**Spring Framework**

Esta configuración solo es para hacer el DAO en nuestra aplicación

**La aplicación**

**El formulario**

Necesitamos nuestro formulario que tendrá los combos. Este será el siguiente:

<form>  
 <label for="productCode">Product code</label>  
 <select id="**productCode**" name="product" onchange="showProducts()">  
 </select><br/>  
 <label for="product">Product</label>  
 <select id="**product**" name="product"></select>  
 </form>

Note en cuenta los “id” de cada elemento. Estos son importantes para poder identificar cada objeto del formulario

Como se ve, ningún select tiene valores.

Primero, cargaremos los valores del primer combo. Esto se hace desde javascript en la misma página.

**La carga de datos**

Este cargará los valores del primer combo

$(document).ready(function(){  
 $("#**productCode**").load("ProductCodeServlet");  
 })

Esto hace lo siguiente: Cuando el objeto “document” (el documento del HTML) esté listo, llamará a la función que está inscrita.

Esta función hace lo siguiente: al objeto llamado “productCode” cargará los valores que están en el URL “ProductCodeServlet”. Este “ProductCodeServlet” es un servlet que habremos creado. Su método de atención es simple:

protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 ProductCodeDao dao = DaoFactory.getInstance().getProductCodeDao();  
 response.setContentType("text/html;charset=UTF-8");  
 PrintWriter out = response.getWriter();  
 out.println("<option value=''>Seleccione producto</option>");  
 List<ProductCode> list = dao.getProductCodes();  
 try {  
 for (ProductCode productCode : list) {  
 out.printf("<option value='%1s'>%2s</option>", productCode.getProdCode(), productCode.getDescription());  
 }  
 } finally {  
 out.close();  
 }  
}

Solo muestra los valores option que se mostrará en el combo de códigos. Estos datos se insertarán como por “arte de magia” en el combo “productCode”.

Ahora, cuando se seleccione un elemento del combo “productCode”, vemos que su atributo “onchange” está llamando a la función “showProducts()”. Ahora, veremos cómo es este método:

function showProducts(){  
 //obtiene los objetos productCode,   
 var code=$("#**productCode**").val(); //.. y se obtiene el valor  
 //llama al servlet con el parametro seleccionado  
 $("#**product**").load("**ProductServlet**", *{productCode:code}*)  
 }

Notar que ahora, la función load() para el objeto product, está recibiendo un argumento

{productCode:code}

Eso es, que va a llamar al servlet ProductServlet y enviará el parámetro “productCode” con el valor contenido en “code”.

Ahora, el servlet de ProductServlet, es el siguiente:

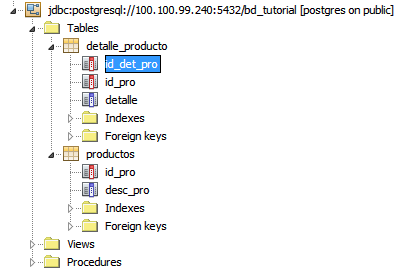
protected void processRequest(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 response.setContentType("text/html;charset=UTF-8");  
 PrintWriter out = response.getWriter();  
 String productCode = request.getParameter("productCode");  
 ProductDao dao = DaoFactory.getInstance().getProductDao();  
 List<Product> list = dao.getProductsByCode(productCode);  
 try {  
 for (Product product : list) {  
 out.printf("<option value='%1s'>%2s</option>", product.getProductId(), product.getDescription());  
 }  
 } finally {  
 out.close();  
 }  
 }

**Uso de Jquery con Ajax**

Hola otra vez por aqui retomando el blog despues de un largo receso, como comprenderan con la familia y el trabajo se nos hace dificil a veces publicar algun post, pero bueno manos a la obra.

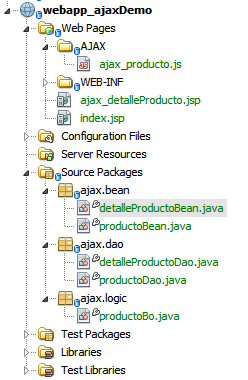
Este post es ha solicitud de unos compañeros que estan aprendiendo a trabajar con [jsp](http://es.wikipedia.org/wiki/JSP), [ajax](http://es.wikipedia.org/wiki/AJAX) y [postgresql](http://www.postgresql.org/) y necesitaban alguna guía práctica para trabajar la parte del llenado de combos de forma dependiente justamente utilizando [ajax](http://es.wikipedia.org/wiki/AJAX) y obteniendo la data desde una base de datos.

