Web 3: Theorievragen

No Scriptlets

What is the advantage of using expression language instead of JSP scriptlets and JSP expressions?

Geen javacode tussen de html.

What is the difference between the . operator and the [ ] ?

. is enkel voor Javabean of Map.

[] is voor Javabean, Map, Array, List…

What does JSTL stand for?

JSP Standard Tag Library.

Name the 6 language constructs of JSP.

Scriplet:	`<% List<Country> countries = (List)request.getAttribute("countries"); %>`
Expression:	`<%= country.getName() %>`
Directive:	`<%@ include file="header.jsp" %>`
Declaration:	`<%! int counter = 0; %>`
EL:	`${ country.name }`
Action:	`<jsp:include page="header.jsp"/>`

Is JSTL used in the View, Model or Controller?

View.

When is a Java class a JavaBean?

- Een public constructor zonder argumenten.
- Een getter en een setter definiëren een property.

JDBC

What does JDBC stand for?*

Java DataBase Connectivity

What is the advantage of JDBC?

- Write once, run anywhere.
- De data verdwijnt niet als de connectie gesloten wordt.

Is JDBC used in View, Model or Controller?

Model.

XSS

Explain XSS. Use an example and describe in details the solution in order to avoid XSS.

Er kan kwaadaardige code ingevoerd worden in een website, meestal met een script in de browser.

```
<input … value=“”/><script>alert(“lol”);</script>” />
```

Oplossing: html-symbolen omzetten naar html-entiteiten met de <c:out> tag.

```
<input … value=“<c:out value='${param.username}' />” />
```

What does XSS stand for?

Cross Site Scripting

Hashing

Explain why we need to hash our passwords?

Zodat de wachtwoorden niet in plain text in de database staan.

What is the reason to add a salt before hashing a password?

Om het extra moeilijk te maken om een dictionary attack te doen op de database. Hierbij worden er miljoenen mogelijkheden vergeleken met de inhoud van de database. Deze mogelijkheden zijn dan bestaande woorden zoals de woorden in een woordenboek.

SQL Injection

What is SQL injection?

Er wordt kwaadaardige code geïnjecteerd in een veld om gevoelige data te bemachtigen of om de database aan te passen.

Give an example how SQL Injection work?

```
SELECT * FROM person
  WHERE email=‘’ OR 1=1 OR ‘1’=‘1’;
```

What is the solution we saw for SQL Injection?

Prepared statements, hierbij staan er placeholders in plaats van waardes.

What are the advantages of prepared statements.

- Sneller.
- Duidelijkere syntax.
- De structuur van de query ligt vast en is dus veilig.

Name 4 risks we discussed while creating websites.

- Gevoelige informatie in GET.
- Wachtwoorden in plain text.
- XSS.
- SQL Injection.

Cookies

Is a cookie saved at the client side or at the server side?

Client side.

Explain how a cookie works.

De server maakt een cookie en geeft dit aan de client in de response header. De browser stuurt dit cookie dan mee met elke request header.

How can you set the expiration time of a cookie?

```
cookie.setMaxAge(24*60*60) // 24 hour
cookie.setMaxAge(0) // immediately
cookie.setMaxAge(-1) // when browser closes
```

Is a cookie safe? Why or why not?

Nee, de cookies kunnen aangepast worden.

Sessions

Is a session saved at the client side or at the server side?

Server side.

Explain how sessions works.

De server maakt een sessie object aan, slaat de informatie op in de sessie en stuurt het id van de sessie naar de client in de response header. De browser stuurt het id mee met elke request header. De server gebruikt dit id om de bijhorende sessie te vinden en de server krijgt de informatie van de sessie.

How can you set the expiration time of a session?

```
session.setMaxInactiveInterval(seconds)
```

web.xml

server default

How can you end a session?

```
session.invalidate()
```

Is a session safe? Why or why not?

Nee, session hijacking is mogelijk omdat enkel het id beschikbaar is.

When is it better to use a session, when is it better to use a cookie?

- Session: veiliger en meer data mogelijk.
- Cookies: enkel gebruiken wanneer er weinig data moet verstuurd worden.

Authorization

Enum met roles.

Configuration

Where do you store properties in a web application?

In een configuration file (web.xml).

What is the ServletContext?

1 instantie per applicatie die meta-informatie over de webapplicatie bevat en methodes heeft om te communiceren met de container.

When should you use an error page?

Wanneer er zich een error voordoet waar de gebruiker niets aan kan doen.

What makes an error page different from an normal JSP page?

Wanneer er een exception gegooid wordt die niet opgevangen wordt, wordt er automatisch naar de error page genavigeerd. Dit moet ook zo aangegeven worden in de web.xml.

How long should you keep your database connection open for a web application? Why?

Totdat de methode is uitgevoerd. Op deze manier is de applicatie schaalbaar wanneer het door veel gebruikers tegelijk gebruikt wordt.

What is the advantage of reading the properties in a ContextListener?

