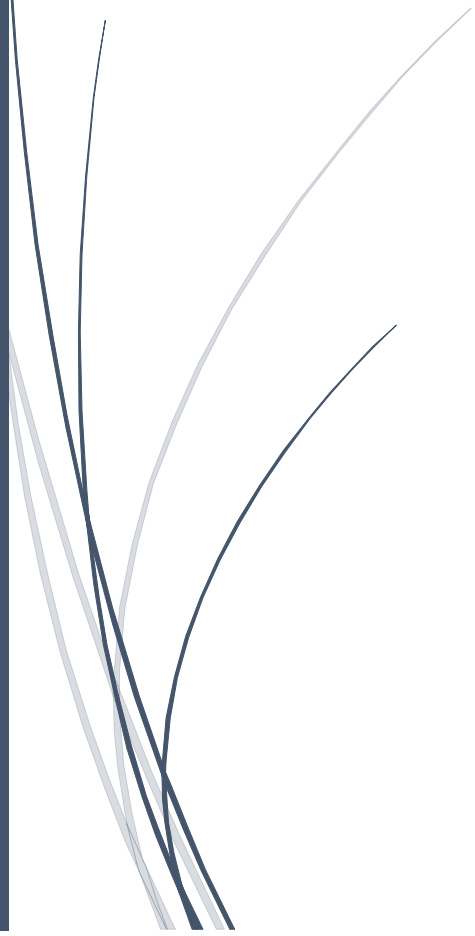
CVO Antwerpen
centrum voor volwassenenonderwijs

Schooljaar 2014-2015

Persoonlijk dossier

Eric Michiels – Oman Biodiversity



CVO ANTWERPEN – CAMPUS HOBOKEN
HBO5 INFORMATICA – PROGRAMMEUR ANALIST

Inhoudsopgave

Inleiding

Vooraleer ik dieper inga op de problemen die ik ben tegengekomen in de loop van dit project, is het belangrijk dat er eerst een aantal zaken worden uitgeklaard. Opdat de problemen en de oplossingen die in dit document aan bod zullen komen beter verstaanbaar zijn.

Het project is geschreven in 'JAVA' waarbij er gebruik is gemaakt van de JDBC library voor de backend code en (voornamelijk) AJAX voor de front-end. Bij aanvang van het project, nog voor de eerste vergadering, is beslist om Hibernate niet te gebruiken. Dit is geen keuze tussen goed of slecht maar een voor de hand liggende keuze geweest, aangezien enkel Tom (Adriaens) en ik ervaring hebben met Hibernate en tijdsdruk een grote rol speelde bij de uitvoering van het project. Het projectwerk is namelijk pas van start gegaan toen de tentoonstelling al 2 weken bezig was.

Indien u het algemene dossier reeds gelezen heeft mag u onderstaande paragraaf overslaan.

Onze werkgever is Filip Keunen, zaakvoerder van het concepten bureau "Polyphaemus". In opdracht van de regering van Oman en de Oman Animal and Plant Genetic Resources Center(OAPGRC) is er een mobiele tentoonstelling opgestart waarbij kinderen tussen de leeftijd van 8 en 12 jaar, de fauna en flora ontdekken die het land te bieden heeft.

Op deze manier wordt er getracht om van jongs af aan een respect voor de natuur aan te leren en zo een bevolking tot stand te krijgen die milieubewuster omgaat met zijn omgeving. De opdracht bestaat uit het voorzien van een database alsook een web applicatie voor het beheer hiervan. De wetenschappers die meewerkten aan het project kunnen op deze manier via een web applicatie, die tijdens de tentoonstelling aan de bezoekers wordt aangeboden, de geselecteerde fauna en flora digitaal beschikbaar stellen. Deze laatstgenoemde applicatie geeft de bezoeker van de tentoonstelling de mogelijkheid om met zijn/haar mobile device een QR-code in te scannen en zo een kwartetkaart van het dier of de plant waar ze bij staan kunnen bekijken op dit toestel. Tevens is het mogelijk om in de algemene one-page applicatie meer te weten te komen over: de tentoonstelling, de overige werelden (dieren, zee, plant, schimmels), social media en spelletjes.

Problemen

Probleem 1: Een foto opslaan in de databank

Een probleem waarmee iedereen in bijna elke taal geconfronteerd wordt, is “hoe kan ik een foto opslaan in een databank en daarna weer tevoorschijn halen op de website?”. De manier waarop men dit kan doen ligt voor de hand, als men weet hoe een servlet juist werkt.

