Java Server Pages

Leereenheid 4

Java Server Pages

INTRODUCTIE

Bij deze leereenheid hoort hoofdstuk 5 van Murach en Steelman, waarin u leert een eenvoudige webapplicatie te ontwikkelen met behulp van Java Server Pages.

LEERDOELEN

Na het bestuderen van deze leereenheid wordt verwacht dat u

- vorm en betekenis kent van de vijf JSP-tag-paren (scriptlets, expressies, directives, commentaar, declaraties)
- de in MS5 behandelde methoden van HttpServletRequest, GenericServlet en ServletContext kunt gebruiken
- de documentatie van deze klassen kunt vinden
- op drie verschillende manieren een JSP-pagina kunt opvragen
- weet hoe een JSP-pagina op de server wordt verwerkt en welke consequenties dat heeft met name voor het gebruik van attributen (instance variables)
- een eenvoudige webapplicatie kunt ontwikkelen bestaande uit HTML-pagina's, JSP-pagina's en Java-klassen
- de betekenis kunt geven van de volgende kernbegrippen: JSP-scriptlet, JSP-expressie, request-object, JSP-directive, JSP-commentaar, JSP-declaratie.

Studeeraanwijzingen

Hoofdstuk 5 van Murach en Steelman moet volledig bestudeerd worden. Tot de voorkennis voor deze cursus behoort basiskennis van HTML. Indien u niet over deze kennis beschikt, kunt u eerst MS4 *(A crash course in HTML)* doornemen.

In deze leereenheid ontwikkelt u voor het eerst zelf webapplicaties; we gaan ervan uit dat u daarvoor een Java-ontwikkelomgeving gebruikt. Op Studienet staat informatie over de ontwikkelomgeving die door de Open Universiteit wordt ondersteund. U kunt daar de software ophalen en handleidingen raadplegen voor het gebruik ervan, waaronder een handleiding voor het werken met webapplicaties. Als u al gewend bent aan een andere ontwikkelomgeving en daar liever mee werkt, is dat ook prima.

Hoofdstuk 3 van Murach en Steelman bevat aanwijzingen voor het maken van een webapplicatie in NetBeans. Voor andere ontwikkelomgevingen moet u zelf de documentatie raadplegen.

De studielast van deze leereenheid bedraagt circa 5 uur.

LEERKERN

1 The Email List application

Studeeraanwijzing

Bestudeer MS5_p138-143.

OPGAVE 4.1

Is de HTML-pagina getoond op MS5_p141 een statische of een dynamische pagina?

OPGAVE 4.2

Op MS5_p142 is sprake van een impliciet request-object. Waar komt dat object vandaan?

OPGAVE 4.3

Toon een mogelijke HTTP-request die verstuurd wordt na een klik op de verzendknop in het bovenste scherm van figure 5-1.
Neem aan dat het HTTP 1.1-protocol wordt gebruikt en dat er slechts één header is, en wel Host.

OPGAVE 4.4

In de derde alinea van MS5_p142 staat “the scriptlet is executed”? Wat wordt daar precies verwerkt?

2 How to code a JSP

Studeeraanwijzing

Bestudeer MS5_p144-149.

MS5_p145

Het laatste voorbeeld op MS5_p145, waarin een regel vijf keer wordt getoond, is de moeite van een zorgvuldige bestudering waard. Merk op dat een scriptlet midden in een blok kan eindigen; de afsluiting van dat blok vindt dan elders (in een andere scriptlet) plaats. Tussen die twee scriptlets staat HTML; deze verschijnt in de uitvoer elke keer dat het betreffende blok wordt verwerkt. Wordt het blok niet verwerkt, dan wordt er voor dat blok ook geen HTML gegenereerd.

Voorbeeld

Het volgende fragment leest een parameter getal uit, converteert deze naar een int-waarde en toont in de HTML alleen een melding als die waarde 0 of negatief is.

```
<%
  String sgetal = request.getParameter("getal");
  int getal = Integer.parseInt(sgetal);
  if (getal <= 0) {
%>
    U moet een positief geheel getal invoeren
<%
  }
%>
```

De accolade na de test en dus ook de afsluitende accolade in de tweede scriptlet zijn verplicht! Als ze ontbreken volgt soms een foutmelding. Om onduidelijke redenen wordt de code soms wel geaccepteerd maar heeft dan niet het gewenste effect: de HTML-uitvoer wordt dan altijd gegenereerd.

