# Exercises: Creating a Blog with HTML5, JavaScript and Kinvey

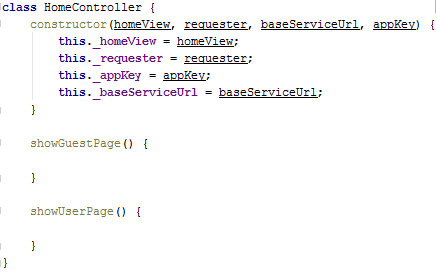
Problems for exercises and homework for the [“Software Technologies” course @ SoftUni](https://softuni.bg/courses/software-technologies).

If you follow the steps of this exercise, correctly, you will implement a simple blog SPA (Single-page Application) which can be of great use to you in the future.

Now that we have some of the app.js logic implemented and a Home View, it is time to create a Home Controller.

## Creating the controller class

We already know how to make classes, having already made a Home View class, so let’s use the same strategy here.

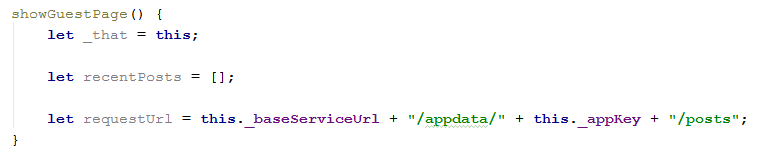


We have a constructor which accepts 4 parameters, a Home View, which is the view for the current controller. Controllers are linked to views in order to create the logic behind the MVC system. Currently we have no models, so we’ll just call it VC system. The controller also has a requester, because it will make requests. The requests need a url and appKey in order to build a request URL.

Apart from the constructor we have 2 functions, which define the logic for showing the guest page and the user page.

## Creating the logic of the show guest page function

Now, let’s create the **showGuestPage** functionality which is on the Controller side.

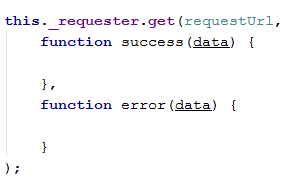


As we see we have a reference to the class again, externally, which means that in the near future we might need to use it in a function. We also have an array called recentPosts, which will hold the data of the recent Posts for the sidebar. Then we have a requestUrl, which is built from several strings.

## Making the request

We need to extract from the Kinvey database all the posts and render them. This is done by making a **GET** request.

The requester we have from the framework has a **get** function which defines logic for GET requests. The implementation of the function accepts 3 arguments – the requestUrl and 2 functions – success callback function, which will be executed if the request succeeds and an error callback function, which will be executed if the request fails.



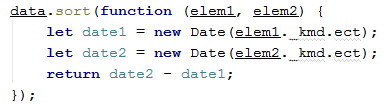
As we see both callback functions accept an object **data**, which is basically the response data. Let’s implement the error callback function because it’s easier.



The showPopup function is an element of the Notification Service implemented in the framework. Basically we want a notification popup of type **error** with the given text. That way we can inform the user that something went wrong with his request.

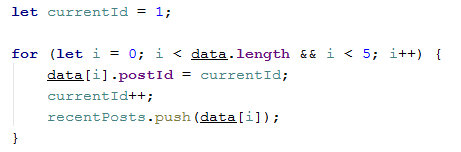
## Making the success callback function

Now let’s implement the success callback functionality. Assuming the request has succeeded, the data object should be an array of posts. We need to show those posts in order of being created, so let’s sort them.



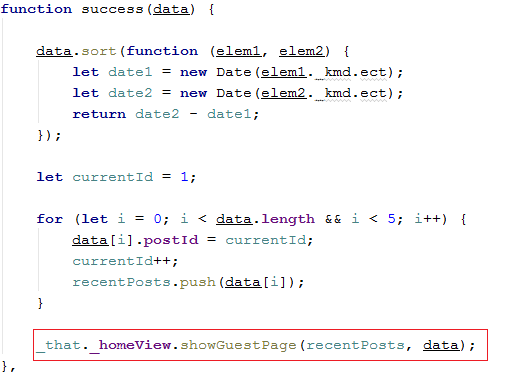
Using the sort function we are sorting two elements by date. Every Kinvey database object has a member called “**\_kmd”** which holds metadata. **ect** member from the **\_kmd** member is basically the time of creation of the current element.

Now we need also need to fill up the recent posts array. The recent posts array basically holds the 5 most recent posts, if there are that much. So let’s fill it up.



We’ve taken the 5 most recent posts of the sorted data array with posts, assuming there are more than 5. If there aren’t we just take them all. But what is this postId? Now, the meaning of the recent posts in the sidebar is for them to link to the posts on the main content, by clicking each of the recent posts it focuses your page view on the post that corresponds to it. You will see how that is done later.

Now that we have all the data needed, and it is processed, sorted and formatted correctly, we can pass it to the view which will render the page.



We pass the recentPosts filled up array and the data sorted array from the response to the view. With this we are basically ready with the guest home page functionality in the Home Controller.

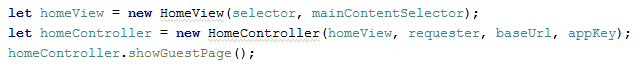
## Implementing the User Home page

Following the steps above, implement the **showUserPage** functionality. The only thing different is that you must pass extracted and processed data to the **showUserPage** function of the **view**.



## Revision

Now go to **app.js** and create an instance of the Home Controller class. Use the **showGuestPage** function to test your work till now.



If you’ve done everything correctly you should see the following:



