120912 Victor Rudolfsson

Datamodellering & Databasesystemer

Oblig 6

Introduction

This part consists of two files: index.php, and fetchxml.php.

Because I'm not entirely sure of what I'm actually supposed to write in this PDF, I resort to further explaining the purpose of each PHP file in Oppgave 1, and some troubles I had with the XSD in Oppgave 2. I had originally included the contents of both files in this PDF, but because of formatting issues – especially with the XSD's nesting levels – I removed them again and instead decided to just attach them.

As requested, all files are included in this zip-file.

Oppgave 1

This part consists of a single file, largely split in two: index.php.

The first thing we do, is to check whether there's a search string – if there isn't, we'll create a HTML page and some simple CSS. The only thing this page does is present the user with a text input field and a submit button.

After entering [comma separated] search strings for product IDs, the page reloads and sends the GET parameters to itself, thus passing the first test – this time there is a search string.

First we set up a PDO object and establish a database connection. Then, we strip away whitespaces from the search string – after all, we don't know if the user wrote ",," or ", " or " , " between the product IDs, or just accidentally hit space before or after.

Next, we'll use the attached DTD file to create a DOMDocument, and set the specified attributes for the root node. We then append a *Produkter* element to this, after which we continue to the database queries.

I decided to prepare the statement, even though it's only run once, mainly for escaping purposes. So, to prepare the statement we first have to check whether we have one, or several product IDs and prepare and execute it accordingly.

Last but not least, we'll iterate through each returned row and store every product ID in an array as to make sure we don't add it twice. After adding a product ID element, if it occurs again, we'll only include the ProduktDetalj element – and if we hit a new one we'll create a new product ID.

Oppgave 2

This assignment took me quite some time to get straight. I felt that XSD had more exceptions than it had rules – for example, you can have a reference to another element but only if it's defined up at the top (not nested in somewhere); or that an element may contain an attribute with specific rules, but not at the same time as it's of a certain type – then the rules must be defined up top and referenced instead, and so on.

Apart from this, I really did enjoy XSD. XSD felt more logical than DTD for XML validation, as it felt more like writing a template in the same kind of way as compared to DTD which just didn't.

?>