OPGAVE 4.5
Schrijf een combinatie van HTML, JSP-scriptlets en JSP-expressies waarmee de tafels van 2 tot en met 10 getoond worden. Een tafel heeft de volgende vorm:

```
1 x 2 = 2
2 x 2 = 4
…
10 x 2 = 20
```

Tussen twee tafels moet een extra lege regel staan.

MS5_p146,147

Het boek van Murach en Steelman behandelt lang niet alle klassen en methoden uit de servlet-packages. U zult daarom naast het boek soms ook de documentatie van deze packages moeten raadplegen. Op Studienet vindt u een link naar de documentatie (de packages staan niet in de standaard API specification).

OPGAVE 4.6
Gebruik de Servlet API Documentation om
a een Java-opdracht te schrijven die laat zien met welke HTTP methode een HTTP-request is verstuurd.
b een Java-opdracht te schrijven die de waarde toont van de header met naam Content-Length in de request.
c Java-opdrachten te schrijven die alle request-headers plus hun waarden tonen.
Aanwijzingen
– Het type van het request-object is HttpServletRequest, een interface uit de package javax.servlet.http
– Bekijk in de gewone API specification de interface Enumeration

Hoofdstuk 18 van Murach en Steelman (geen tentamenstof) geeft meer voorbeelden van het omgaan met HTTP-requests en -responses.

3 How to request a JSP

Studeeraanwijzing

Bestudeer MS5_p150-153

Uw eerste eigen webapplicatie is, zoals dat hoort, een variant op Hallo wereld.

FIGUUR 4.1 Webapplicatie Le04Kleuren

OPDRACHT 4.7

Ontwikkel, installeer en test een webapplicatie die de gebruiker begroet op een pagina met zelf gekozen kleuren voor de achtergrond en de tekst. Bij het starten van de webapplicatie ziet de gebruiker het scherm in figuur 4.1 (links). De rechterkant van figuur 4.1 toont uit welke kleuren gekozen kan worden. Als de gebruiker een naam heeft ingevoerd, twee kleuren heeft gekozen en op de verzendknop klikt, wordt een pagina getoond in de gekozen kleuren, met de tekst "Hallo *naam*".

Aanwijzingen

– De kleuren van achtergrond en tekst in HTML kunnen als attributen worden opgenomen in de <body>-tag, als volgt:

```
<body bgcolor=achtergrondkleur text=tekstkleur>
```

Gebruik voor het gemak de Engelse kleurnamen (white, black, blue, green, red en yellow), die door vrijwel alle browsers worden ondersteund.

– Kijk desgewenst op MS4_p128,129 hoe u een keuzelijst codeert in HTML.

4 How to use regular Java classes with JSPs

Studeeraanwijzing Bestudeer MS5_p154-159.

5 How to use three more types of JSP tags

Studeeraanwijzing Bestudeer MS5_p160-165.

MS5_p164,165

Zie paragraaf 2 van leereenheid 3

De informatie over de verwerking van servlets en JSP-pagina's op MS5_p164 is zeer belangrijk. Een 'instance variable' is een attribuut van de servlet-klasse die voor de pagina wordt gegenereerd. Het gebruik van dergelijke attributen in JSP-pagina's is eigenlijk nooit nodig, moet zeer sterk ontraden worden. Het is belangrijk dat u dit heel goed beseft, want de fouten die ontstaan door het gebruik van attributen worden pas zichtbaar wanneer een webapplicatie door verschillende gebruikers tegelijk wordt benaderd. Wanneer u uw applicaties alleen zelf test, zult u die fouten dus nooit ontdekken.

