Webontwikkeling 2

WEEK 1 // CLIENT-SERVER

1. *Wat is het client-server model?*

Het client-servermodel beschrijft de relatie tussen twee computerprogramma's in een proces. Deze relatie is asymmetrisch. De cliënt verstuurt een service verzoek aan een ander programma, de server. De server verwerkt het verzoek en stuurt een response. Hoewel het client-server principe kan worden gebruikt door programma's binnen een enkele computer, is het belangrijker binnen een netwerk.

2. *Wat is het verschil tussen de rol van de client en de server in het client-servermodel?*

CLIENT	SERVER
Heeft service nodig	Biedt een service aan
Draait een client-programma	Draait een server-programma
Deelt geen resources	Deelt resources met de client
Begint de communicatie	Wacht op vraag
Vraagt (request)	Antwoordt (response)

3. *Wat is het http-protocol?*

HTTP staat voor Hypertext Transfer Protocol en is een protocol dat de communicatie over het web beschrijft.

1) Hoe boodschappen gevorm en overgebracht worden
2) Hoe browsers en web servers met deze boodschappen moeten omgaan

Er zijn verschillende soorten boodschappen mogelijk (GET, POST, PUT, DELETE, HEAD, ...)

4. *Waarvoor mag je een HTTP GET request gebruiken?*

Een HTTP GET-request wordt gebruikt om pagina's of objecten op te vragen via URL. Dit gebeurt wanneer je op een link klikt, of zelf deze URL ingeeft in de adresbalk van je browser.

5. *Hoe is een GET request opgebouwd?*

Een GET-request bestaat uit een request-line en enkele header-lijnen. De request-lijn is de eigenlijke request en de header-lijnen zijn extra informatie. Het einde van een GET-request wordt aangeduid met een lege lijn (CRLF = Carriage Return Line Feed).

De request-line bestaat uit het soort boodschap, het pad van de pagina op de server en de HTTP-versie.

6. *Hoe is de URL van een GET request opgebouwd?*

http://localhost:8080/Web_1_Servlet/index.html

Een URL bestaat is opgebouwd uit http://, dat het protocl aanduid, gevolgd door de hostnaam en eventueel een poortnummer. Achter de host bevindt zich het pad van de pagina. Host en path worden onderscheiden met een slash (/). Achter het path worden de parameters meegegeven in een querystring. Een querystring bestaat uit key-valueparen (key=value). Verschillende key-valueparen worden onderscheiden met een ampersand (&). Path en querystring zijn onderscheiden met een vraagteken (?). De host wordt ook als headerregel meegegeven in de GET request.

7. *Hoe is een GET response opgebouwd?*

De GET-response bestaat uit een responseregel die de eigenlijk response is. De responseregel bevat een statuscode. Deze toont of het ophalen van de data gelukt is of welke errors er opgetreden zijn. Na de responseregel zijn er net zoals in de GET request verschillende headerregels mogelijk. Na deze headerregel volgt weer een lege lijn (dubbele CRLF). Onder deze witte lijn wordt de body van het ogevraagde object getoond.

Enkele HTTP error codes:
1xx – Informational : request is ontvangen en het proces gaat door
2xx – Succes : request ontvangen, begrepen, geaccepteerd en succesvol verwerkt
3xx – Redirection : de client moet bijkomende actie ondernemen om de request af te werken
4xx – Client error : een foutboodschap die door de client veroorzaakt is, zoals het verkeerd typen van een URL
5xx – Server error : Fout opgetreden aan serverkant

8. *Wat is een WAR file?*

Een WAR-file is een Web Application Archive file. Het is een gecomprimeerd bestand met daarin de gecomprimeerde java klassen (.class files), libraries en webbestanden die in de juiste folderstructuur gezet zijn.

Namelijk een root die een WEB-INF folder bevat en de verschillende webbestanden. WEB-INF zie een van volgende vragen.

9. *Wat is het verschil tussen de stappen build en deploy?*

Build betekent het project in de IDE opslaan als war-file. De javaklassen worden gecompiled en alle bestanden en libraries worden in de juiste folderstructuur geplaatst. Deze folderstructuur wordt gecomprimeerd tot een war-file.

Deployen is de war-file in de juiste folder van de web container zetten (tomcat: webapps). De war-file wordt gedecomprimeer/geunzipped op deze locatie. In de tomcatmanager browse | deploy.

