavaBean soll nur instanziiert werden, falls das Formular mit dem POST Befehl aktiviert wird. Die get...() Methoden sollen Introspection ausnutzen.*

Sie können den useBean Tag für die Instanzierung der JavaBean verwenden. Als Property verwenden wir "\*", weil wir das Java Reflection Interface/ JSP Engine Introspection verwenden wollen. In diesem Fall müssen die Felder des Formulars und im JavaBean gleich benannt werden.

*Verschieben Sie die JSP Seite zu Tomcat.*

Verschieben Sie die JSP ins passende Verzeichnis auf dem Tomcat Server. In meinem Fall ist dies D:\jakarta-tomcat-3.2.3\webapps\examples\Uebungen\Forms\Musterloesung>

*Entwickeln Sie eine passende JavaBean, mit Properties, die den Namen im Formular entsprechen.*

Sie finden die entsprechende JavaBean im JBuilder Projekt. Die get...() und set...() Methoden sind in unserem Fall simpel. Aber irgendwie muss man sich ja mit der Technologie vertraut machen.

*Übersetzen Sie die JavaBean.*

Dieser Schritt wird auch im JBuilder erledigt.

*Verschieben Sie die JavaBean Klasse zum Tomcat Server.*

```
D:\jakarta-tomcat-3.2.3\webapps\examples\WEB-INF\classes\formbean>
```

*Starten Sie das Beispiel.*

Sie sehen oben, wie die Beispieleingabe und -Ausgabe aussehen kann. Viel Spass beim Testen und beim Entwickeln weiterer Varianten.

# JAVA SERVER PAGES - PRAXIS

<b>JAVA SERVER PAGES - ÜBUNGEN.....</b>	<b>1</b>
1.1.1. <i>Generelle Bemerkungen zu den Übungen.....</i>	<i>1</i>
1.1.2. <i>Die Anatomie einer Übung .....</i>	<i>2</i>
1.1.3. <i>Design Ziele für die Übungen.....</i>	<i>3</i>
1.1.4. <i>Grundlegende JavaServer Pages Übungen.....</i>	<i>4</i>
1.2. <b>INSTALLATION UND KONFIGURATION VON TOMCAT.....</b>	<b>5</b>
1.2.1. <i>Voraussetzungen.....</i>	<i>5</i>
1.2.2. <i>Aufgaben .....</i>	<i>5</i>
1.2.3. <i>Musterlösung.....</i>	<i>5</i>
1.2.4. <i>Demonstration .....</i>	<i>6</i>
1.3. <b>AUSNAHMEBEHANDLUNG IN JSP .....</b>	<b>8</b>
1.3.1. <i>Voraussetzungen.....</i>	<i>8</i>
1.3.2. <i>Rahmenprogramme .....</i>	<i>8</i>
1.3.2.1. <i>ERRHANDLER.JSP .....</i>	<i>8</i>
1.3.2.2. <i>ERRORPAGE.JSP .....</i>	<i>9</i>
1.3.3. <i>Aufgaben .....</i>	<i>9</i>
1.3.4. <i>Musterlösung.....</i>	<i>10</i>
1.3.4.1. <i>errhandler.jsp.....</i>	<i>10</i>
1.3.4.2. <i>errorpage.jsp.....</i>	<i>10</i>
1.3.5. <i>Demonstration .....</i>	<i>11</i>
1.3.6. <i>Lösungshinweise.....</i>	<i>13</i>
1.4. <b>JSP OBJEKT SCOPES VERSTEHEN.....</b>	<b>14</b>
1.4.1. <i>Voraussetzungen.....</i>	<i>14</i>
1.4.2. <i>Rahmenprogramme .....</i>	<i>14</i>
1.4.2.1. <i>Counter.jsp.....</i>	<i>14</i>
1.4.2.2. <i>CounterBean.java.....</i>	<i>15</i>
1.4.3. <i>Aufgaben .....</i>	<i>15</i>
1.4.4. <i>Musterlösungen .....</i>	<i>15</i>
1.4.4.1. <i>Counter.jsp.....</i>	<i>15</i>
1.4.4.2. <i>CounterBean.Java .....</i>	<i>16</i>
1.4.5. <i>Demonstration .....</i>	<i>16</i>
1.4.6. <i>Lösungshinweise.....</i>	<i>17</i>
1.5. <b>FORMULARE MIT JSP BEARBEITEN.....</b>	<b>18</b>
1.5.1. <i>Voraussetzungen.....</i>	<i>18</i>
1.5.2. <i>Rahmenprogramme .....</i>	<i>18</i>
1.5.2.1. <i>FormBean.java.....</i>	<i>18</i>
1.5.2.2. <i>Form.jsp.....</i>	<i>19</i>
1.5.3. <i>Aufgaben .....</i>	<i>21</i>
1.5.4. <i>Musterlösung.....</i>	<i>22</i>
1.5.4.1. <i>Form.jsp.....</i>	<i>22</i>
1.5.4.2. <i>FormBean.Java .....</i>	<i>24</i>
1.5.5. <i>Demonstration .....</i>	<i>26</i>
1.5.6. <i>Lösungshinweise.....</i>	<i>28</i>