OPGAVE 4.8

Stel u maakt een webapplicatie waarvoor de gebruiker moet inloggen (denk bijvoorbeeld aan Marktplaats, EBay of Blackboard). Inloggen gebeurt via een JSP-pagina. Wanneer een al ingelogde gebruiker deze pagina opvraagt, wilt u in plaats van een inlogscherm alleen een knop met opschrift 'Log uit' tonen. U probeert dit te bereiken door een attribuut (instance variabele) 'ingelogd' te declareren. Heeft dit attribuut de waarde false, dan toont u de inlogpagina; na een geslaagde verwerking van de log-in geeft u het attribuut de waarde true. Heeft het attribuut de waarde true, dan toont u een uitlogpagina; na het uitloggen zet u het attribuut terug op false.

a Leg uit waarom deze oplossing niet werkt. Gebruik daarbij een scenario waarin verschillende gebruikers deze pagina opvragen.

b Helpt het om de variabele 'ingelogd' te declareren binnen een gewone scriptlet?

Attributen noch lokale variabelen kunnen gebruikt worden om te registreren wat een bepaalde gebruiker eerder met de applicatie heeft gedaan. Toch willen we dat vaak wel weten. Hoe dat wel kan, leert u in leereenheid 7.

6 How to work with JSP errors

Studeeraanwijzing

Bestudeer MS5_p166-170

MS5_p168,169

Merk op dat de servlet die door Tomcat gegenereerd wordt voor een JSP-pagina geen directe subklasse is van HttpServlet. De klasse is een subklasse van org.apache.jasper.runtime.HttpJspBase, die zelf wel weer een subklasse is van HttpServlet.

7 Exercises

OPDRACHT 4.9

a Maak een webproject uit de bouwsteen Le04Email, die overeenkomst met het project ch05email van Murach en Steelman.

b Voer onderdeel 2 tot en met 8 uit van Exercise 5-1 op MS5_p171.

Aanwijzing bij 5-1.4:

Genereer bij voorkeur een .war-file en installeer de applicatie binnen Tomcat.

c Voer onderdeel 9 uit van Exercise 5-1.

d Voer onderdeel 10 en 11 uit van Exercise 5-1.

OPDRACHT 4.10

Ontwikkel, installeer en test een webapplicatie met de startpagina getoond in figuur 4.2. Na verzenden wordt een pagina getoond als in figuur 4.3.

De Caesarcodering is een zeer eenvoudige (en zeer makkelijk te kraken) vorm van geheimschrift, waarbij elke letter wordt vervangen door een andere letter die een vast aantal posities verderop zit in het alfabet. Dat aantal is de sleutel. De twee regels hierna tonen de codes van elke letter bij een sleutel 5 (de code staat steeds onder de letter).