Dezelfde properties moeten niet elke keer opnieuw aangemaakt worden (DRY).

What is the first method that will be executed when your application is deployed?

init()

What is the last method that will be executed when your application is stopped?

destroy()

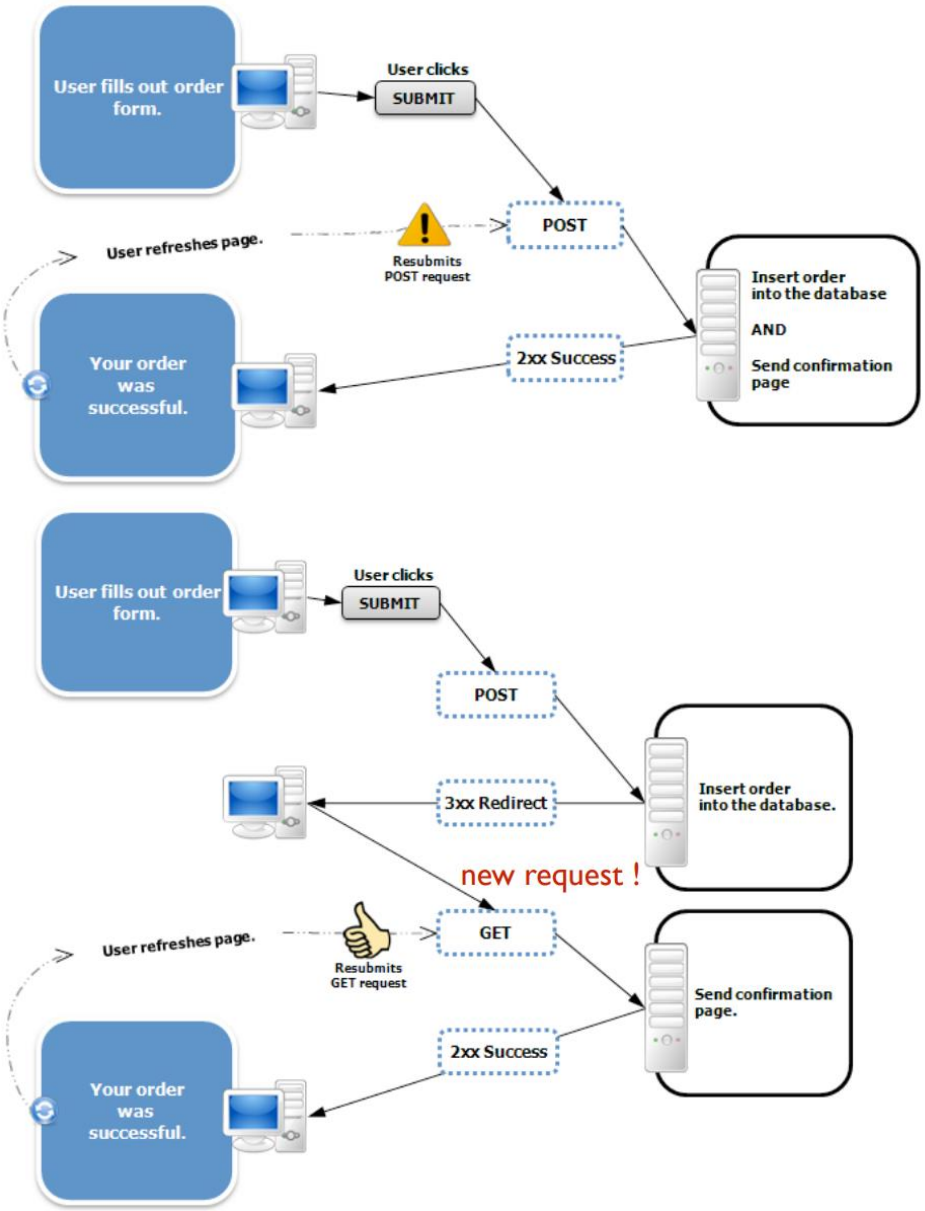
Post/Redirect/Get

Leg uit wat het probleem is dat Post/Redirect/Get oplost?

Wanneer de user de pagina refresht waar hij net een POST-request heeft naar gestuurd, kan hij de POST-request nog eens opsturen. Op deze manier kan bijvoorbeeld een bestelling per ongeluk twee keer geplaatst worden.

```
request.getRequestDispatcher("index.html").forward(request, response);
```

Leg de figuren uit.



Hoe moet je Post/Redirect/Get implementeren?

De request wordt niet geforward, maar de response wordt geredirect.

```
response.sendRedirect("Controller?action=welcome");
```

Wat is het verschil tussen forward en sendRedirect?

request.forward()

- Wordt server side uitgevoerd.
- Kan binnen de server gebruikt worden.
- Informatie in de request blijft beschikbaar.

response.sendRedirect()

- Wordt client side uitgevoerd.