Bueno, lo primero que debemos tener en cuenta es la estructura de tablas en la base de datos desde donde obtendremos la informacion. Para este ejemplo tome como ejemplo la siguiente estructura.

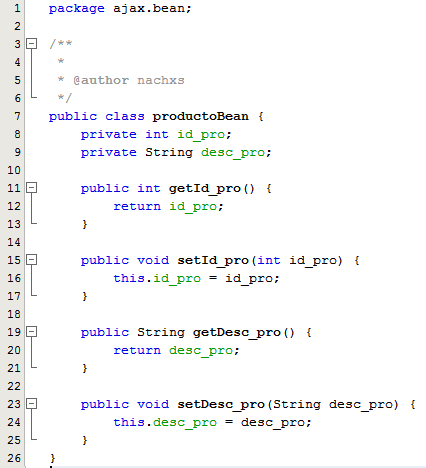
[](http://nachxs.files.wordpress.com/2009/03/tablas.png)

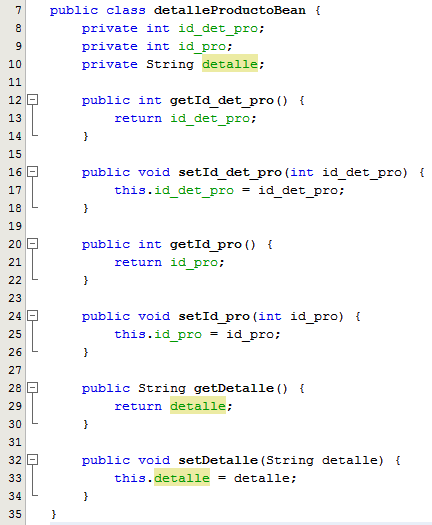
como podemos observar la base de datos se llama **bd\_tutorial** y las tablas serian **productos** y **detalle\_producto** respectivamente.

Una vez creado las tablas, ahora lo que hacemos es crear nuestro proyecto web al cual llamaremos **webapp\_ajaxDemo** y comenzar armar nuestra estructura de archivos jsp, javascript y clases java, las cuales accederan a la informacion de la base de datos. Creamos el proyecto y generamos la siguiente estructura propuesta:

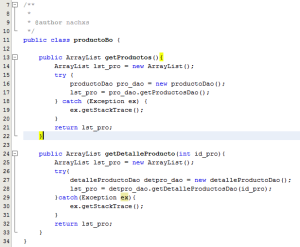
[](http://nachxs.files.wordpress.com/2009/03/proyecto.png)

Ahora, lo que haremos sera ver como obtener la informacion de la base de datos, para esto vamos a trabajar nuestras clases java empezando por los beans en donde almacenaremos la informacion  y terminando por las clases  DAO , las cuales seran las que recogeran la informacion de la base de datos.  Lo primero es tener la estructura de los archivos **productoBean.java** y **detalleProductoBean.java**, mas o menos seria la siguiente:

[](http://nachxs.files.wordpress.com/2009/03/producto1.png)

[](http://nachxs.files.wordpress.com/2009/03/detalleproducto.png)

Una vez detallada la clases Bean, ahora nos enfocamos en las clase BO y DAO, yo siempre hago esta distribucion para tener en cuenta el manejo de capas, esto ayuda mucho. Ahora veremos la estructura de la clase **productoBo.java**, aqui es donde definiremos los metodos que tendran acceso a los métodos de las clase DAO.

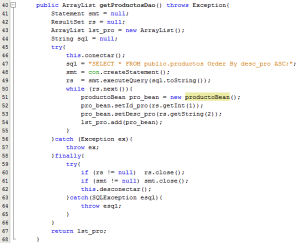
[](http://nachxs.files.wordpress.com/2009/03/productobo.png)

Los métodos que usaremos **getProducto** y **getDetalleProducto** los cuales nos devuelven como resultado un **ArrayList** de Beans esto con la finalidad de poder tener toda la informacion y poder seleccionar lo necesario. Ademas hacen referencia a métodos de las clases DAO la cuales ya tenemos definidas.