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
f g h i j k l m n o p q r s t u v w x y z a b c d e
```

Bij deze sleutel wordt het woord “onderzoeksgelden” afgebeeld op “tsijwetjpxljqijs”.

Voorafgaand aan het coderen worden alle hoofdletters in het bericht vervangen door kleine letters. Symbolen die geen letters zijn, worden bij codering overgeslagen. De code wordt getoond in groepen van vijf letters.

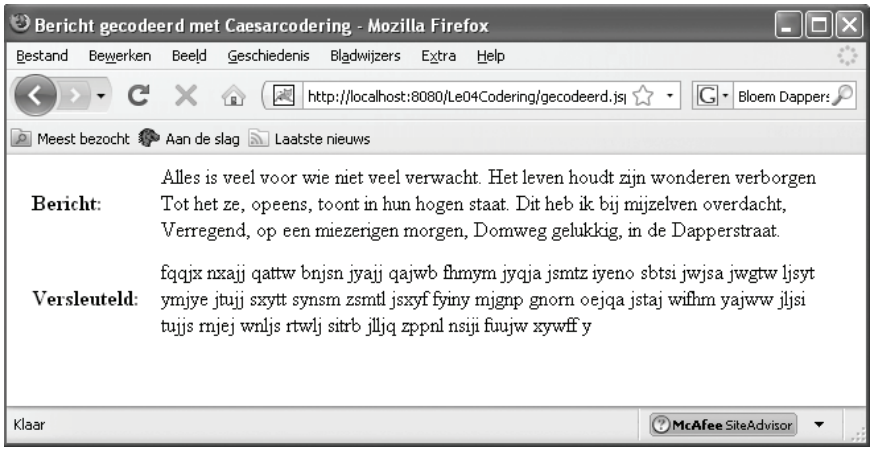
Aanwijzingen

– Op MS4_p130,131 kunt u vinden hoe een tekstgebied wordt gecodeerd in HTML.

– Denk na over welke HTTP-methode u gebruikt voor het verzenden van de gegevens

– Neem in het project een domeinklasse CaesarCodering op, die de codering verzorgt en ook een (gecodeerde) boodschap kan opdelen in groepen van vijf letters.

FIGUUR 4.2 Startpagina voor Caesarcodering



FIGUUR 4.3 Gegenereerde pagina voor Caesarcodering

ZELFTOETS

1 Welke van de volgende JSP-fragmenten zijn correct (dat wil zeggen: leiden niet tot foutmeldingen)? Ga er van uit dat alle gebruikte variabelen zijn gedeclareerd en van het juiste type zijn. Geef kort aan wat er fout is aan de incorrecte fragmenten.

a
```
<% import java.util.*; %>
```
b
```
<% request.getParameter(naam); %>
```
c
```
<%= 3 %>
```
d
```
<%= request.getParameter(naam); %>
```
e
```
<%@ page import java.util.*; %>
```
f
```
<%
  while (x > 0) {
%>
    x is nog steeds positief<br>
<%
    x = x - 1;
  }
%>
```
g
```
<%
  if (x > 0) {
   <%= x%>
  } else {
   <%= -x>
  }
%>
```

2 De variabele maanden bevat als waarde een ArrayList van Strings met alle Nederlandse namen voor de maanden (dus {“januari”, “februari”, …, “december”}. Welke HTML-code wordt gegenereerd door het volgende JSP-fragment?

```
<%
  for (String maand: maanden) {
    if (maand.indexOf('r') != -1) {
%>
  <%= maand %><br>
<%
    }
  }
%>
```

3 Welke uitspraken over het opvragen van JSP-pagina’s zijn correct?

a Een JSP-pagina kan zowel met HTTP-methode get als met HTTP-methode post worden opgevraagd.
b Wanneer een JSP-pagina voorkomt als waarde van het attribuut href binnen een link (anchor, A tag, <a ...>), is de methode altijd post.
c Aan een JSP-pagina die wordt opgevraagd vanuit een link (anchor, A tag, <a ...>), kunnen geen parameters worden meegegeven.

4 Gegeven een webwinkel met een JSP-pagina die wordt gebruikt voor het afrekenen van de gekochte artikelen. Een beginnende webprogrammeur wil de totale omzet bijhouden van deze artikelen. Hij neemt daartoe de volgende JSP-fragmenten op in de pagina:

```
<%!
  double totaalOmzet = 0;
%>

<%
  String strBedrag = request.getParameter("bedrag");
  double bedrag;
  ...
  totaalOmzet = totaalOmzet + bedrag;
%>
```

Op de puntjes staat code die de string omzet naar een waarde van type double.
a Leg uit wat deze twee fragmenten bewerkstelligen.
b Noem tenminste één reden waarom deze code niet voldoet.

TERUGKOPPELING

1 Uitwerking van de opgaven

4.1 Dit is een gewone, statische HTML-pagina; de pagina bevat immers geen enkel variabel element.

4.2 De code in de scriptlet wordt geplaatst in de doGet-methode van de servlet die gegenereerd wordt uit de JSP-pagina. Het request-object is een van de twee parameters van deze methode.

4.3 De URL in dit scherm is http://localhost:8080/ch05email/. Deze URL bevat een aanduiding van een map maar niet van een bestand; daarvoor wordt dus een standaard gebruikt. Welk bestand dat precies wordt, wordt bepaald op de server; het "path-to-resource"-deel van de HTTP-request bevat dus ook geen bestandsnaam. Het poortnummer wordt opgenomen in de host. De HTTP-request ziet er dan dus uit als volgt:

