P7 Project

Web Engineering Assignment

Alexander Nilsson - anilss11
Christian Jødal O'Keeffe - cjok11

Architecture and Patterns 1

1.1 Choice of Data Set and its Structure

We have decided to use the Mondial data set structure as a base for the database structure. We choose this geographical data set over the others because we found it well structured and suitable to implement as a database. The Mondial data is structured in a hierarchy which relative ease can be projected into a database and insures a relative low data redundancy. A issue that could arise would be connected to the low redundancy which can lead to more joins to achieve the desired information. We choose the low redundancy approach to lower the space usage and also the observation that relevant data would not involve a lot of join actions.

The Modial data sets hierarchy starts with declaration of continents. Then each country refers to the continent it resides in. Each country then have a series at city connected to it or provinces which contains the cities. Each level in the hierarchy have a set of attributes that describes information.

1.2 MVC pattern

The model-view-control pattern is a pattern which insures data binding between the interface, the view and the data, the model through interaction from the user, the controller. The general idea is to create a model which contains the data which is to be processed in somewhere and proceed to load the data into the model. Next step is to create views to display the information to the users in manageable and nice way. The last step is to create the controller, methods and functions for the user to manipulate the shown information and aggregated it back to model which then is updated.

1.2.1 Model-part

There are different structures that can used to construct the model of the data. One way is to a gateway, which through some functions finds the needed data in the database. Another method is to create an objects or records which is used to contain the data. A third method is to use transaction to get and update data in the database. A fought method is to create a object to contain the data and another object to map the data between the database and the application.

FiXme Fatal: skrive om view og controller parts ?

This is a widespread technique to create web application and us supported by many frameworks. An example could be angular which practices two way binding in their constructions.

1.3 Implementation of MVC

The implemention of the MVC pattern have been made using PHP. As data storing, a MySQL database has been used.

1.4 Choice of Pattern in Model

We have chose to have a simple domain model with class table inheritance. This has been chosen due to the close connection between the database and the domain model. This will ensure that all columns are relevant for the rows, and the domain model will easily reflect the content of the database. In comparison to a concrete table inheritance, the primary keys are easily handled. In contrary to both the concrete table inheritance and the single table inheritance the class table inheritance do not do any irrelevant joins, and do not fetch any irrelevant data. This will ensure small, precise tables, which is faster to load from the database. In that way, we will not have to fetch on, possible very big, table. It has the consequence that multiple calls to the database could be required, but it is only the relevant data, which is fetched.

1.5 About the View

To make the view, a template view pattern has been used. Each view has a template, with a output function for outputting the HTML, with contact to the respective controller to access the data.

1.6 About the Controller

There will be a controller corrosponding to each view.

1.7 Implementation of Model

Each table in the database has one model class, and one meta data class. The meta data class is the direct representation of one row in the table. An example can be seen on Listing 1.1. The getters is a direct copy of the elements in the table on the database.

```
<?php
class Continent{
  private $id;
  private $name;

  public function __construct($id,$name){
    $this->id = $id;
    $this->name = $name;
  }

  public function getId(){return $this->id;}
  public function getName(){return $this->name;}
}
?>
```

Listing 1.1: An example of the continent meta data class.

The model class for a table contains all database calls, an array of objects of the meta data class, and functions to update the database. An example can be seen on Listing 1.2.

```
public $string;
  public $listOfCountries;

    public function __construct(){
        $this->getData();
    }

  private function getData(){
    $con = conFatory();

    $result = mysqli_query($con,"SELECT * FROM country");

    $counter = 0;
    $this->listOfCountries = array();

    while($row = mysqli_fetch_array($result)) {
      $tmp = new Country($row[’id’],$row[’name’],$row[’capital’],
          $row[’population’]);
      array_push($this->listOfCountries,$tmp);
      $counter++;
    }

    mysqli_close($con);
  }
```

Listing 1.2: An part of the country model class.

Data I: XML 2

2.1 Implementation of XML

To read data from a xml document we decided to implement a controller which would extract the data from the xml document. The extracted data are then stored in the model, organize of meta-data classes designed for the data. We use the extracted data to fill a MySQL database which allows to access the data through SQL quarries.

We starts by checking if the xml file is there. If this is the case it loading into a simplexml object by the simplexml_load_file method. We then traverse through the simplexml children which represents the xml structure and contains the data. Through different switches we construct objects represent the traversed data. This can be seen on code sample Listing 2.1.

```
if(isset($_FILES["InputFile"]))
  {
      $this->xml=simplexml_load_file($_FILES["InputFile"]["tmp_name"]);
        $this->status = "uploaded";

      $this->counter = 0;
      $this->citycounter = 0;
      $this->continentcounter = 0;

      foreach ($this->xml->children() as $child) {
        switch($child->getName()){
          case "continent":
            $this->continentname= "";
            $this->continentid= "";
            foreach ($child->attributes() as $continentData){
              switch($continentData->getName()){
                case "name":
                  $this->continentname = $continentData;
                  break;
                case "id":
                  $this->continentid = $continentData;
                  break;
          }
        }
```

Listing 2.1: Code snippet illustrating how we reads the xml document.