- Kan binnen en buiten de server gebruikt worden.
- Informatie in de request gaat verloren.

Front Controller Continued

Leg het Front Controller patroon uit.

De front controller is een 'Single Point of Entry' dat de requests behandelt. Het zorgt voor de business processing en de keuze van de juiste view.

Verschillende stukjes code kunnen uitleggen van de verschillende opties en hun voor- en nadelen kunnen uitleggen.

```
protected void processRequest(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
    try {
        String action = request.getParameter("action");
        RequestHandler handler =
                            handlerFactory.getHandler(action, service);
        String destination = handler.handleRequest(request, response);
        RequestDispatcher view = request.getRequestDispatcher(destination);
        view.forward(request, response);
    } catch (Exception e) {
       throw new ServletException(e.getMessage(), e);
    }
}
```

Opties:

- Simple Factory
  - Code niet 'closed for modification'
  + Geïsoleerd

```
public class HandlerFactory {

    public RequestHandler getHandler(String action,
                                     ShopService service) {
        RequestHandler handler = null;

        if (action.equals("overview")) {
            handler = new PersonOverviewHandler(service);
        } else if (action.equals("signUp")) {
            handler = new SignUpHandler();
        } else if (action.equals("confirmSignup")) {
           handler = new ConfirmSignupHandler(service);
        } else if (action.equals("…")) {
            handler = new … Handler();
        }
    }
}
```

```
public class HandlerFactory {
    private Map<String, RequestHandler> handlers = new HashMap<>();

    public HandlerFactory(ShopService service) {
        handlers.put("overview", new PersonOverviewHandler(service));
        handlers.put("signUp", new SignUpHandler());
        handlers.put("confirmSignup", new ConfirmSignupHandler(service));
        …;
   }

    public RequestHandler getHandler(String key) {
        return handlers.get(key);
    }
}
```

- Factory met reflection
  + Code is ‘closed for modification’
  - De naam van de action moet overeenkomen met de naam van de handler

```
public class HandlerFactory {

   private RequestHandler getHandler(String handlerName, ShopService model)
         throws ServiceException {
      RequestHandler handler = null;
      try {
         Class handlerClass = Class.forName("controller."+ handlerName);
         Object handlerObject = handlerClass.newInstance();
         handler = (RequestHandler) handlerObject;
         handler.setModel(model);
      } catch (ClassNotFoundException e) {
         throw new ServiceException(e);
      }

      return handler;
   }
}
```

- Factory met reflection en configuration file
  + Code is ‘closed for modification’
  + De naam van de actie moet niet overeenkomen met de naam van de handler
  - Veel XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <entry key="userOverview">controller.handler.UserOverviewHandler</entry>
    <entry key="confirmSignup">controller.handler.ConfirmSignupHandler</entry>
    <entry key="signUp">controller.handler.SignUpHandler</entry>
     …
</properties>
```

```
private HandlerFactory handlerFactory;

public void init() throws ServletException {

     super.init();
     …
     try {
         ShopService service = new ShopService(…);

         InputStream input = context.getResourceAsStream(“/WEB-INF/handler.xml");
         Properties properties = new Properties();
         properties.loadFromXML(input);

         handlerFactory = new HandlerFactory(properties, service);
      } catch (Exception ex) {
           …
      }
}

public class HandlerFactory {
     private Map<String, RequestHandler> handlers = new HashMap<>();

     public HandlerFactory(Properties handlers, ShopService model) {
       for(Object key : handlers.keySet()) {
          RequestHandler handler = null;
          String handlerName = controllers.getProperty((String) key);
          try {
             Class<?> handlerClass = Class.forName(handlerName);
             Object handlerObject = handlerClass.newInstance();
             handler = (RequestHandler) handlerObject;
          } catch (ClassNotFoundException e) {
             …
          }
          handler.setModel(model);
          handlers.put(key, handler);
     }

     public RequestHandler getHandler(String key) {
         return handlers.get(key);
     }
}
```

Controller:

```
protected void processRequest(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
     try {
         String action = request.getParameter("action");
         RequestHandler handler = handlerFactory.getHandler(action, service);
         handler.handleRequest(request, response);
     } catch (Exception e) {
        throw new ServletException(e.getMessage(), e);
     }
}
```

Common Problems

Explain: application scoped vs. request scoped database connection.

- Application scoped: er is slechts één gebruiker die van de applicatie gebruik maakt (desktop applicatie).
- Request scoped: er zijn meerdere gebruikers die tegelijk van de applicatie gebruik maken (web applicatie).

How long should you keep your database connection open for a web application? Why?

Vanaf het aanmaken van de request tot de request gesloten wordt. Dit is de enige manier om de applicatie schaalbaar te maken wanneer er veel verschillende gebruikers tegelijk gebruik van maken.

Where do you open a connection?

In de repository class in het begin van elke CRUD methode.

Where do you close a connection?

In de repository class op het einde van elke CRUD methode (try-catch-**finally**).