10. *Waarvoor dient de folder WEB-INF?*

In de WEB-INF folder zitten alle bestanden die niet beschikbaar zijn voor het publiek van de web-applicatie. Alle publiekelijke bestanden zitten gewoon in de root van de webapplicatie, van de WAR file. Deze WEB-INF folder bevat ook een web.xml bestand en een lib-folder die de libraries bijhoudt.

11. *Wat is de betekenis van een HTTP error code die begint met een 2?*

Deze error code betekent succes. Dit betekent dat de request correct is aangekomen, begrepen en geaccepteerd is door de server. Daarna verwerkt geweest is door de server.

WEEK 2 // SERVLET

1. *Wat is het verschil tussen een HTTP server en web container?*

Wanneer dynamische content gegenereerd moet worden volstaat een HTTP server niet meer. Er is dan een web container nodig die met Java kan werken (of PHP). De HTTP server zit in deze web container. Wanneer een HTTP request door de client wordt doorgestuurd, zal de HTTP server aan de Java klasse vragen om de dynamische HTML te genereren. De javaklasse geef deze terug aan de HTTP server die een HTTP responsebericht zal maken en deze naar de client sturen. Browser verwerkt dit responsebericht, haalt data uit dit responsebericht en toont de pagina.

De HTTP server antwoordt op HTTP requests en geeft HTML pagina's terug. Een webcontainer kan dit ook, maar kan ook met Java of Php werken. Er is een klasse in de webcontainer die HTML kan genereren, dit wordt de servlet genoemd.

2. *Hoe noem je de Java klasse die, wanneer er een request binnenkomt door de web container aangeroepen wordt o dynamische HTML te genereren?*

Deze wordt de servlet genoemd en wordt door de webcontainer herkend en beheerd. Het is een gewone java klasse.

3. Wat is het verschil tussen een gewone Java klasse en een servlet?

Een servlet is ook een gewone java klasse. Deze klasse wordt gebruikt om dynamische HTML te genereren. Deze klasse extends HttpServlet. Deze klasse wordt door de container uitgevoerd. Deze klasse bevat init(), doPost(), doGet(), destroy() en service() methodes. De doPost() en doGet() zijn nog leeg en moeten dus geschreven worden in de servlet. De init() methode wordt eenmalig aangeroepen bij het aanmaken van de servlet. Service() wordt door de webcontainer aangeroepen telkens wanneer de servlet wordt aangeroepen en roept doGet() of doPost() op, telkens in een andere thread.

Bij het opstarten van de webcontainer wordt het servlet object aangemaakt, eerst met constructor en dan door init().

4. *Hoe weet de web container welke servlet hij moet aanspreken wanneer een request binnenkomt?*

Vroeger via web.xml. Nu worden annotations gebruikt. Alle javaklassen worden overlopen. Wanneer de container @Webservlet ("/Dice") tegenkomt . De container gaat kijken of de URL bij de annotatie overeenkomt met de URL uit de request.

5. *In welke methode moet ik mijn Java code zetten om het antwoord op een GET request te genereren?*

Deze moet in de doGet() komen. De webcontainer maakt een HttpServletRequest object met de gegevens van de request en passed deze door samen met een leeg HttpServletResponse object aan de doGet() methode.

WEEK 3 // JSP

1. *Waar staat de afkorting JSP voor?*

Java Server Pages, het zijn html-pagina's met Java code in.

2. *Waarom zijn de servlets (deels) vervangen door JSP?*

HTML in Java klassen schrijven zorgt voor onleesbare code. Bovendien is er geen ondersteuning voor de validatie van de HTML code.

3. *Wat is het verschil tussen een JSP –en een HTML-pagina?*

JSP-pagina's lijken sterk op HTML pagina's. JSP-pagina heeft bovenaan enkele JSP directives. Dit is info voor de compiler. Deze directive worden aangeduid met een @ in de scriplets. Java code kan geschreven worden tussen JSP scriplets (<% %>). Deze scriplets hebben toegang tot enkele globale variabelen zoals out.

4. *Toon de inhoud van een variabele message (String) in een JSP pagina met een gewone JSP scriplet en een JSP expression.*

Een JSP expression wordt gebruikt om een waarde te tonen als String. Hierdoor geen gebruik van out.println() meer nodig.

5. *Hoe herken je een JSP directive? Waarvoor dient het?*

Een JSP directive is te herkennen aan de @ die tussen de scriplets komt te staan, net na de eerste %. De JSP directives bevatten extra informatie voor de compiler. Het geeft weer welke klasse geimporteerd moeten worden en welke jspf (jsp fragments) geïnclude moeten worden. (page voor import klassen, include voor JSPF). JSPF is voor het hergebruiken van JSP of HTML code. De directive met de include regel komt op de plaats waar je de stukken JSP of HTML wilt laten zien. Een JSP directive is een element dat relay messages stuurt naar de JSP container die de manier waarop JSP-page gecompiled wordt, beïnvloedt.