Een servlet kan data doorgeven van de ene servlet naar de andere (of van de ene pagina naar de andere), maar heeft ook een GET en POST methode. De laatste 2 methodes kunnen worden gebruikt bij o.a. images naar een database weg te schrijven of om bytearrays op te halen uit de databank en te converteren vooraleer ze als image worden doorgegeven. Er zijn meerdere manieren om dit te bewerkstelligen, maar hieronder geef ik u de manier waarop ik het opgelost heb.

Code knipsel:

```
Part filePart = request.getPart("upfileOrganism");
InputStream fileContent = filePart.getInputStream();
byte[] bytes = IOUtils.toByteArray(fileContent);

if (bytes.length == 0) {
    BufferedImage image = ImageIO.read(URI.create("http://fuengirolapoolleague.com/bar_images//no-image-availabl
    ByteArrayOutputStream b = new ByteArrayOutputStream();
    ImageIO.write(image, "jpg", b);
    bytes = b.toByteArray();
}
```

De Part variabele die opgevraagd wordt krijgen we via een input box op de website.
Hierna converteer je die variabele naar een Inputstream.
Een databank kan echter geen Inputstream opslaan, dus moeten we de bytes van deze file hebben. Dit doe je door een JAR te importeren, nl. de IOUtils.lib. Door de methode.toByteArray aan te roepen kan je een Inputstream converteren naar een byte Array en die kan je op zijn beurt weer opslaan in een database als een Blob variabele.

Een belangrijk gegeven is dat je specifiek moet aangeven dat de servlet multipartconfig gegevens moet verwerken. Bovenaan de servlet moet dit gedefinieerd worden als @Multipartconfig, alsook in de form op de website moet dit worden aangegeven. Dit doe je in de <form> tag als volgt:

```
<form class="form form-horizontal" id="create-organism-form" data-toggle="validator"
enctype="multipart/form-data" >
```

Door een multipartconfig annotatie te gebruiken, kan je de Part waarde van een bestand opvragen door bijvoorbeeld de.getPart() methode aan te spreken.

Probleem 2: Een foto weergeven op de detailpage.

Een aansluitend probleem dat de kop opsteekt bij foto's in een webapplicatie is: "hoe krijg ik de geconverteerde foto terug als een werkbare foto op de website en hoe kan ik dit dynamisch laten inladen?".

De rest van de website werd dynamisch geladen via AJAX, maar een foto wordt opgeslagen als een bytearray en dit bleek toch niet zo gemakkelijk bij het inladen. Een manier om dit te converteren en via AJAX te doen is door te converteren naar een Base64 String. Na ettelijke dagen dit te hebben geprobeerd zonder resultaat heb ik ervoor gekozen om dit via een afzonderlijke servlet te doen. De code hiervan is als volgt:

In de DAL van het organisme vragen we de bytes:

```
public static byte[] selectPhotoById(int id) throws SQLException {

        byte[] photo = null;
        conn = DataSource.getConnection();
        stmt = conn.prepareStatement("SELECT organism.photo\n"
                                   + "FROM organism\n"
                                   + "WHERE organism.organism_id=?");
        stmt.setInt(1, id);
        ResultSet rs = stmt.executeQuery();
        rs.next();
        photo = rs.getBytes("photo");
        conn.close();
        return photo;
}
```

In de service spreken we deze methode aan als volgt:

```
public static byte[] selectPhotoById(int id) {

    byte[] photo = new byte[2];
    try {
        photo = DAL.DaOrganism.selectPhotoById(id);
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    }
    return photo;
}
```

Nu we de bytes hebben voor de foto die bij het geselecteerde organisme hoort kunnen we deze gebruiken in de servlet.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);

                byte[] pic = ServOrganism.selectPhotoById(Integer.parseInt(request.getParameter("id")));
                if (pic.length > 2) {
        response.reset();
                response.setContentType("image/jpeg");
                response.setContentLength(pic.length);
                response.getOutputStream().write(pic);
                    }
}
```

Dit moet altijd in de doGet methode van de servlet, omdat je gegevens opvraagt van de server. De conversie van een bytearray naar een foto is veel korter en kan gedaan worden met de response.getOutputStream().write() methode.

Via AJAX kan deze servlet vervolgens worden aangesproken samen met de andere variabelen als volgt:

```
// select detail from organism
$(document).on('click', 'table #detail-organism-btn', function () {
    document.getElementById('detail-organism').style.display = 'block';
    document.getElementById('fade').style.display = 'block';

    var id = ($(this).attr("value"));
    $.ajax({
        url: 'SelectOneOrganismById?id=' + id,
        type: 'GET',
        dataType: 'JSON',
        cache: false,
        async: true,
        beforesend: function () {
            $('#detail-organism').html('');
        }
    }).done(function (data) {
        $('#img-detail').html('<img class="img-responsive img-thumbnail" src="SelectPhotoById?id=' + id + '" height="200px" width="200px">');
```

Voor de detailpage bleek dit niet te werken, omdat de detailpage zich op een andere JSP bevond dan de homepage. Daarom is er bij de detailpage gekozen voor een aparte servlet die een Array List doorstuurt naar de detailpage en de variabelen worden inline geladen vanuit deze Array List.

In de admin applicatie daarentegen kan deze beschreven methode wel worden gebruikt, aangezien we hier werken met pop-up-forms die op dezelfde pagina worden geladen.

Probleem 3: Regex vs. non-ASCII

De belangrijkste vereiste bij een web applicatie die gebaseerd is op een databank is het voorkomen van SQL injectie enerzijds en html of javascript injectie anderzijds. Natuurlijk moet het wel nog mogelijk zijn om non-ASCII karakters toe te voegen, aangezien dit een web applicatie is voor een biologische tentoonstelling (wetenschappelijke namen in het Grieks).

Hierdoor kwam ik uit op 3 regex statements voor de admin web applicatie. Tijdens de tientallen tutorials die ik doorgenomen heb, kwam ik terecht op een website waarbij een regex kan getest worden en je enorm veel informatie geeft bij de betekenis van alle karakters.

Input vs. Textarea vs. E-mail

Input

Regex: <input class="form-control" type="text" name="organism-common-name" maxlength="50" data-delay="1200" pattern="[^()[\]{}*&^%$<>#0-9@!]+$" required/>

In deze velden is alles toegelaten behalve de tekens die zouden kunnen aantonen dat de gebruiker probeert om html code toe te voegen zoals bv: <script></script>.

Om die reden zijn de tekens /, \,[,],<,>, niet toegelaten. Overige tekens die niet zijn toegelaten zijn er om ervoor te zorgen dat de gebruiker tot een zeker consistentie gedwongen wordt en correcte data invoer.

Volledige ontleding zoals gedefinieerd vanuit de testomgeving:

- [^()[\]{}*&^%$<>#0-9@!]+ match a single character not present in the list below
  - Quantifier: + Between one and unlimited times, as many times as possible, giving back as needed [greedy]
  - ()[a single character in the list ()[literally
  - \] matches the character] literally
  - {}*&^%$<># a single character in the list {}*&^%$<># literally (case sensitive)
  - 0-9 a single character in the range between 0 and 9
  - @! a single character in the list @! literally
- $ assert position at end of the string
- " matches the characters " literally
- g modifier: **g**lobal. All matches (don't return on first match)

Textarea

Regex: <textarea class="form-control" rows="3" name="organism-description" data-error="error" data-pattern="/^$|^([^<>\[\]{}\\\\\////]+(\r\n)?)$/g" data-pattern-error="Use of special keys: [\/]{}<> is not allowed"></textarea>

Deze regex verschilt omdat er meerdere lijnen zijn waarbij er moet gecontroleerd worden. Dit kan door in de regex te schrijven dat er controle moet zijn op meerdere lijnen (\n) én op carriage return.

Volledige ontleding zoals gedefinieerd vanuit de testomgeving:

- 1st Alternative: ^$
  - ^ assert position at start of the string
  - $ assert position at end of the string
- 2nd Alternative: ^([^<>\[\]{}\\\\\////]+(\r\n)?)$
  - ^ assert position at start of the string
  - 1st Capturing group ([^<>\[\]{}\\\\\////]+(\r\n)?)
    - [^<>\[\]{}\\\\\////]+ match a single character not present in the list below
    - Quantifier: + Between one and unlimited times, as many times as possible, giving back as needed [greedy]
      - a single character in the list <> literally (case sensitive)
    - \[matches the character [literally
    - \] matches the character] literally
    - {} a single character in the list {} literally
    - \\ matches the character \ literally
    - \\ matches the character \ literally
    - //// the literal character /
  - 2nd Capturing group (\r\n)?
    - Quantifier: ? Between zero and one time, as many times as possible, giving back as needed [greedy]
    - Note: A repeated capturing group will only capture the last iteration. Put a capturing group around the repeated group to capture all iterations or use a non-capturing group instead if you're not interested in the data
    - \r matches a carriage return (ASCII 13)
    - \n matches a line-feed (newline) character (ASCII 10)
- $ assert position at end of the string
- g modifier: **g**lobal. All matches (don't return on first match)

E-mail