2.2 Use of XSLT

We have explored the use of XSLT in our application, and have found out, that it was a fine way of quickly representing the content of a XML. It could be suitable to temporary represent the data, e.g. when verifyring a upload of an XML-file, but we do not intent to represent the whole application with XSLT, due to the lack of communication with the controller, and due to our choice of saving our data in MySQL. The example of our XSLT used for reprecenting a list of countries can be seen on Listing 2.2.

FiXme Fatal: Ret denne!!!

```
<xsl:stylesheet version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html"/>
<xsl:template match="/">
 <html>
  <head>
    <h1>Countries</h1>
  </head>
  <body>
   <table border="1">
    <tr bgcolor="#9acd32">
      <th>Countries</th>
      <th>Population</th>
    </tr>
    <xsl:for-each select="mondial/country">
    <tr>
      <td><xsl:value-of select="name"/></td>
      <td><xsl:value-of select="@population" /></td>
    </tr>
    </xsl:for-each>
  </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Listing 2.2: An example of our XSLT file for listing the countries

2.3 Test of xQuary

The dataset mondial used for this application was quarried with xQuary using the program BaseX. We found it a simple and very effective way of selecting and quary an XML file. It is not used for this application, due to the problems of using xQuary with PHP. We tried to restrict the set to just the country Denmark using xQuary, which worked perfectly, but as we save our data using MySQL in our application, it is not a problem, as the SQL language will suite the application fine for quarrying.

Service Based Integration 3

3.1 Webservices

The motivation for developing webservices is for web application to communicate with each other. E.g. if different applications need to access and modify the same dataset. An example could be an API, which is the access to a database, to which other applications can call.

3.2 Web Service Solutions

An attractive feature for a web application could be to providing services that other applications can use to interact it. To provide this functionality different methods such as: shared database, remote producer call, message bus and file transfer been developed.

Shared database can work for multiple applications within the same organization but on a larger scale involving multiple organization it an very insecure solution and even more complex to pull off.

File transfer is only available within the same application were it can access the data on the harddrive. This principle was generalized so it was possible to communicate between multiple computers through remote procedure calls. Remote procedure calls can work by using synchronous communication where an application sends a request and waits for the answer. This poses some downtime while it waits for the answer. Another way is to use asynchronous communication were a queues are implemented to act as a intermediate step for storing the requests and responses. The remote procedure call abstraction covers how the operating system works and communicates with the communication protocols, how the data is transferred and verification of the delivery.

REST is a series of principles which helps web services to communicate through procedure calls. It utilizes a uniform interface which the applications can communicate through. All transfer goes through the hyperlinks and data are represented in lightweight formats as plain text or JSON. REST works by sending a request through HTTP and the server then processes this request and returns the result as HTTP result eg. an JSON. REST is easily set up does not require a lot of time to set up. A limit of REST is that using all 4 verbs. which allows for database manipulation. Another disadvantage is that REST sends all content for HTTP or HTTPS which means is it possible to preform a man in the middle attack and listen to the transactions.

SOAP is a protocol while REST on the other hand is an architecture. SOAP takes more effort compared to REST to setup but provide more security and a bigger flexibility in terms of requesting data and the format in which you receive. SOAP general is setup by creating an envelopes which describes how to request for messages and how to process them. This message is then process on server site through a transaction manager which insures that the transaction is completed correctly. The disadvantage of SOAP is that it is were complex and requires work to setup probably.

3.2.1 Comparison

The SOAP provides a lot of customization in terms of request and response it also require more processing of the response while REST uses a light-weight format which is easily processed. SOAP also takes a lot for work for setting up while REST is more friendly in that aspect. SOAP at the other hand provides more security while REST requires a bit of implementation to encrypt the transactions.

We choose the REST architecture because it covers the needs our application have in terms of transaction. It is also more desired because the much easier to implement rather than SOAP.

3.3 Implementation

To enable the use of routing, the Fat-Free Framework is used. Each route should specify which part that should be acceced. As an example, if you GET to "api/country", it will return the list of countries. You can also get a specific county if you GET to "api/country/@id". If you want to edit a country, you simply POST to "api/country/@id". An example can be seen on