6. *Wat is de overeenkomst tussen JSP en een servlet?*

De JSP pagina wordt door de Web container gebruikt om een servlet te genereren. Achter de schermen is de werking dus nog steeds zo dat een HTTP-request aankomt en dat de web container hiervoor een request en response object voor klaarzet. Een thread kiest of maakt voor in de servlet te runnen, de service methode wordt opgeroepen van de servlet. Het enige verschil is hier dat er van de JSP pagina een servlet gemaakt wordt.

7. *Is JSP client-side of server-side? Leg uit*.

JSP is server-side. De JSP container op de web server gaat de JSP pagina omzetten naar een servlet. De gebruik merkt hier niets van.

WEEK 4 // GET MET PARAMETERS

1. *Hoe geef je parameters mee met een GET request?*

Met behulp van een querystring kunnen verschillende parameters meegegeven worden in de vorm van een key-valuepaar, die van elkaar onderscheden worden met een ampersand (&). Tussen het pad en de parameters wordt een vraagteken geplaatst. Bij een POST-request zitten de parameters in de body van het verzoekbericht en dus niet zichtbaar voor de gebruiker.

2. *Als ik de naam John en de lievelingskleur Groen wil meegeven met een GET request, hoe ziet de querystring er dan uit?*

http://localhost:8080/WebApplicatie/enquete.jsp?kleur=groen&naam=john

3. *Hoe kan ik in mijn servlet de waarde van de parameter name opvragen?*

Van een JSP wordt een servlet gemaakt. We weten ook dat voor een HTTP-request, de container een request en een respone object aanmaakt. In dit request object worden de

parameters die meegegeven zijn in de URL ook opgenomen. Deze kunnen via het request object in JSP opgehaald worden. Request.getParameter(“key”).

Een JSP declaratie is een scriplet met een !. Dit betekent dat deze code niet veranderd en dus gelijk is voor alle requests, het is een instantievariabele van de servlet. Deze wordt gemaakt bij het aanmaken van de servlet. Er is maar 1 servlet, dus deze zal niet veraneren. De webcontainer maakt van de JSP pagina een servlet (eenmalig). De servlet kan voor een request, de html code in de servlet wegschrijven naar een responseobject. Dit responseobject wordt omgezet door container naar HTTP antwoordbericht dat verstuurd kan wordne naar de client.

4. *Hoe kan ik in mijn JSP de naam en het aantal inwoners van een country-object opvragen?*

Wanneer het country-object in de server als object wordt meegegeven via request.setAttribute(“country”, country), kan het via Integer.parseIn(request.getAttribute(“country”).getAantalInwoners()) worden opgeroepen.

5. *Noem de 2 belangrijkste verschillen tussen een request parameter en een attribuut.*

Men wil zo weinig mogelijk Java in JSP pagina’s schrijven. Daarom wordt Java zo veel mogelijk in een zelfgeschreven servlet gedaan. In deze servlet mag geen HTML geschreven worden. De servlet moet de request en response doorsturen naar de opggeven jsp (zie RequestDispatcher). De servlet doet iets met de parameters die meegegeven zijn en genereert zelf een resultaat. Dit resultaat wil je ook meegeven naar de JSP pagina. Parameters toevoegen in de servlet aan het request object gaat niet, dus gebruiken we attributen.

Parameters worden door de client meeggeven. Attributen kunnen vanuit de code in de servlet worden meegegeven. Attributen kunnen niet alleen strings zijn, maar ellerlei objecten.

De URL zal niet meer Guess.jsp bevatten maar de mappingnaam van de servlet. JSP-pagina’s mogen niet meer direct aangeroepen worden.

WEEK 5 // MODEL-VIEW-CONTROLLER

1. *Hoort volgende regel in model, view of controller?*
   a) <h1><%= country.name %></h>
      View (JSP, pagina)
   b) double precentageOfWorldPopulation = getNumberOfInhabitants*100/722;
      Model
   c) request.setAttribute(“country”, country);
      Controller

2. *Wat is het doel van het MVC patroon?*

Het Model-View-Controller patroon laat toe om de businesslogica van de UI te onderscheiden en laat zo toe om elk van deze delen onafhankelijk te ontwikkelen. De model kan dus onafhankelijk ontwikkelt en herbruikt worden elders.