```
<input class="form-control" type="email"
placeholder="ex_ample123@example.example" data-delay="1200" name="email"
pattern="^\w+([-+.']\w+)*@[a-zA-Z]+([-.]\w+)*\.[a-zA-Z]+([-.][a-zA-Z]+)*$" data-
error="E-mail address is invalid" required>
```

Bij de validatie van e-mailadressen is het belangrijk dat je rekening kan houden met meerdere domeinen bv: student@cvo.antwerpen.be. Dit kan door de regex op te splitsen en validatie op te maken bij elk onderdeel. Deze onderdelen kan men scheiden door gebruik te maken van [] en (). Voor de validatie van e-mailadressen kan men ook gebruik maken van html5 validatie door type=”email” te gebruiken. Dit zal echter maar 1 domein toelaten bv: eric@michiels.be, en houdt geen rekening met eventuele non-ascii tekens.

Volledige ontleding zoals gedefinieerd vanuit de testomgeving:

- ^ assert position at start of the string
- \w+ match any word character [a-zA-Z0-9_]
  - Quantifier: + Between one and unlimited times, as many times as possible, giving back as needed [greedy]
- 1st Capturing group ([-+.']\w+)*
  - Quantifier: * Between zero and unlimited times, as many times as possible, giving back as needed [greedy]
  - Note: A repeated capturing group will only capture the last iteration. Put a capturing group around the repeated group to capture all iterations or use a non-capturing group instead if you're not interested in the data
  - [-+.'] match a single character present in the list below
    - -+.' a single character in the list -+.' literally
  - \w+ match any word character [a-zA-Z0-9_]
    - Quantifier: + Between one and unlimited times, as many times as possible, giving back as needed [greedy]
- @ matches the character @ literally
- [a-zA-Z]+ match a single character present in the list below
  - Quantifier: + Between one and unlimited times, as many times as possible, giving back as needed [greedy]
  - a-z a single character in the range between a and z (case sensitive)
  - A-Z a single character in the range between A and Z (case sensitive)
- 2nd Capturing group ([-.]\w+)*
  - Quantifier: * Between zero and unlimited times, as many times as possible, giving back as needed [greedy]
  - Note: A repeated capturing group will only capture the last iteration. Put a capturing group around the repeated group to capture all iterations or use a non-capturing group instead if you're not interested in the data
  - [-.] match a single character present in the list below
    - -. a single character in the list -. literally

- \w+ match any word character [a-zA-Z0-9_]
  - Quantifier: + Between one and unlimited times, as many times as possible, giving back as needed [greedy]
- \. matches the character . literally
- [a-zA-Z]+ match a single character present in the list below
  - Quantifier: + Between one and unlimited times, as many times as possible, giving back as needed [greedy]
  - a-z a single character in the range between a and z (case sensitive)
  - A-Z a single character in the range between A and Z (case sensitive)
- 3rd Capturing group ([-.][a-zA-Z]+)*
  - Quantifier: * Between zero and unlimited times, as many times as possible, giving back as needed [greedy]
  - Note: A repeated capturing group will only capture the last iteration. Put a capturing group around the repeated group to capture all iterations or use a non-capturing group instead if you're not interested in the data
  - [-.] match a single character present in the list below
    - -. a single character in the list -. literally
  - [a-zA-Z]+ match a single character present in the list below
    - Quantifier: + Between one and unlimited times, as many times as possible, giving back as needed [greedy]
    - a-z a single character in the range between a and z (case sensitive)
    - A-Z a single character in the range between A and Z (case sensitive)
  - $ assert position at end of the string
  - " matches the characters " literally
  - g modifier: **g**lobal. All matches (don't return on first match)

Probleem 4: Non-ASCII in een MySQL databank

Het probleem van non-ASCII tekens hadden we pas door toen de databank in gebruik werd genomen. We hoopten dat het door een aanpassing in de regex opgelost kon geraken, maar dit was helaas niet het geval. De reden hiervan hebben we pas na ettelijke uren ontdekt. De database was nl. opgemaakt in een latin1 configuratie, en voor non-ASCII tekens moet deze opgemaakt zijn in UTF-8. In plaats van de hele databank opnieuw aan te maken hebben we geprobeerd om de reeds bestaande database aan te passen.

Dit kan gedaan worden als volgt bij een nieuwe databank:

CREATE DATABASE mydb

DEFAULT CHARACTER SET utf8

DEFAULT COLLATE utf8_general_ci;

Bij een conversie van een bestaande databank:

ALTER TABLE `organism` MODIFY COLUMN `common_name` VARCHAR(255) CHARACTER SET utf8;

Eén van de belangrijkste aandachtspunten hierbij zijn dat zowel de charset als de collatie aangepast worden naar de nieuwe charset. Na de conversie van de databank zijn ook alle insert servlets aangepast, opdat alle data correct geconverteerd wordt van UTF-16 naar UTF-8.
Dit wordt gedaan als volgt:

```
response.getWriter().write(Service.ServOrganism.insert(new String(request.getParameter("organism-scientific-name").getBytes("iso-8859-1"), "UTF-8"
        new String(request.getParameter("organism-common-name").getBytes("iso-8859-1"), "UTF-8"),
        new String(request.getParameter("organism-local-name").getBytes("iso-8859-1"), "UTF-8"),
        new String(request.getParameter("organism-description").getBytes("iso-8859-1"), "UTF-8"),
        subfamilyId,
        habitatIds,
        new String(request.getParameter("organism-population").getBytes("iso-8859-1"), "UTF-8"),
        seasonIds,
        Boolean.parseBoolean(request.getParameter("organism-indigenous")),
        Boolean.parseBoolean(request.getParameter("organism-cultivated")),
        Boolean.parseBoolean(request.getParameter("organism-endangered")),
        Boolean.parseBoolean(request.getParameter("organism-medicinal")),
        new String(request.getParameter("organism-benefits").getBytes("iso-8859-1"), "UTF-8"),
        new String(request.getParameter("organism-dangerous").getBytes("iso-8859-1"), "UTF-8"),
        new String(request.getParameter("organism-threats").getBytes("iso-8859-1"), "UTF-8"),
        new String(request.getParameter("organism-opportunities").getBytes("iso-8859-1"), "UTF-8"),
        bytes,
        new String(request.getParameter("organism-links").getBytes("iso-8859-1"), "UTF-8"),
        eatenByOrganismIds,
        eatingOrganismIds,
        new String(request.getParameter("organism-food-name").getBytes("iso-8859-1"), "UTF-8"),
        new String(request.getParameter("organism-food-description").getBytes("iso-8859-1"), "UTF-8"),
        geolocationIds));
```

Probleem 5: Hoe kan er gewerkt worden met een QR code?

Een van de belangrijkste features die onze opdrachtgever wou kunnen over beschikken is het gebruik van QR-codes in zijn tentoonstelling. Deze QR-codes werden voorzien bij elke entiteit van de tentoonstelling opdat de bezoekers met hun smartphone of tablet meer konden te weten komen over die entiteit.

Het probleem zat in de link die vereist is om zo'n code op te bouwen. Een QR-code kan je makkelijk genereren via bepaalde websites, maar deze vereisen wel een full path. Daarom heb ik gekozen om af te wijken van de one-pager en een aparte detailpage op te maken.

Een werkend voorbeeld van zo'n QR-code:

Er woedde een hevige discussie over het design van de detailpage, waarbij ik een voorstaander was van zachtere kleuren, maar onze opdrachtgever heeft de knoop doorgehakt en gekozen om te werken met dezelfde tinten van de world sectie.

Dit heb ik geïmplementeerd als volgt:

De gele background van de world sectie als background voor de detailpage, aangevuld met de kleuren van de world waartoe de entiteit behoort. Om deze detailpage dynamisch te kunnen laden heb ik een aparte servlet opgesteld waarbij alle kleuren en informatie van deze entiteit worden doorgestuurd naar de detailpage.

De servlet doet dit als volgt:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);

    int id = Integer.parseInt(request.getParameter("id"));
    String color="";
    String bgcolor="";
    Organism o;
    try {
        o = ServOrganism.selectOneById(id);
        ArrayList<Organism> org = new ArrayList<>();
        org.add(o);

        String world = o.getWorld().getWorldName();
        if (world.toLowerCase().contains("animal")) {color = "#f44336"; bgcolor = "#EF5350";}
        else if (world.toLowerCase().contains("plant")) {color = "#4caf50"; bgcolor = "#9CCC65";}
        else if (world.toLowerCase().contains("micro")) {color = "#ffc107"; bgcolor = "#FFF176";}
        else if (world.toLowerCase().contains("marine")) {color = "#03a9f4"; bgcolor = "#29B6F6";}

        request.getSession().setAttribute("view", org);
        request.getSession().setAttribute("colorscheme", color);
        request.getSession().setAttribute("bgcolorscheme", bgcolor);
        RequestDispatcher dispatcher = request.getRequestDispatcher("organismdetail.jsp");
        dispatcher.forward(request, response);

    } catch (SQLException ex) {
        response.sendError(HttpServletResponse.SC_NOT_FOUND, "");
    }


}
```