```
GET /ch05email/ HTTP/1.1
Host: localhost:8080
```

4.4 De regels code in de scriptlet komen terecht in de gegenereerde servlet. Het verwerken van de scriptlet betekent dat de overeenkomstige regels in de servlet verwerkt worden.

4.5 Deze combinatie ziet er bijvoorbeeld als volgt uit:

```
<%
  for (int tafel = 2; tafel <= 10; tafel++) {
    for (int i = 1; i <= 10; i++) {
%>
    <%= i %> x <%= tafel %> = <%= i*tafel %><br>
<%
    }
%>
<br>
<%
  }
%>
```

In plaats van de regel met drie JSP-expressies kan ook een regel gebruikt worden met slechts één expressie:

```
<%= i + " x " + tafel + " = " + i*tafel %><br>
```

4.6 a Hiervoor kan de methode getMethod van HttpServletRequest gebruikt worden, met de volgende omschrijving.

```
java.lang.String getMethod()
```
Returns the name of the HTTP method with which this request was made, for example, GET, POST, or PUT.

De Java-opdracht ziet er dan als volgt uit:

```
String methode = request.getMethod();
```

b Hiervoor kan de methode getHeader van HttpServletRequest gebruikt worden, met de volgende omschrijving.

```
java.lang.String getHeader(java.lang.String name)
```
Returns the value of the specified request header as a String.

De Java-opdracht ziet er dan uit als volgt:

```
String contentLength =
  request.getHeader("Content-Length");
```

c Hiervoor kunnen de methoden getHeaderNames en getHeader worden gebruikt. De eerste levert een lijst op van alle headernamen. Deze is van type Enumeration; dit is een voorganger van Iterator. Een Enumeration kan niet worden doorlopen met een for-each lus; in plaats daarvan worden de methoden hasMoreElements() en nextElement() gebruikt. Zie voor details de API documentatie.
De gevraagde code is:

```
Enumeration<String> headerNamen =
                          request.getHeaderNames();
while (headerName.hasMoreElements()) {
  String naam = headerNamen.nextElement();
  String waarde = request.getHeader(naam);
  System.out.println(naam + " " + waarde);
}
```

4.7 Deze webapplicatie bestaat uit twee delen, een HTML-pagina (die we index.html hebben genoemd) en een JSP-pagina die bij klikken op de verzendknop als antwoord wordt teruggestuurd. We tonen eerst de relevante delen van de HTML-pagina.

```
<html>
<head>
...
<title>Hallo in kleuren</title>
</head>
<body>
<form action="kleuren.jsp" method="get">
  <h3>Hoe heet je?</h3>
  <input type="text" name="naam">
  <p></p>
  <h3>Kies een kleur voor de achtergrond:</h3>
  <select name="achtergrond">
    <option value="white" selected>wit</option>
    <option value="black">zwart</option>
    <option value="blue">blauw</option>
    <option value="green">groen</option>
    <option value="red">rood</option>
    <option value="yellow">geel</option>
  </select>
  <p></p>
  <h3>Kies een kleur voor de tekst:</h3>
  <select name="tekst">
    ...
  </select>
  <p>
  <input type="submit" value="Verzenden">
</form>
</body>
</html>
```

De options in de tweede selectie zijn hetzelfde als in de eerste, alleen is nu de optie zwart aangewezen als selected.

Het relevante deel van de bijbehorende JSP-pagina met de naam kleuren.jsp ziet er als volgt uit:

```
<html>
<head>
...
<title>Hallo in kleuren</title>
</head>
<%
  String naam = request.getParameter("naam");
  String achtergrond = request.getParameter("achtergrond");
  String tekst = request.getParameter("tekst");
%>
<body bgcolor=<%= "\"" + achtergrond + "\"" %>
      text=<%= "\"" + tekst + "\"" %> >
<h1>Hallo <%= naam %></h1>
</body>
</html>
```

Let even op de JSP-expressies in de body-tag. Als bijvoorbeeld de kleuren zwart en geel zijn gekozen, dan moet de gegenereerde HTML er als volgt uitzien:

```
<body bgcolor="black" text="yellow">
```

De vereiste aanhalingstekens moeten in de JSP-expressie expliciet worden toegevoegd. Het werkt overigens ook als u dat niet heeft gedaan want browsers zijn heel soepel bij het interpreteren van HTML-code, maar het voldoet niet aan de standaard.

U kunt de pagina's ontwikkelen en testen in uw ontwikkelomgeving of in een browservenster; raadpleeg hiervoor de documentatie op Studienet. U kunt eventueel de JSP-pagina apart testen door in de URL al parameters mee te geven, bijvoorbeeld als volgt:

http://localhost:8080/Le04Kleuren/kleuren.jsp?naam=Jan&achtergrond=black&tekst=yellow

4.8 a Als een variabele wordt gedeclareerd als attribuut, dan is er op de server maar één instantie van dat attribuut, die gedeeld wordt door alle gebruikers. Stel u voor dat Jeanne als eerste de JSP-pagina opvraagt. Het attribuut ingelogd heeft de waarde false. Jeanne krijgt de inlogpagina te zien, logt in en het attribuut krijgt de waarde true. Nu wil Marius inloggen en vraagt de pagina op. Helaas krijgt Marius nu niet het login-scherm te zien. De waarde van het attribuut is immers true: de servlet toont daarom het uitlogscherm.
b Ook dit werkt niet. Variabelen die zijn gedeclareerd binnen scriptlets, worden in de gegenereerde servlet lokale variabelen van een methode die in een aparte draad wordt verwerkt, elke keer dat de pagina wordt opgevraagd. Bij een volgende verwerking is sprake van een nieuwe aanroep naar deze methode en dus ook van een nieuwe incarnatie van de lokale variabele. Deze kan niet gebruikt worden om informatie te onthouden over een eerder opvragen van de pagina.

4.9 a Geen terugkoppeling.

Exercise 5_1.5

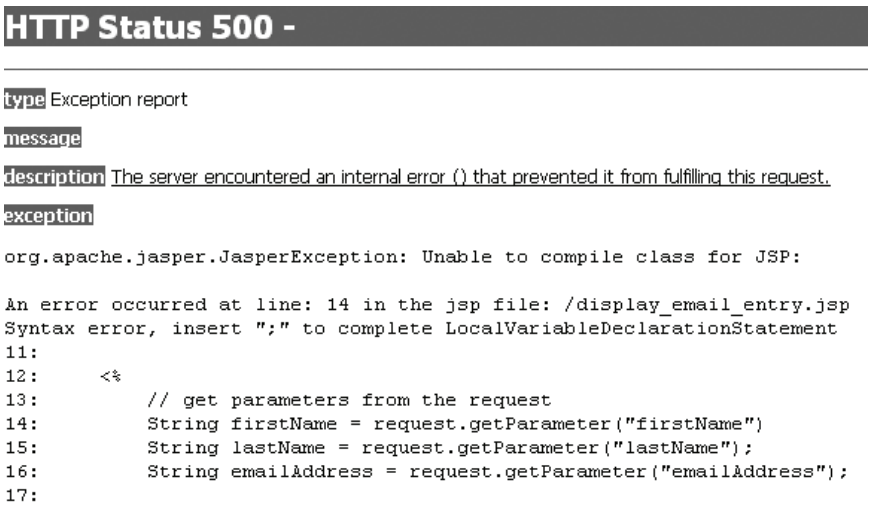
b Indien u het .war-bestand geplaatst heeft in de map webapps van Tomcat en de applicatie vanuit uw browser heeft verwerkt, vindt u het bestand EmailList.txt in webapps/Le04Email/WEB-INF.
Heeft u de applicatie verwerkt vanuit uw ontwikkelomgeving dan kunt u de locatie van het EmailList.txt-bestand eventueel opzoeken met behulp van de zoekfunctie van Windows.

Na het invoeren van twee gebruikers ziet dit bestand er bijvoorbeeld als volgt uit:

```
jantje@planet.nl|Jeanne|Smit
m.klinkhamer@ou.nl|Marius|Klinkhamer
```

Exercise 5_1.7

Zie figuur 4.4.



FIGUUR 4.4 Foutmelding van Tomcat bij syntaxisfout in scriptlet

c De gewijzigde pagina ziet er uit als volgt; wijzigingen zijn grijs gemarkeerd:

```
<html>
<head>
    <title>Murach's Java Servlets and JSP</title>
</head>
<body>
  <h1>Join our email list</h1>
  <p>To join our email list, enter your name and
   email address below. <br>
   Select the types of music you're interested in.<br>
   Then, click on the Submit button.</p>

  <form action="display_music_choices.jsp" method="get">
  <table cellspacing="5" border="0">
    <tr>
      <td align="right">First name:</td>
      <td><input type="text" name="firstName"></td>
    </tr>
```

```
    <tr>
      <td align="right">Last name:</td>
      <td><input type="text" name="lastName"></td>
    </tr>
    <tr>
      <td align="right">Email address:</td>
      <td><input type="text" name="emailAddress"></td>
    </tr>
  </table>
  <p>
    I am interested in these kinds of music:
  </p>
  <p>
    <select name="musicTypes" multiple="multiple">
      <option value="Rock" selected>Rock</option>
      <option value="Country">Country</option>
      <option value="Bluegrass">Bluegrass</option>
      <option value="Folk">Folk</option>
    </select>
  </p>
  <p>
    <input type="submit" value="Submit">
  </p>
</form>
</body>
</html>
```

Merk op dat we de methode meteen weer hebben gewijzigd in get (zie onderdeel 5-1.11) en dat de verzendknop nu buiten de tabel is geplaatst. Binnen de <select>-tag mag de waarde van het attribuut multiple ook worden weggelaten, maar de webstandaards schrijven voor attributen altijd een waarde te geven.

d De body van de pagina display_music_choices.jsp ziet er als volgt uit:

```
<body>
<!-- import packages and classes needed by the scripts -->

<%
  // get parameters from the request
  String firstName = request.getParameter("firstName");
  String lastName = request.getParameter("lastName");
  String[] types =
      request.getParameterValues("musicTypes");
%>
<h1>Thank you for joining our email list, <%= firstName %>
<%= lastName %>.</h1>
<p>
We'll use email to notify you whenever we have new releases
for these types of music:
</p><p>
<%
  for (String type: types) {
%>
    <%= type %><br>
<%
  }
%>
</p></body>
</html>
```

De waarden die zijn gekozen uit de list box worden allemaal met dezelfde parameternaam doorgegeven. Selecteert de gebruiker bijvoorbeeld de waarden Rock en Bluegrass, dan ziet het bijbehorende deel van de URL er als volgt uit:

musicTypes=Rock&musicTypes=Bluegrass

4.10 De webapplicatie bestaat uit een Java-klasse, een HTML-pagina en een JSP-pagina. We tonen van alle drie de relevante delen, zonder het commentaar. Bij de uitwerkingen op Studienet kunt u de volledige uitwerking vinden.

Ten eerste tonen we een mogelijke uitwerking van de domeinklasse CaesarCodering. De klasse heeft statische methoden codeer en formatteer.

```
package domein;

public class CaesarCodering {
  private static final String alfabet =
                              "abcdefghijklmnopqrstuvwxyz";

  public static String codeer(String bericht, int sleutel) {
    bericht = bericht.toLowerCase();
    String code = "";
    for (int c=0; c < bericht.length(); c++) {
      int indexOfC = alfabet.indexOf(bericht.charAt(c));
      if (indexOfC != -1) {
        int indexVerschoven = (indexOfC + sleutel) % 26;
        code = code + alfabet.charAt(indexVerschoven);
      }
    }
    return code;
  }

  public static String formatteer(String code) {
    String geformatteerd = "";
    for (int c=0; c < code.length(); c++) {
      geformatteerd = geformatteerd + code.charAt(c);
      if ((c + 1) % 5 == 0) {
        geformatteerd = geformatteerd + " ";
      }
    }
    return geformatteerd;
  }
}
```

Ten tweede tonen we de HTML-pagina.