3. *Benoem elk onderdeel van het MVC patroon en leg uit.*

Model – Alle data, toestand en businesslogica
View – Visuele representatie van het model
Controller – reageert op input van gebruikers en stuurt de communicatie tussen view en model

Het voordeel van MVC is dat de klassen die de logica bevatten, de model klassen, herbruikt kunnen worden in andere vormen, andere applicaties.

4. *Vul een gegeven workflow in.*

WEEK 6 // FORMS

Forms zorgen ervoor dat bij user-input de URL niet handmatig opgesteld moet worden.

1. *Wat is het verschil tussen het attribuut id en het attribuut name bij een HTML input element?*

Het attribuut id wordt gebruikt voor verwijzingen vanuit code. Name daarentegen is de naam, de key, van de parameter die zal meeggeven worden. Wanneer labels gebruikt worden die niet genest zijn, wordt met de for-attribuut naar het input element verwezen. Hierbij moet for=id. Men kan het label ook gewoon nesten.

2. *Waarvoor dient het attribuut type bij een HTML input element?*

Er zijn allerlei soorten inputvelden voor een formulier. Het type specifeert welk soort inputveld het is, kleur, radiobuttons, checkboxes, tekst, password,...

3. *Als ik een form submit, wordt er dan een HTTP GET of een POST uitgevoerd, hoe komt dit?*

Standaard wordt er een GET request uitgevoerd wanneer niets wordt meegegeven. De method kan als attribuut worden meegegeven in de <form>. De parameters worden bij GET in de URL meegegeven. Bij POST in de body van HTTP verzoekbericht.

4. Waarheen wordt volgend formulier gestuurd:

<form method=”POST” action=””>?

Het formulier, wanneer gesubmit, wordt naar de zelfde pagina gestuurd. Er wordt dus niets doorgestuurd.

5. *Verklaar verschil tussen volgende lijnen:*
    a. *<form method=”POST” action=””>*
    Er gebeurd niets, komt op zelfde pagina terecht.

    b. *<form method=”POST” action=”Course”>*
    De form wordt doorgestuurd naar de Course-servlet, die indien doPost() geschreven is, dit zal afhandelen.

    c. *<form method=”POST” action=”result.jsp”>*
    Gaat navigeren naar result.jsp, maar aangezien in doPost() in servlet niet geweest is, gaan attributes niet geset zijn en gaan de getAttributes null geven. In sommige gevallen kan dat tot exceptions leiden zoals nullpointerexceptions.

    d. *<form method=”GET” action=”result.jsp”>*
    Zelfde als bij POST maar dan kunnen de parameters gezien worden in URL (querystring).

6. *Hoe worden de parameters meegegeven in een POST request?*

In een POST-request komen de parameters in de body te staan van het HTTP request. Deze zijn dus niet zichtbaar in de URL.

7. *Wanneer gebruik je een GET en wanneer een POST request?*

GET :
- Ophalen data, navigaite
- GEEN side-effect (= Als het de state modified van een ander element)
- IDEMPOTENT, dat wil zeggen, dat het niet uit maakt of je HTTP request 1 of 100 keer stuurt, de request zal altijd opnieuw volledig behandelt worden

POST bij:
- Wel side-effect
- Toevoegen, verwijderen, wijzigen

8. *Noem 3 voordelen van een POST request.*

Er is geen restrictie op lengt en type. Goed voor security. En er is geen visibility va user input.

9. *Noem 3 nadelen van een POST request.*

Er is geen vorm van bookmark, cache. Parameters worden niet in de history bewaard. Er is geen backbutton.

10. *Leg uit: een GET request moet idempotent zijn.*

Idempotent wil zeggen dat het niet mag uitmaken hoeveel keer een HTTP request wordt gestuurd via GET methode. Elke request moet afzonderlijk worden afgehandeld, helemaal opnieuw. Mag dus geen state gaan bijhouden van de vorige requests. Deze afzonderlijke requests moeten telkens zelfde resultaat geven.

WEEK 7 // SELENIUM

1. *Je kunt Selenium gebruiken vanuit een gewone Java klasse of vanuit een JUnit tesklasse. Wat is het voordeel van de JUnit testklasse?*

Het is minder omslachtig om te controleren of de bekomend output juist is. Dit kan met assert. Bovendien meerdere testen in een testklasse tegenover slechts 1 in een main method van een gewone klasse. Bovendien krijg je automatisch een testrapport waarop men kan zien of testen slagen of niet (dus gewenst resultaat).