```
$f3->route('GET /country',
    function($f3) {
        echo json_encode($f3->cmodel->listOfCountries);
    }
);
```

Listing 3.1: The API method for getting the list of countries.

The way of using routing was found easy to understand - both for the developers and for the users of the API. To communicate with the database, the API use the model classes made to our application.

Requirments 4

To decide what our applications functionality should be, we will look into the requirements of application and discuss what is important. We start by looking into what the goals for the application are.

A goal for our application is to provide a geographical database for the users, providing a service where they can find relevant information in relation to their search. Developments can in the same way use the service in their application to retrieve information from the database. Another goal could be to provide a tool or service for updating the geographical data in the database. We will provide an example of how one of these goals can be analyzed.

- **Actors:** The actors are the users of the web application and us as the providers.
- **Goals:** To provide the users with information relevant to their search in return for viewing some advertisement.
- **Tasks:** One tasks is to keep the database updated so the users always have access to relevant information. Another tasks is to query the users search to the database and present the result to the users.
- **Resources:** The resource we can provide is the information while the users provide us with views.
- **Soft goals:** That the users are satisfied with the services we provide and they meets their expectations.
- **Dependencies:** For our web application to work it is dependent on a connection to the server.

With the goal in place, we now look at the value exchanges in the between the users and our application. One value exchange could be between the normal users where we provide relevant information to their search together with some advertisements which the users will see together with the information. The income will be through a value exchange point between our application and the advertisement company, where we provide them with views while they provide us payment for the views. We will give an example of how the value exchange between the normal user and the application could be analyzed:

- **Actors:** The user creates the business value through visiting our service and view the advertisement.

- **Value Object:** The value object sent from our application to the user is information with the geographical data, where the value object from the users is viewing the advertisement.
- **Value Port:** The port is our web application.
- **Value Interface:** If the view it in a larger scale, it could be between our application and the advertisement site.
- **Value Exchange:** We exchange information for page views.
- **Market Segments:** We have the users, administrators and the advertisement firm.
- **Value Activity:** When the users enter the site.
- **Dependency Path:** The connection between the user and our application, through the value port.

There would also be a value exchange between the our application and the geographical data administrates where we provide them with the database and service to update it while they provide us with updated information.

The workflow of our application is, e.g. the case when a user view the list of countries. The task for user user is to enter the site and chose the condition for the view (e.g. sort alphabetically). The site should the activate the procedure to call the database and receive the relevant information, as well as calling the advertising firm to get the advertisements. This could also be described by using Petri Nets or similar models.

The information flow is quite similar to the value exchange, as the information flows from the administrators, through our database and to our users through our site.

These different analysis methods for the requirements, each provide valuable information - in some areas they provide the same information, in other unique information about the requirements. As an example, the actors a described in several of the methods. In regard to our application, the value exchange method was especially suitable, as it provided a lot of information about the different aspects about what the web application should handle and provide.

Design 5

A very common design issue of an application, is how to navigate through and present data. We have in our application chosen a navigation design, as well as en presentation for our different views.

5.1 Navigation Designs

There are a number of different design for navigation.

- **Step Navigation**: Is navigation which works in steps, like arrows beneath a picture in a slide show or gallery.
- **Paging Navigation**: A number of pages to step through, like a search result.
- **Breadcrumb Trail**: A list of the parent categories of navigation.
- **Tree Navigation**: A tree structure, where you can explore the different subtrees in the navigation.
- **Site Maps**: A map of the sites content, set up like a folder structure.
- **Directories**: A list of directories.
- **Tag Clouds**: A number of tags, which can vary in size according to their use.
- **A-Z index**: An alphabetic navigation.
- **Navigation Bars and Menus**: A top bar, with links to navigate.
- **Vertical Menu**: A vertical menu, often placed in the side, with links to navigate.
- **Dynamic Menu**: A possibility to dynamically fold our a new sub menu when hovering over an item.
- **Drop Down Menu**: A drop down menu with the possibility to select an item.

These are some of the different navigation designs to chose from when designing your navigation.

5.2 Navigation Design Techniques

There are different design techniques to represent the selected design. We will describe and compare the "Navigation Views" and "WebML Hypertext Schema".

- **Navigation Views**: Navigation views consists of a list of items, which makes then suitable for querying. They are defined as object orientated views, with view name, base data, and other data.
- **WebML Hypertext Schema**: WebML shares the notation of the elements with the navigation views, but it differs as it enables different ways for selecting data within navigation nodes.

WebML can be used to more complex applications than the simple navigation views, and have many possibilities for extension compared to the navigation views. The navigation views is more simple and is suitable for minor or simple applications compared to WebML.