De id krijgt de servlet door vanuit de world sectie, zoals gedefinieerd in de view.js file:

```
$.ajax({
    url: 'SelectAllOrganismByWorld',
    type: 'GET',
    dataType: 'json',
    cache: false,
    async: true
}).done(function (data) {
    data.forEach(function (o) {
        if (o.world.worldName.toLowerCase().indexOf('plant') > -1)
        {
            $plantOrganisms.append('<a href="organism?id=' + o.organismId + '"><div class="col-lg-3 col-md-3 col-sm-3 col-xs-12 organisms
        }
        else if (o.world.worldName.toLowerCase().indexOf('animal') > -1)
        {
            $animalOrganisms.append('<a href="organism?id=' + o.organismId + '"><div class="col-lg-3 col-md-3 col-sm-3 col-xs-12 organism
        }
        else if (o.world.worldName.toLowerCase().indexOf('marine') > -1)
        {
            $marineOrganisms.append('<a href="organism?id=' + o.organismId + '"><div class="col-lg-3 col-md-3 col-sm-3 col-xs-12 organism
        }
        else if (o.world.worldName.toLowerCase().indexOf('microbial') > -1)
        {
            $microbialOrganisms.append('<a href="organism?id=' + o.organismId + '"><div class="col-lg-3 col-md-3 col-sm-3 col-xs-12 organ
        }
        else
        {
            $plantOrganisms.append('<li>No organisms yet</li>');
            $animalOrganisms.append('<li>No organisms yet</li>');
```

Deze data wordt rechtstreeks opgeroepen op de detailpage en inline verwerkt als volgt:

```
<%@page import="BLL.Geolocation"%>
<%@page import="BLL.Season"%>
<%@page import="BLL.Habitat"%>
<%@page import="java.util.List"%>
<%@page import="org.apache.jasper.tagplugins.jstl.ForEach"%>
<%@page import="java.util.ArrayList"%>
<%@page import="BLL.Organism"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%
    ArrayList<Organism> organism = (ArrayList<Organism>) session.getAttribute("view");
    String headcolor = session.getAttribute("colorscheme").toString();
    String bgcolor = session.getAttribute("bgcolorscheme").toString();
%>
<!DOCTYPE html>
<html>
    <head>
        <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
        <link rel=stylesheet href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css">
        <link rel="StyleSheet" type="text/css" href="css/main.min.css">
        <title>Detailed view</title>
    </head>
    <body>
        <%for (Organism o : organism) {
        %>
        <div class="col-lg-12 detail-back">
            <div class="col-lg-8 col-md-offset-2 detail-header" style="background-color: <%= headcolor%>">
                <h2 class="text-center"> <%= o.getCommonName()%></h2>
            </div>
            <div class="col-lg-8 col-md-offset-2 detail-container rsp" style="background-color: <%= bgcolor%>">
                <br>
                <br>
                <div class="col-lg-3 col-md-offset-1 text-left">
                    <a href="index.jsp#worlds"><button class="button btn-material-orange">Go to worlds</button></a>
                    <br>
                    <br>
                    <img class="img-responsive" src="SelectPhotoById?id=<%=o.getOrganismId()%>">
                    <br>
                </div>
                <div class="col-lg-3 col-md-offset-1 text-left">
                    <fieldset>
                        <legend>Taxonomy</legend>
                        <div class="detail-container">
                            <div>
                                <label>Scientific name:</label>
                                <label class="detail-text"><%= o.getScientificName()%></label>
                            </div>
```

Probleem 6: De presentatie

Van in het begin wou ik onze presentatie zo professioneel mogelijk maken, maar dit bleek makkelijker gezegd dan gedan. Het idee dat in mijn hoofd speelde, was om op dezelfde manier te werk te gaan als bij een TED conferentie. Natuurlijk is dit een pak moeilijker als je niet over de nodige oratorische talenten beschikt, of over het juiste materiaal.