2. *Waartoe dient Selenium?*

Selenium laat toe om automatische te testen, ipv zelf altijd alles handmatig te moeten testen. Zeker bij formulieren invullen is dit een grote tijdsbesparing. Selenium automatiseert de browser en bootst acties van een echte gebruiker na.

3. *Wat is het grote voordel van Selenium testklasse maken?*

Automatisch testen, veel tijdsbesparing.

4. *Je wil de werking van een formulier testen met Selenium, wat test je allemaal?*

Belangrijk om te kijken of wanneer foute input is, deze juist wordt afgehandeld. Hierbij is het belangrijk om randwaarden te testen. Test ook een juiste en foute waarde.

WEEK 8 // FRONTCONTROLLER

Stel dat er gewenst is om meerdere functies te implementeren die gebruik maken van dezelfde HTTP methode (GET of POST). In deze situatie moeten ofwel aparte servlets gemaakt worden voor de verschillende functies ofwel moet gebruik gemaakt worden van een Frontcontrollerpatroon (betere oplossing). Bij het frontcontrollerpatroon wordt een attribuut action meeggeven in de form tag. In de servlet (de enige) wordt gecontroleerd wat de servlet moet doen via deze parameter die wordt meegegeven.

Vb. action=”Country?action=create”
Een front controller zorgt voor een ‘single point of entry’. Alle requests worden in eenzelfde servlet afgehandeld. De frontcontroller delegeert de requests over verschillende methodes voor verschillende actions. Alle GET requests en POST requests komen samen in processRequest() waar de delegatie gebeurt.
Meerdere servlets zorgt ervoor dat de view te veel moet weten over de controllers, nl welke ze moet aanroepen.

WEEK 9 // BETTERFORMS

Validation kan client-side door middel van required bij inputvelden te vermelden. However, kunnen ze bij GET requests nogs teeds gewoon in de URL aanpassen, waardoor foute informatie kan worden opgeslagen of errors optreden. Het is dus belangrijk dat ook server-side gevalideerd wordt, en dat deze validatie visueel getoond wordt aan de gebruiker. Dit doen we door aan te geven aan de gebruiker welke velden juist waren, door ze te laten staan en kleuren te geven. Foute velden worden vermeld en krijgen een rode kleur.

1. *What does JSPF stand for? What is the difference JSP?*

JSP Fragments

2. *How can you reuse parts of HTML?*

Met JSP-fragments kunnen delen herbruikt worden. Dit wordt aangeduid met behulp van een include directive <% @ include file = “header.jspf” %>. De JSP container zal de fragmenten toevoegen aan de JSP page. De JSP pagina wordt daarna gecompileerd tot 1 servlet. Dit noemt men ook wel eens statisch.

3. *When we add validation to a form: what do you have to add in Model, View, Controller?*

Serverside validatie gebeurt door in model exceptions te gooien op de juiste plaats, deze exceptions op te vangen in controller en de informatie door te spelen aan de view. De view moet deze informatie dan tonen.

WEEK 10 // USABILITY OF FORMS

1. Evaluate (a screenshot of) an existing webform
    - What is good from the usability point of view
    - What can be better

Kijk uit naar :
a) Validatie bovenaan

b) Validatie bij elk invulveld
c) Errormessages bij elk invulveld
d) Aanduiden wat verplicht is in label

Relation – dingen pas zichtbaar maken in forms als ze van belang zijn

WEEK 11 // JDBC

1. *What does JDBC stand for?*

JDBC staat voor Java Database Connectivity, het is een Java API (set van klassen die gebruikt kunnen worden). Het laat toe om een Java programma met een database te communiceren via SQL (Standard Query Language).

2. *What is the advantage of JDBC?*

JDBC kan met verschillende relationele databases communiceren, spreadsheets, ...

3. *Is JDBC used in view, model of controller?*

JDBC wordt gebruikt in model.

Connection = sessie met database
Statement = Object om SQL instructions mee uit te voeren
Execute instruction = SQL Query, ofwel statement.executeUpdate("query") voor inserts, updates, deletes die het aantal beïnvloede rijen teruggeeft, anderzijds, statement.executeQuery("query") om data op te halen (select) en geeft een resultset (part of data) terug.

Connection connection = DriverManager.getConnection("url", properties);
Statement statement = connection.createStatement();
*ResultSet result = statement.executeQuery("SELECT * FROM myTable");*
String name = result.getString("Name"):
MyObject object = new MyObject(name);

URL = jdbc:postgresql://gegevensbanken.khleuven.be:51415/1TX6
= <vendor>://<server host>:<port>/<dbname>
Properties = userid, password, ssl=true, sslfactory: org.postgresql.NonValidatingFactory