5.3 Navigation

We chose to use navigation view design to create an abstract overview of have our web application. This should provide us with an idea and a way to describe how the navigation would work in our web application.

We start by creating the navigation views for the different pages on our web application. We will use one concrete example to describe the general process. The chosen example is the country view of our application, which is responsible for showing all the countries from the database. This view contains a list of *country* which have *Name*, *ID* and *Population* as attributes. These attributes should be displayed in the tables columns. Each *country* then links into another view were it is possible to edit the given *country*. The country view also contains a link to a add view were it is possible to add countries to the database.

The navigation context scheme is used to describe the context of the navigation views.

We have chosen to use a navigation menu at the top of our web application so the users have fast access to the different features that our web application provides.

We uses vertical menus to provide an overview of available functions on a given page and a navigation menu to navigate through the different pages.

5.4 Presentation

For our web application we have used concrete presentation design method to structure how our application should be presented to the consumers. This method was chosen over abstract data view and abstract presentation design because it allows us to model our application on a more detailed level while others might omit some of the details. Another attractive feature of the concrete presentation method it uses objects in its model.

We have the design a application *top menu* which contains links for the consumers to navigate through the application. For each page in our application we have different presentation bodies containing the specific pages content presented in e.g. tables.

5.5 Reflection

Working which these methods have given us insight in how you can design and prepare for implementing a web application. We think some of the method are better than others but it might depend on the context, our choice can be seen the other sections in this chapter .

FiXme Fatal: muligvis lav ref engang

A problem we encountered was that we already implemented our application and therefore we were a bit influenced by our existing solution.

Data II: Semantic Web Technologies 6

6.1 RDF/RDFS

RDF is an extension of the XML language, which consists of a triple containing a resource, a property and a value. Is a a way of organizing your XML file with this tuple, making it easier to search and storing data.

It has been implemented in our project, by the possibility to export the list of countries with some attributes to a RDF file. We used a PHP library, EasyRDF, to present our data.

We found this way of organizing your data as convenient to some purposes, as it is can be structured very close to our internal class representation. It is a format intended for computers, and not computers, but as the purpose was to store the information for a computer to read later, this is not disadvantage. The EasyRDF was very easy to use and implement, and the RDF format was in general an easy but effective extension of the XML language.

6.2 SPARQL

SPARQL is language used to query search results from a given RDF data set. It works by selecting some data attributes from the data set. These result are restricted by the "where" closure, where the user can specify one or more predicates to find the result which they want. It is also possible to restrict the result in term of size by the "filter" option.

We used SPARQL to search through different data sets to see how it actual works and the results would be from the different searches. We think that SPARQL provides good qualities in terms of searching through large RDF data sets by allowing us to design very specific queries. The general flexibility in terms of end result is also a nice feature that adds the possibilities to customize the results to fit once need.

But as a side note we also think that it can a bit confusing setting the query up for getting the right result.

Quality I: performance and dataflow testing 7

7.1 Performance Test

Our application should be performances tested, to ensure the stability and user experience when the system would be released and hopefully used a lot. This could be done, by generating traffic to our site, e.g. by the use of unit tests. Back Box testing should be used to test load time, and ensure that the site is performing fast enough with greater traffic. This should be followed by white box testing, which could reveal the internal performance, and thereby finding the bottlenecks. This could be to see the performance index of the network and the CPU utilization, and thereby determine if the bottleneck if because of network problems, or it is due to heavy computations. This could be solved by changing the software, or upgrading the hardware. Furthermore, the number of function calls and percentage of time spend on each part of a program, can tell which parts of the program to optimize to get a better performance.

7.2 Application Bottleneck

When developing an web-application it is important to design it, so it can handle a lot of traffic. A method to help the developers in creating a robust design in terms of performance is to look for bottlenecks in your web application. When a bottleneck is located you either looks for the source on a hardware level or on the code level. With the source to problem it is now up to the developers to solve it one way or another.

A bottleneck in our web application could generally be our database. A problem could arise when a lot of users uses the function to update the database which means our database needs to handler a lot of update request at once. A similar problem could be when a lot of users would request the country web page.

Another bottleneck could be when a lot of users wants to see some potential complex statistic of a lot of data, which would put the server under a lot of calculation pressure as well as wait for the database to handle the request.

7.3 Control Flow Graph

We have made a control flow graph for our setData function in the country model. The code for the control flow graph can be seen on Listing 7.1.