```
<html>
<head>
<title> Bericht coderen met Caesarcodering</title>
</head>
<body>
<h1>Bericht coderen met Caesarcodering</h1>
```

```
<p>
Met deze pagina kunt u een bericht coderen met behulp van de
Caesarcodering, die elke letter in een bericht een gegeven
aantal posities opschuift (dit aantal is de sleutel).
Hoofdletters worden voor coderen omgezet naar kleine
letters. Cijfers, spaties en leestekens worden weggelaten.
</p>
<form action="gecodeerd.jsp" method="post">
  <table>
    <tr>
      <td width="100px">Sleutel:</td>
      <td><input type="text" name="sleutel" size="2"></td>
    </tr>
    <tr>
      <td>Bericht:</td>
      <td>
        <textarea name="bericht" rows="5" cols="60">
        </textarea>
      </td>
    </tr>
    <tr>
      <td><input type="submit" value="Verzenden"></td>
  </table>
</form>
</body>
</html>
```

Merk op dat als methode POST is gebruikt en niet GET. Bij het gebruik van GET verschijnt de oorspronkelijke tekst in de URL; bij codering ligt dat niet voor de hand. Ook is het denkbaar dat de invoer te lang wordt.

Tot slot tonen we de pagina gecodeerd.jsp. Merk op dat de klasse CaesarCodering geïmporteerd is met behulp van het page directief.

```
<html>
<head>
<title>Bericht gecodeerd met Caesarcodering</title>
</head>
<body>
<%@ page import="domein.CaesarCodering" %>
<%
  int sleutel =
    Integer.parseInt(request.getParameter("sleutel"));
  String bericht = request.getParameter("bericht");
  String code = CaesarCodering.codeer(bericht, sleutel);
%>
<table cellspacing="10">
  <tr>
    <td width="10%"><b>Bericht:</b></td>
    <td width="80%"><%= bericht %></td>
  </tr>
  <tr>
    <td><b>Versleuteld:</b></td>
    <td><%= CaesarCodering.formatteer(code) %></td>
  </tr>
</table>
</body>
</html>
```

2 Uitwerking van de zelftoets

1 a Incorrect. Import-opdrachten kunnen niet voorkomen binnen scriptlets.
b Correct, mits naam een gedeclareerde variabele is van type String. Omdat het hier een scriptlet betreft en geen expressie, wordt geen uitvoer gegenereerd, maar dat hoeft niet per se.
c Correct. Een JSP-expressie is alles wat naar String geconverteerd kan worden; een losse constante mag dus ook.
d Incorrect. De puntkomma aan het eind moet weg.
e Incorrect. De juiste vorm is

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

f Correct.
g Incorrect. JSP-tagparen kunnen niet genest worden; een JSP-expressie kan dus geen onderdeel uitmaken van een scriptlet.

2 De uitvoer bestaat uit een lijst van alle maanden waar een r in zit, dus

```
januari
februari
maart
april
september
oktober
november
december
```

3 a Correct. Zie MS5_p150-154.
b Incorrect. Een JSP-pagina wordt vanuit een anchor-tag juist altijd met de get-methode opgevraagd. Zie MS5_p150,151.
c Incorrect. Zie MS4_p150,151.

4 a Het eerste JSP-fragment toont een declaratie van een attribuut. Dit attribuut wordt gedeeld door alle draden die gestart worden bij het opvragen van de pagina.
Het tweede JSP-fragment toont een gewone scriptlet. Elke draad heeft daarom een eigen variabele aankoopbedrag en elk aankoopbedrag wordt opgeteld bij de ene totaalomzet.
b Er zijn twee redenen waarom dit niet goed werkt.
– Het optellen van aankoopbedrag bij totaalOmzet is niet threadsafe; verschillende optellingen in verschillende draden kunnen elkaar voor de voeten lopen waardoor sommige aankoopbedragen verloren kunnen gaan. Zie MS4_p164,165.
– Tenzij de waarde van de variabele bij het afsluiten van de server ergens wordt opgeslagen, houdt deze alleen de omzet bij sinds de laatste keer dat de server is herstart. Bij het opnieuw starten van de server wordt de variabele immers weer op 0 gezet.