Computación Orientada a Web

Entrega Tema 4 & 5

UNIVERSITAT DE BARCELONA

Sebastian Andrade Zanotti - NIUB17692754

Barcelona, España, 02/05/2022

Apartado 1

Caracteristicas:

Consta de los mismos 3 archivos php del apartado anterior, el cliente, el servidor, y el data_base_connect que crea la conexión con la base de datos. El server.php no viene al caso en este apartado, porque no se modificó. Todos los cambios están en el cliente.php. El script de cliente.php básicamente consiste en el document.observe, que es básicamente el window.onload donde tenemos todos los listeners, los events handlers, y fuera del mismo tenemos las funciones a las que llama cada handler. En el document.observer tenemos declaradas varias variables que contienen el ID del form, el nombre del usuario, correo, etc… diferentes elementos del form. Esto actualmente no hace falta, pero como originalmente estaban asignadas como document.getElementByID(" "), era más cómodo tenerlas declaradas desde un principio.

Funcionalidades:

La función CheckName, verifica el nombre que introduce el usuario, comparándolo con un regexp definido en el onload. Esto es posible gracias al event listener que hay en onload que llama a CheckName cada vez que el valor de entrada “name” cambia, y pasa a la función el regexp correspondiente a name, que es el mismo regexp que usábamos en la entrega anterior desde el server. Si el nombre es incorrecto, el valor en el form se borra directamente, el campo de input se sacude, y se pone de color rojo por un par de segundos, indicando asi al usuario que su entrada es incorrecta. Originalmente solo se mostraba un alert(), pero me parecía más incómodo ya que había que constantemente hacer click en el mensaje. Aquí me curo en salud, y si el input es incorrecto lo descarto, y doy feedback visual al usuario de que está mal. La función de CheckMail hace exactamente lo mismo pero usando el regex del email. Por último está la función Checkform, que verifica el nombre y el email con su respectivo regexp. Esta función en realidad es innecesaria, porque su trabajo ya lo hacen las otras 2 funciones, pero originalmente no sabia que se podían despejar los valores de los inputs, entonces lo que pasaba era que al hacer checkName, sale el alert, pero si el usuario no cambiaba nada podía igual hacer submit. La función está actualmente, por si acaso un usuario tiene un autocomplete que llene todos los campos, y por algún motivo el event listener no detecte el cambio. Es una precaución extra más que una necesidad.

Comparación con el código ejemplo:

El código en general es bastante similar a los ejemplos, ya que se hicieron en orden. Primero implementamos el windows.onload, usando document.getElementByID(" ") en las variables, los handlers que verifican cambios usan “change” y el que verifica el form usa onsubmit. Luego cuando usamos scriptaculous reemplazamos todos los getElement por su respectivo $, y así.

El código no tiene ningún handler en el html y ejecuta los ficheros JS después del body para que no sea obstructivo. También todos los event handlers llaman a una función, no ejecutan directamente una función como alert(). El único posible riesgo es el uso de variables globales en mi caso con los regexp, pero entiendo que si está en el onload, no es una variable global, así que entiendo que no hay problema en declarar los regexp ahí.

Apartado 2

Caracteristicas:

En este apartado se cuenta con muchos más ficheros php que funcionan como servidores, los cuales son llamados desde el mismo cliente, cliente.php. Los php servidores son: server, getCities, getCollition, y gethint. Se explicarán más a detalle en funcionalidades. Data_base_connect.php sigue siendo un archivo requerido para conectarse con la base de datos igual que en la práctica anterior, con la diferencia de que ahora también conecta con la table de “actores” proporcionada en el campus para usarla con el autocompleter.

Old_code.php contiene fragmentos del código de cliente con funciones antes de que se cambiaran a Ajax con scriptaculous. Por tanto, este código solo está como referencia. Y ServerSideScript.php estaba para probar el ejemplo del autocompleter, y tampoco se usa.

Por otra parte, gracias al uso de Ajax, ya es posible implementar el dropdown dinámico en el que al elegir un país, modifica las opciones de elegir ciudad, en el mismo cliente.

Todas las funciones que hacen Ajax.request utilizan el metodo post para que no se manden los parámetros por url, por seguridad. A excepción del showHint, porque el autocompleter del ejemplo utilizaba

Funcionalidades y Comparación con el código:

Server.php se utiliza para mostrar los resultados de la reserva al cliente, y efectuar la reservación en la base de datos “reservations”, que está en world.

A diferencia del apartado anterior, el listener onsubmit se cambió por un onclick, ya que daba problemas si queríamos visualizar el resultado de la reserva en el mismo cliente, en lugar de usar una acción para navegar a otra página. De todas formas este listener onClick llama a la misma función checkForm, que al igual que en apartado anterior revisa que los inputs sean correctos comparándolos con su respectivo regexp. Con la nueva implementación del dropdown dinámico, se agregó una condición que verifica que el usuario haya seleccionado un país y una ciudad (sobre todo porque se permite dejar la opción vacía, que es como inicialmente se muestra en la página). Esto nos lleva a getCollition.php.

Server.php no se llama directamente. Cuando todo se cumple en checkForm, se hace un Ajax.request que llama a getCollition.php el cual verifica que no haya reservas ya existentes para el mismo usuario dentro de una fecha específica. getCollition.php retorna un error o un status correcto, que para deliberadamente se ejecute el onSuccess o onFailure del Ajax.request que ocurre en checkForm. Si no hay colisiones, se llama el onSuccess el cual llama a la función submitForm donde ocurre otro Ajax.request donde se llama a server.php para que se efectúe la reserva y se muestre el resultado en la web. Si ocurre un onFailure, se llama a la misma función failureFunc, donde dependiendo del error notificará un error general de la base de datos, o notificará al usuario que ya tiene una reserva de hotel existente, lo cual quiere decir que se llamó el onFailure desde el Ajax.request de checkForm.

Server.php es prácticamente el mismo código que se usó en apartados anteriores, pero solo la parte php, sin html.

GetCollition.php es una simple query a la base de datos que busca al usuario con su email, y verifica que la fecha de la reservación no sea 5 días antes o 5 días después de la fecha existente en la reserva (porque asumo que todas las reservas de hotel full package duran 5 días).

Las funciones de checkName y checkMail son iguales a las del apartado anterior.

ShowHint es la función que utilizamos para implementar el autocompleter de Ajax, el cual llama a getHint.php.

GetHint.php hace una query a la base de datos de actores de imdb_small, usando el input del usuario para que le muestre nombres similares al que está introduciendo. El php pone el resultado de la lista en forma de tabla <lu> para poder mostrarla con el autocompleter de Ajax. A diferencia del código ejemplo, el autocompleter se llama desde el listener, con “keyup” en lugar de desde el onload, porque lo interesante está en que muestre strings similares a los del usuario, y no una lista estática.

Por último está la función getCities, la cual llama a getCities.php donde se hace una query de la ciudades según el código del país, igual que en la entrega anterior. La diferencia es que utiliza un código similar al del gethint proporcionado en los primeros slides de la diapositiva, cuando el hint era un string y no una lista. Por lo que el php retorna un string con 6 ciudades separadas por comas. Con el onSuccess, se llama a la función getCities2, donde utilizando un for, separamos el String en cada coma usando split(“,”) para obtener cada ciudad y agregarla como nueva opción en el dropdown. Logrando así, un dropdown dinámico que depende del país que eliges.

Si desde la función getCities el usuario no elige un país, el dropdown se actualiza a “select city” que es una opción sin valor.