```
public function setData($objectToChangeToo){
  if ($objectToChangeToo != null)
  {
    $this->dbReplace($objectToChangeToo->getId(), $objectToChangeToo->getName(), $objectToChangeToo->getCapital(), $objectToChangeToo->getPopulation());
  } else {
    echo "ERROR";
  }
}

private function dbReplace($searchKey, $name, $capital, $population){
  $con = conFatory();
  mysqli_query($con,"UPDATE country SET name='". $name ."', capital='". $capital ."', population=". $population ." WHERE id='". $searchKey. "'");
  mysqli_close($con);
}
```

Listing 7.1: The setData function from the country model

The control flow graph can be seen on Figure 7.1.

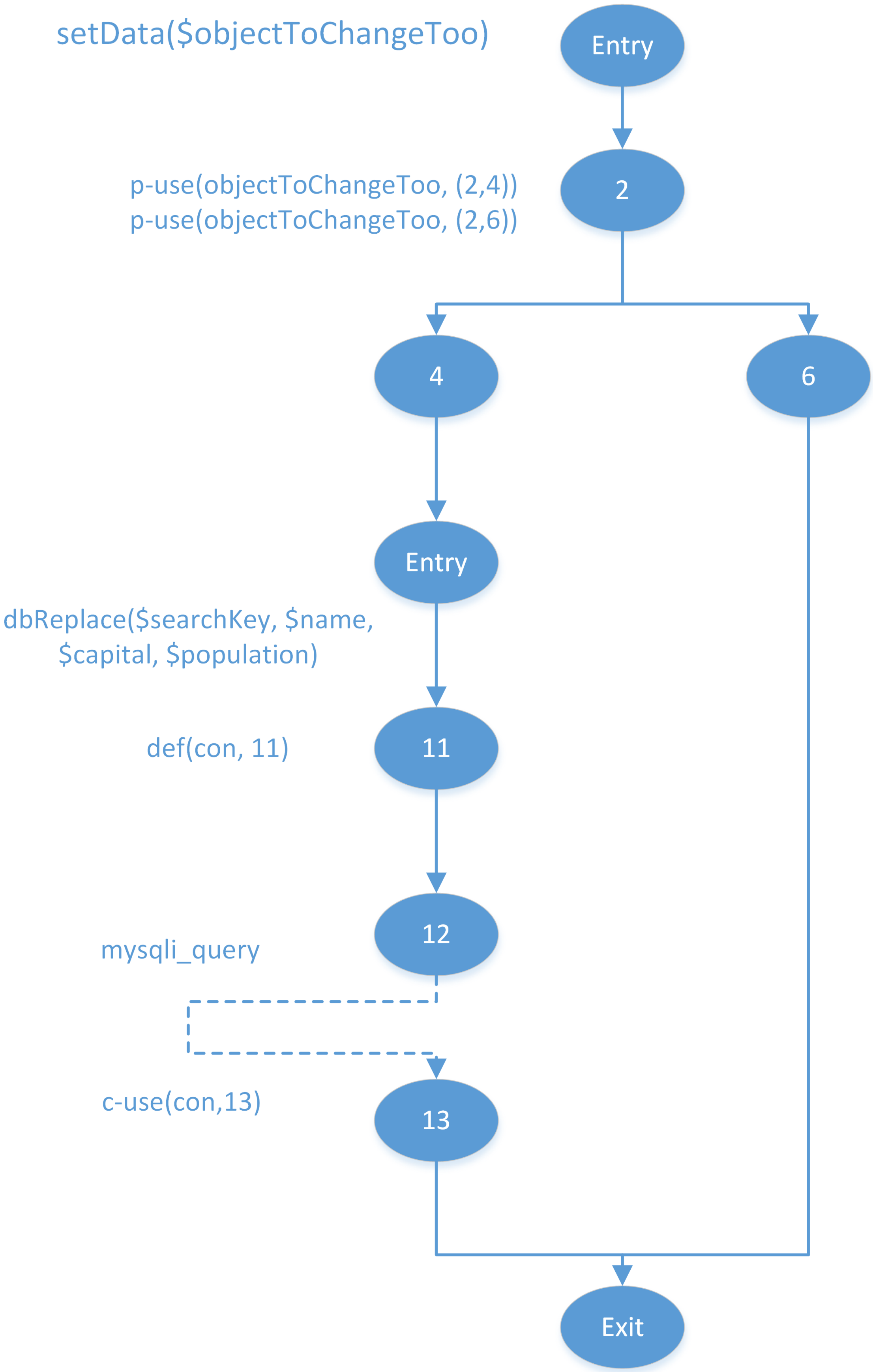
setData($objectToChangeToo)
Entry
p-use(objectToChangeToo, (2,4))
p-use(objectToChangeToo, (2,6))
2
4
6
Entry
dbReplace($searchKey, $name, $capital, $population)
def(con, 11)
11
mysqli_query
12
c-use(con,13)
13
Exit

Figure 7.1: The control flow graph for Listing 7.1

7.4 Analysis of Control Flow Graph

This control flow graph can be used to specify your unittests, as we can see which paths that should be accessed to test the program fully. It can be seen, that the objectToChangeToo variable is the essential variable, in regard to which path is followed. When writing the unittest, this variable should be tested in regard to make the program choose the right path. Furthermore, the graph makes is possible to see at which state the variables is changed, and thereby in which that the unittest should verify the state of the variable.

7.5 Advantages and Disadvantages of Data Flow Testing

An advantage of data flow testing is that it allows the developer to find errors at a very low level of the application and helps for pinpointing the location and cause of the error. This feature is a consequence of following the data flow very closely.

Another advantages is that the data flow test provides a system that you as a developer can use to test your application on different levels as: function level, object level and application level.

A disadvantage is that the data flow requires a lot of time to preform on large complex systems, a workaround is to only preform the test on the most important functions. Another disadvantage is that the data flow graphs, used in the testing, easily gets outdated and needs a lot of maintains to keep it up to date if changes are made to the program.

Quality II: usage analysis and A/B testing 8

8.1 Usage Analysis

Usage analysis is a analysis method were you look the web log for your servers. General you will need to extend these logs with additional information in order to preform the more in depth analytic

Usage analysis can provide a variety of information of your web application which you can use to derive an array of things from. It can for example be used to find which pages in your web application are used and which of the pages that are the most demanded. The analysis can also uncover pages which are not reachable or pages which are not used at all by looking at the number of requests and which pages their are requesting. The analysis can also be used to find missing links by observing the navigation pattern of the users and thereby determine abnormal behavior.