Gelukkig had ik nog enkele mogelijkheden via mijn Adobe account, en kon ik een proefperiode van 30 dagen gebruiken voor Adobe AE. Dit programma staat je toe om video's te maken die op elke manier configureerbaar zijn en waar je echt het maximum kan halen uit je presentatie.

Het doel was om een intro te kunnen maken, waarin ik zeer snel een inzicht kon geven in onze structuur en onze missie. Dit is me gelukt na ettelijke dagen online tutorials te volgen.
Het resultaat was een film van ongeveer 1 minuut en ongeveer 17 manuren die ik hieraan besteed heb. Ik moet er wel bij zeggen dat ik tijdens die 17 uren ongeveer 6x gecompileerd heb, en dit compileren kon oplopen tot een half uur. Dus het effectieve aantal gepresteerde uren ligt op +/- 14 uur, wat nog steeds gigantisch is, maar ik had dan ook nog geen ervaring met multimedia en Adobe AE.

Een bijkomend obstakel bij het werken met veel animaties is de file-grootte. Het totale bestand (intro + presentatie) was ongeveer 5 GB. Dit is uiteraard veel te veel, en zouden we onmogelijk (gratis) in de Cloud kunnen steken. Natuurlijk kan je de kwaliteit bijschaven, maar dit zou al het werk dat hieraan vooraf gegaan is, teniet doen. Daarom heb ik gebruik gemaakt van de freeware "Handbrake". Deze freeware geeft je de mogelijkheid om de gradatie van kwaliteit aan te passen en zo je file te resizen. Door hiermee slim om te gaan, ben ik erin geslaagd om de grootte terug te schroeven tot een kleine 300 MB, en toch enige vorm van HD-kwaliteit te behouden.

Als ik vandaag de keuze had om het project opnieuw te doen, had ik het niet anders gedaan. Zolang de werkuren productief zijn en met een duidelijk doel voor ogen is het werk verantwoord, op voorwaarde dat het een meerwaarde creëert. En dit laatste is, ook al stinkt eigen lof, zeker van toepassing.

Besluit

Dit project was van in het begin een ware uitdaging. De tijdsdruk was enorm, de communicatie met het buitenland een ramp en nieuwe technologieën gebruiken staat altijd gelijk aan nieuwe bugs leren kennen. Maar de samenhorigheid van de groep was telkens een constante. Er werd gewerkt, veel en graag. Er werden elke dag nieuwe ideeën naar voren gebracht en er was ruimte voor een debat hierover in een amicale sfeer waarbij userexperience en simplicity hoog in het vaandel werd gedragen.

Samenwerken aan een applicatie die effectief in gebruik is genomen door honderden mensen was een drijfveer die niet kon geëvenaard worden. Had ik dit projectwerk willen omruilen voor een stage bij een bedrijf, en op basis van mijn prestaties bij die onbekende werkgever gequoteerd worden? Neen. Een stage was voor mij geen optie, aangezien ik reeds werk en dit hoogstwaarschijnlijk niet had kunnen combineren. Maar als die stage inhield dat je meewerkt aan een project van A tot Z, van BSS tot OSS en je hier zelf actief onderdeel van zou kunnen uitmaken, dan is mijn antwoord volmondig: “ja”.

Bronvermelding

https://docs.moodle.org/23/en/Converting_your_MySQL_database_to_UTF8#Converting_a_database_containing_tables

https://www.blueboxcloud.com/insight/blog-article/getting-out-of-mysql-character-set-hell

http://paulkortman.com/2009/07/24/mysql-latin1-to-utf8-conversion/

https://dev.mysql.com/doc/refman/5.0/en/charset-applications.html

http://www.bothernomore.com/2008/12/16/character-encoding-hell/

https://servlets.zeef.com/bauke.scholtz

https://regex101.com/#javascript