Ahora, veremos las estructuras y codificacion de las clases  que obtienen la información de la BD **productoDao.java** y **detalleProductoDao.java** respectivamente:

Para las clases **productoDao.java** y **detalleProductoDao.java,** por cuestiones de espacio solo estoy mostrando los métodos que acceden a la información, se asume que ustedes ya tienen definido los métodos de conexion y desconexion respectivamente.

**productoDao.java:**

[](http://nachxs.files.wordpress.com/2009/03/productodao.png)

**detalleProductoDao.java:**

**[](http://nachxs.files.wordpress.com/2009/03/detalleproductodao.png)**

Una vez ya completada la estructura y codificacion de las clases java, ahora pasaremos a codificar las paginas JSP**,** lo primero que haremos sera mostrar el primer combo que serian los productos, para esto mostraremos una parte de la codigo del archivo **index.jsp** la cual sufrira alguna moficiacion mas adelante, por el momento tenemos lo siguiente:

**[](http://nachxs.files.wordpress.com/2009/03/combo_1.png)**

Fijemos en los recuadros de color rojo, en el primer recuadro lo que hago es importar la clase **productoBo** y **productoBean**, dado que en el primero se tiene el metodo que obtendria la informacion de los productos, y en el segundo recuadro es donde muestro los productoslos cuales son almacenados en beans del tipo **productoBean** por eso la declaracion en el primer recuadro.

Deben tener en cuenta que en la linea de código en donde muestro el detalle del producto en la opcion value el combo se esta poniendo el id del producto, esto para que cuando mandemos el producto se tome el id para poder realizar la busqueda en el detalle y no el nombre del producto.

Ahora veremos la parte de AJAX, para esto tenemos un archivo javascript **ajax\_producto.js** en el cual tendremos el siguiente código:

[](http://nachxs.files.wordpress.com/2009/03/ajax_producto1.png)

Vemos que estamos declarando la funcion **getDetalleProducto** al cual le estamos enviando como parametro id\_pro, luego en la parte mas abajo vemos que hacemos la referencia a la pagina **ajax\_detalleProducto.jsp**, que en la que recuperaremos la informacion de la tabla de detalles enviando como parametro el id\_pro y ademas un identificador para hacer refresco a nuestras busquedas, esto por un problema en algunos navegadores con el manejo de su cache. Y la funcion **resultado\_detalle** que es donde le indicamos donde mostrarnos el resultado.

Fijarse bien en esto ultimo ya que como veran estamos utilizando el identificador del elemento en donde pondremos el resultado que para este caso es **result\_detalle.**

una vez terminado esto lo que haremos sera codificar la pagina que obtendra la informacion del detalle de producto, que en este caso es **ajax\_detalleProducto.jsp**

**[](http://nachxs.files.wordpress.com/2009/03/detalle_ajax.png)**

Ahora solo nos queda adecuar la pagina **index.jsp** para que pueda hacerse la invocación, mostraremos el codigo segmentado por cuestiones de espacio.

[](http://nachxs.files.wordpress.com/2009/03/index_1.png)

En este primer segmento, podemos ver que aqui se hace la llamada a la libreria javascript **ajax\_producto.js** la cual contiene a la función **getDetalleProducto,** fijense en el evento **onchange** del combo, alli es donde llamamos a la funcion que retornaria la informacion del detalle.

Ahora para el segundo segmento vemos el **id=”result\_detalle”** que es el identificador que se utiliza para mostrar el combo con el detalle.

[](http://nachxs.files.wordpress.com/2009/03/index_2.png)

Y practicamente esto seria todo. Se debe tener en cuenta que esto se hizo de forma rápida solo para cuestiones de enseñanza, esto se puede mejorar notablemente tal vez utilizando algun framework como [jQuery](http://jquery.com/) o mejorando el código propuesto. Espero les sea de utilidad para algunos que recien empiezan con esto