## Clases y controladores:

Las clases y controladores que utiliza webStore son clases que gestionan la funcionalidad, casi por completo, de la web. En la mayoría de los casos utilizan funciones estáticas para poder ser llamadas sin instanciar la clase. Su principal diferencia es que utilizamos las clases para representar objetos cargando datos desde la base de datos, mientras que los controladores son clases que afectan al comportamiento de la web, a ficheros y a otros elementos que no tienen nada que ver con la base de datos. Las clases están definidas como súper clases para poder ser extendidas después. Así si tenemos la clase SuerArticulo, podemos extenderla después en la clase Articulo que será la que se utilizará. **Nunca** se utilizan las súper clases de manera directa. WebStore buscará la clase sobre escrita en su carpeta correspondiente, en caso de no encontrarla creará automáticamente la clase y la incluirá en la web. La carpeta de sobre escritura se almacena en la variable ***\_WS\_OVERRIDE\_DIR\_*** y dentro de dicha carpeta tendremos la carpeta “classes” que almacena las clases y “controllers” que almacena los controladores

* Clases:
  + Ruta de la carpeta: *\_WS\_DIR\_*/core/classes/
  + Variable de sistema: *\_WS\_CLASSES\_DIR\_*
  + Carpeta de sobre escritura: *\_WS\_DIR\_*/override/classes/
* Controladores
  + Ruta de la carpeta: *\_WS\_DIR\_*/core/controllers/
  + Variable de sistema: *\_WS\_CONTROLLERS\_DIR\_*
  + Carpeta de sobre escritura: *\_WS\_DIR\_*/override/controllers/

### Funciones remotas

Desde JavaScript podemos ejecutar funciones remotas definidas en las clases. Para ello tienen que estar declaradas como estáticas y públicas y su nombre debe empezar por “static”. Además el nombre de la función debe ser “camel case”. De esta forma, si queremos obtener los datos de un producto, tendríamos que crear una función dentro de la ficha de esta manera:

public static function remoteDatosProducto($data){

………………..

}

Esta función recibe la variable $data que es un objeto que contiene en sus propiedades las variables recibidas vía javascript.

Para llamar a estas funciones remotas, debemos componer un objeto Json con la información que vamos a mandar, la clase/controlador a llamar y la función remota a ejecutar:

var dataToSend = {

controller:"CabeceraCesta", //clase o controlador a utilizar

method:"emtpyCart", //Método a llamar

data: data, //Datos a enviar

};

Vemos que el método no lleva la palabra remote ni la primera letra en mayúscula.

El campo “data” es un objeto que contiene los campos con su respectivo valor.

Esta información ha de enviarse al servidor como texto plano. Para ello convertimos el objeto JavaScript a Json y lo pasamos a base64

Cuando enviemos por ajax la información, recibiremos la respuesta en el mismo formato, en base64 y Json.

## Eventos Javascript

### Evento de carga

Se declara un evento llamado “WSLE” (Web Store Load Event) que es el encargado de lanzarse cuando se haya cargado la página. Para asignarlo haremos lo siguiente en javascript:

document.addEventListener(‘WSLE’, function(){

});

Y dentro de la función especificaremos el código a ejecutar

## Controladores

### ArmiSync

Es el encargado de realizar la sincronización con ficheros generados por programas Arminet. Al margen de sobrescribir métodos propios de esta clase podemos personalizar la lectura de los ficheros teniendo en cuenta estas 2 funciones:

1. *nombreTabla*\_procesarFichero($ruta\_del\_fichero, $linea\_de\_inicio);
2. post\_*nombreTabla*\_procesarFichero($ruta\_del\_fichero, $linea\_de\_inicio);

En *nombreTabla* indicaremos el nombre de la tabla de la Base de Datos que estamos procesando. Si escribimos la primera función para una tabla concreta, se ejecutará esta función en vez de la función nativa (por defecto se usa en la tabla categorías-productos). En la segunda función podemos lanzar un proceso que se ejecutará justo después del procesado de la tabla (por defecto se usa para reindexar productos después de haberlos leído).