Youssef Taoudi - yousseft@kth.se
11 januari 2018

Seminarie 4 - Rapport

Internet Applications, ID1354

Introduction

The task for this seminar was to implement a comment application using a javascript framework for the view model layer without using the DOM tree.

Literature

I mainly followed the knockout.js tutorial on the knockout website and their example programs.

Method

I started out by creating an application that could output, delete and enter comments without getting values from the server (all values were reset on reload) and then I used ajax to send and receive deleted and newly entered data to the server.

I used the knockout framework for this application. I did not use the MVC architecture. I learned how to use jquery and ajax to send and receive data from a server, how to encode PHP into son, how to user javascript frameworks and how to code in javascript in general.

Result

GITHUB: https://github.com/Taoudi/ID1354

I will divide this section into several sections that explain different functionalities in the program

HTML and Javascript:

```
//Giving the array values
for(var i = 0;i<=commentList.length -1;i++ ){

    if(typeof username !=="undefined" && username === userList[i]){
        this.comments.push(new Comment(commentList[i],userList[i],true ));
    }
    else{
        this.comments.push(new Comment(commentList[i],userList[i], false));
    }
};
```

Figure 1

Giving array values

Comments is an observable array and is given values from the server through an ajax function that is not included in the figure. The values added in figure 1 are then read from in the HTML as seen in figure 2. The array

```
<div class="textblock">
    <h1>User Comments</h1>
    <ul id="usercomment"  data-bind="foreach: comments">
        <li><p><strong data-bind="text: comment"></strong> - <strong data-bind="text: user">
        </strong></p>
              <button data-bind="visible: display(), click: $parent.deleteComment.bind($data,
              comment,user)" >Delete Comment</button>


        </li>
    </ul>
</div>
```

Figure 2

HTML that reads the elements from the array

```
this.deleteComment = function(post_comment, post_username){

      self.comments.remove(this);
      $.post("http://localhost/sem4/recept/DeleteComment.php", {
  comment: post_comment,
  username: post_username
);
    }
```

Figure 3

The delete function in the modelview

is iterated through and comments and username are binded to a strong element and are displayed on the webpage. For each entry where the logged in user matches the user variable in the comment object, a delete button is displayed which allows a user to delete the current comment through the delete function (figure 3). This is how observable arrays and values are used throughout the application, they are defined in javascript and can be displayed and interacted with through calling functions in js by events in HTML.

```
  this.enterComment = function (comment) {
$.ajax({
      url: "http://localhost/sem4/recept/UserInfo.php",
      async: false,
      dataType: 'json',
      success: function(data) {
          username = data.username;
      }
  });

      this.comments.push(new Comment(comment, username,true));
    $.post("http://localhost/sem4/recept/AddComment.php", {
  comment: comment,
  username: username
});
  };
```

Figure 4

Javascript for enterComment function

Sending and Recieving Data:

Data is sent and received by using AJAX. An example is when entering a new comment, we have to tell the server which comment has been entered and by which user. For example, in figure 4, we have an enterComment function that is called by a user through a submit form in HTML. We get the submitted comment as a parameter and the username of the user through an jquery ajax function from the server, or more specifically from the UserInfo.php file as seen in the URL. The data type is encoded as json from PHP. Now that we have the username and comment, we create a new Comment object and push it into the observable array so that it is added to the comment list that is displayed to the user. Now if we were to leave it as it is, the newly added comment would disappear on reload since it is not saved on the server, which is where we receive our data from. This means we have to send back the data to the server, through a jquery post function and add the new comment to the database.

Sending and recieving data from server is done several times in the application, for example when getting the initial values for the observable array, when deleting comment and when adding new comment.

All functionalities are inside a viewmodel which has two parameters, a comment list and a user list which holds the values for all comments in the database and they are fetched using a jquery.ajax.

```
var commentList = [];
var usernameList = [];
var page = "";
$.ajax({
        url: "http://localhost/sem4/recept/ListHandler.php",
        async: false,
        dataType: 'json',
        success: function(data) {
            for(key in data){
       $.each(data[key], function(key, val) {
           if(key=="comment"){
        commentList.push(val);
           }
           else if(key=="username"){
           usernameList.push(val);
           }
    });
        }
        }
    });

ko.applyBindings(new AppViewModel(commentList, usernameList));
```

Figure 5

Initial values and creating view model

Discussion

The assignment was to create a comment application using a view model in javascript without using DOM. The hardest part of this task was to use the ajax functions with jquery. I had a problem with the URL while using MAMP and I couldn't get the function to work. Eventually I fixed the problem by using Mac's native web server instead. Another problem I stumbled upon was that I could not use outside variables when calling a function from HTML. It seemed to be a scope issue so I fixed it by saving the keyword 'this' in a variable 'self'. It also took me a while to figure out how to bind data to functions in HTML

Comments about the course

It took approximately 20 hours to solve this task, mainly because I could not figure out how to set the correct URL.