With usage analysis we use of the benefits that this method provides to improve our web application. We could for example use it for understand the users behavior and optimize our web applications navigation. We can also use it to find unused pages and functionality, and evaluate if they need a redesign to fit the users needs more or be removed.

To use it we would have to extend our web logs and design analysis models to actually extract the wanted information our of the logs. To extend our web logs we could use e.g. use log4php, a library for extending the web logs for php applications.

8.2 Implementation of AB-testing

In AB-testing, users are randomly exposed to two versions of a page, giving the possibility to compare the results, finding the best version of the site.

If we were to implement AB-testing in our project, we would need resources not given for this mini project. It could be possible for us to implement the coding part of AB-testing if given enough time, but we do not have the users for our page, compared to the number needed for performing an AB-test.

If we pretend we would have the needed resources, AB-testing could be implemented by making two different views of a page, e.g. the edit country page. This could easily be done, as it is only the view that should be changed, and the functionality could be preserved from the model and controller. We could then make two views of the edit country page, with two different layouts, and measure the speed, or how easy the user changes the country, to make the page as effective as possible.

8.3 Advantages and Disadvantages of AB Testing

An advantage of A/B testing is that it allows the developers to test a new design compared to the current design. The test is measured through a series evaluation criteria. The developers can then use the result of test to evaluate the new design and check it actual improves the product compared to the old design.

Another advantage is that, it is fairly easy to implement A/B testing if the application uses the MVC pattern. The reason for this, is that you have to create a view for the new design and redirect to it, so both the model and controller can be reused.

A disadvantage of A/B testing is that it requires a lot of users to be able to provide a result that can be used. Another disadvantage of A/B testing is that it requires some time to implement the rerouting of users and if it is a complex system that already reroutes to internal servers, even more hardware power is needed to reroute the users to the test pages.

8.4 Implementation of log4php

log4php can quite easily be implemented in our PHP framework. When included, you can specify the different logs in the PHP code. An example can be seen on Listing 8.1.

