Práctica 6
JavaScript / AJAX / JSON / Boostrap

Descargue e importe el proyecto **TTPSJavascript.war** provisto por la cátedra y realice los siguientes ejercicios.

1. **Desarrollo en Javascript**: implemente las siguientes funciones Javascript para completar la funcionalidad de los distintos botones de la página. En todos los casos en que se modifique el modelo, antes de retornar deberá invocarse la función **dibujarModeloMensajes(unModelo)** para redibujar la vista.
   - **quitarUltimoMensaje()**: elimina el último mensaje del modelo.
   - **agregarMensaje()**: agrega un nuevo mensaje al modelo.

     Nota: analice las funciones **getAutorIngresado()** y **getMensajeIngresado()** provistas en **ttps.js** utilicelas para obtener el texto escrito en los inputs. También utilice la función **limpiarCampos()** para vaciar estos campos.

   - **habilitarDeshabilitarBotonesEliminar()**: modifica el atributo **puedeEliminarse** de cada uno de los mensajes del modelo. En caso de que los mismos estén en **false** (botones deshabilitados), los valores deberían ser seteados a **true**, caso contrario se setearán a **false**.
   - **eliminarMensaje(posición)**: elimina el mensaje que se encuentra en la posición pasada como parámetro. Para resolver este punto, puede consultar la documentación de los siguientes métodos Javascript para el manejo de arreglos:

     http://www.w3schools.com/jsref/jsref_indexof_array.asp

     http://www.w3schools.com/jsref/jsref_splice.asp

   - **asignarPuntajes()**: a cada objeto mensaje del modelo, se le añade un atributo **calificacion** con un valor numérico del **1** al **5**. El criterio de asignación de puntajes puede ser al azar.
   - **eliminarMensajesConPuntajeMenorA(unPuntaje)**: elimina del modelo aquellos mensajes que tengan una puntuación menor a tres.
   - **mostrarMensajesAgrupadosPorAutor()**: muestra los mensajes agrupándolos por autor.

     Nota: analice la función **dibujarModeloMensajesAgrupadosPorAutor(modeloAgrupado)**, la cual dibuja los mensajes (agrupados por autor) recibidos como parámetro de acuerdo al siguiente formato:

```
{
        autorString : 'un nombre',
        mensajes : [ mensaje, mensaje, mensaje ]
},
{
        autorString : 'un nombre',
        mensajes : [ mensaje, mensaje, mensaje ]
},
...
```

Realice una copia del proyecto **BlogDeMensajes** de la práctica 5 y realice los siguientes ejercicios

2. **Uso de AJAX y JSON**
    a. Escriba un servlet llamado **MensajesJSON**, que dado un requerimiento de tipo GET retorne los mensajes en formato JSON y para los requerimientos de tipo POST reciba, también en formato JSON, nuevos mensajes para publicar en el sitio.
    b. Modifique la página **visualizarMensaje.jsp** para que la misma cargue a través de AJAX los mensajes provistos por el servlet **MensajesJSON**. Agregue además la funcionalidad de mostrar los mensajes ordenados cronológicamente y agrupados por autor.
    c. Modifique la página **agregarMensaje.jsp** para que la misma realice la publicación de mensajes utilizando a través del servlet **MensajesJSON**

3. **Bootstrap**.
    a. Tenga abierto el sitio http://getbootstrap.com/css/ para ver ejemplos de las distintas componentes con el formato de bootstrap.
    b. En la carpeta **webapps** de su sitio agregue de ser necesario las subcarpetas correspondientes a bootstrap.
    c. Modifique la visualización de su sitio de modo que use la diagramación y los estilos definidos por Bootstrap.
    d. Verifique si su aplicación es “responsiveness”.