```php
include(’Logger.php’);
$logger = Logger::getLogger("main");
$logger->info("This is an informational message.");
$logger->warn("I’m not feeling so good...");
```

Listing 8.1: An example of an use of log4php

In our project, we would locate the logs in the controller, logging each time the user make an error in the forms, and the time used on the different pages. This can be used to see which page result in the fewest user errors, and which page makes the user complete their task in shortest time.

Web Application Security 9

9.1 Security

Security is an important part of designing a web application because a vulnerable system can result in the attackers exploiting the system to personal use and potentially hurt the users of the system.

Some of the areas the attackers could exploit, to gain authenticated access to your web application, are: buffer overflow, input validation attack, format string attack, SQL injection, access control weakness, authentication and session.

In an attack by exploiting buffer overflow, the attacker overflows a variable through input field in the web application and forces a system error. In a input validation the attacker tries to exploit input fields which certain input e.i. string escape followed by a SQL statement, also known as a SQL injection. A SQL injection attack can be used to extract data from the database, to delete the database or to store procedures to deny users access to a service or execute remote commands. A format string attack the attacker tries to provoke a system error through parsing e.g. a string value to a integer variable. In a access control weakness attack the attacker exploits weak points in the API e.i. calls to the API that have no or weak access control. In the authentication and session attack the attacker exploit e.i. a cookie to gain user access to a web application.

9.2 Possible Attacks on Our Application

Our application has a lot of problems in regard to security. As example, no input field, e.g. on the edit country page, is validated. The input values should of course be validated and escaped to ensure that e.g. SQL injection is not possible. The login has not been implemented, so it is possible for everyone to use our edit country method. The API of the application do not use any API keys or other authentication, which will make it very easy to abuse our API. The API does not validate the inputs, of escape the inputs, which again make SQL injection possible. The site does not take care of encodings or overflows either.

9.3 Preventing Attacks in our Web Application

As we have determined in previous section, our system contains security breaches. To fix a lot of the security breaches, we could follow the defensive practice and evaluate the input of all the input fields and check to see if their content can not be parsed, if they contain characters to escape a string or if they contain any SQL statements. A flaw to this approach is it require us, the programmers, to actively remember to preform the checks at each input. Another practices we could use is the detection and prevention practice were you analyze or find patterns in the SQL request and only execute the request that does not raise an alert.

There are no login on our web application which means everybody can access functionality in the web application to change the database which could be exploited. Therefor by implement login and sessions we can make it harder for the attackers to gain access to database. By implementing sessions we also improve the security of the API, by validating the API calls and check if they are valid.

Scalability 10

10.1 Scalability Strategy

A web store or other kind of web applications income in someway origins from the users and therefore it is important to provide a nice user experience. Therefore it is important that the users can easily access the web application without high latency or unavailable pages due to high traffic on the server. Therefore it is important that the web application is scalable so it possible to scale the application to more servers in order to handle more traffic and thereby more customers.

General when scaling an application it is a requirement to redirect the users, to the servers which are not stressed, before it makes any sense. For this there exists different techniques e.g. redirection, round robin DNS, load balancing switch and load balancing proxy. With this the hardware is able to handle the extra traffic but the software still needs to handle the scaling too.

A method for scaling an application is to use replication of the database, so the database and the application code is located on each of the servers. This increases the traffic that the web application can handle by allowing more accesses to the data and more processing power. A down side to this implementation is that you need to handle the synchronization between the databases in order for theirs data to be consistent. Another downside is a lot of duplicated data is stored across multiple databases.

Another method for scaling is called content aware and centers around have a set of sub databases containing the data for each of the applications pages and a central database containing all the data. This approach allows the application to scale and rises the user capacity while keeping the data redundancy low. A downside is that a server still can experience heavy load if a lot of users access the same specific page. A solution to this problem could be to utilize database replication of the pages with a lot of daily access but it would require additional work.

A third method could be to utilize catching to minimize the number of queries to the central database. The idea is to try and catch as much of relevant information to the users activity to minimize the need for new queries. The upside to this approach is that it allows for scaling without the need for storing duplicated data and low latency at the front-end. A downside to this method is that the catching needs to implemented in a smart way so it does not consumes all the user memory or it stores to little to actually work.

We choose to use catching because the structure of application fits the general concept of catching in that if e.g. the user looks for different cities in the city table, where it ideal to store all the cities at the user. With this choice we also evade the issue of have to deploy multiple databases needed for the other methods.

10.2 Implementation of Content-Blind Caching

The use the content-blind caching on our product, a structure should be made, so the code of the application can easily be copied out to different servers. A central database should be accessible for the servers, which will require us to make a server structure, which can ensure this connection (or write an API). Our servers should store each query, which makes is possible only to fetch data once from the central database, for each query. As our data does not change much, it is considered safe to store the data from the query. This could for example be stored in XML files or similar. An implementation on the servers could be, is to implement a library, trough which all database query must be made. It checks if the data is present in the XML files, if not it request the data from the central database. All modification query must be made directly to the central database.

Fault Tolerance and Transactions 11

11.1 2 Phase Commit Protocol

2 phase commit protocol (2PC) purpose is, to ensure every function in a group is evaluated, or no function is evaluated. This will ensure that if an error occurs in one function, the other functions are rolled back, so no wrong value is stored. An example of a use of this, could be an exchange of values, e.g. money for a product. If something goes wrong, so the money is not delivered, the product should not be sent. 2PC is implemented by a coordinator and a number of participants. The coordinator will coordinate the participants, starting by sending a prepare statement to each participants. If each of the participants return yes, the coordinator send the commit statement to each of the participants, which will commit their prepared log. An example can of this can be seen on **??**

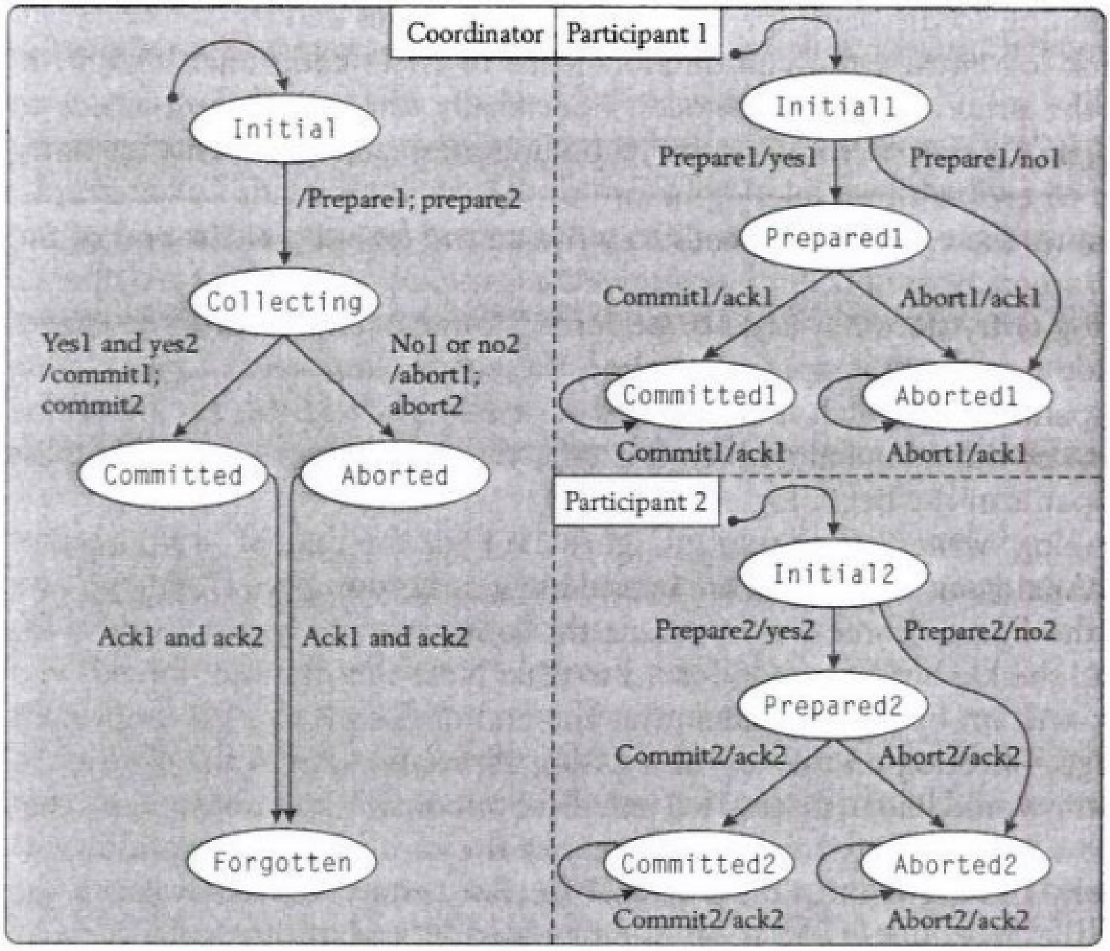

Figure 11.1: An example of a 2 Phase Commit Protocol

An implementation of 2PC in our project could be between the commercial site providing advertising and our site. The value exchange between these two sites could be validated by 2PC. Another solution could be in the back-end, if multiple tables should be updated. It could be validated that each table update would be performed correctly, and alternatively roll back the changes. In this case back-end would be the coordinator for 2CP, with the database as one of the participants.

11.2 Fault Tolerance

A web application usually utilize multiple servers in order to fulfill its designed purpose and this fact can lead to errors caused by the communication between the servers and services. These errors can cause a lot of problems if left unattended. The solution to this problem is to recover from them instead of trying to prevent them. To recover from this communication problems it is essential that the communication it done autonomously so the data it kept consistent. Transactions is used to obtain this property, which is known as a term in the world of databases.

So in order to decrease the fault rate and keep our data consistent in the web application we have to implement recovery. This is a consequence of having communication between multiple server and services. As earlier stated we utilizes the REST principle between the back-end and front-end which means that communication between this two ends already implements a form of autonomously in terms of the call to the back-end, but this does not take other communications into consideration. An example could the communication between the back-end and the database or the front-end and a server which provides the web application with some sort of relevant content. To implement recovery on these communication channels we have to choose the protocol we want to use. We selected the 2phases protocol because it provides the necessary recovery and it fits our communication setting, our application talks with less of 5 services at the time. Another solution would be to connect to the other servers directly through hardware but because of location this can become highly impractical.

For 2phased protocol there are different optimization strategies that solves some of the problems with 2phased protocol. The consequences of the choice of protocol would require additional logging because the system most have some sort of way to recreate its state after a crash.

11.3 Advantages and Disadvantages of 2PC

The advantages of 2PC is, that it will enable error recovery. If a participant produces an error, the coordinator would accept this, and tell other participants not to perform their actions. This will prevent the page for given an error or miscalculation. The problems with 2PC could be that it cannot guarantee a non-blocking behavior, which can make the system halt. It can also be hard to resolve an error, if the error occurs in the coordinator.

All in all, 2PC can be a very powerful tool in error recovery, but it is not perfect, and you should be clear of what to do if e.g. the coordinator produces an error.