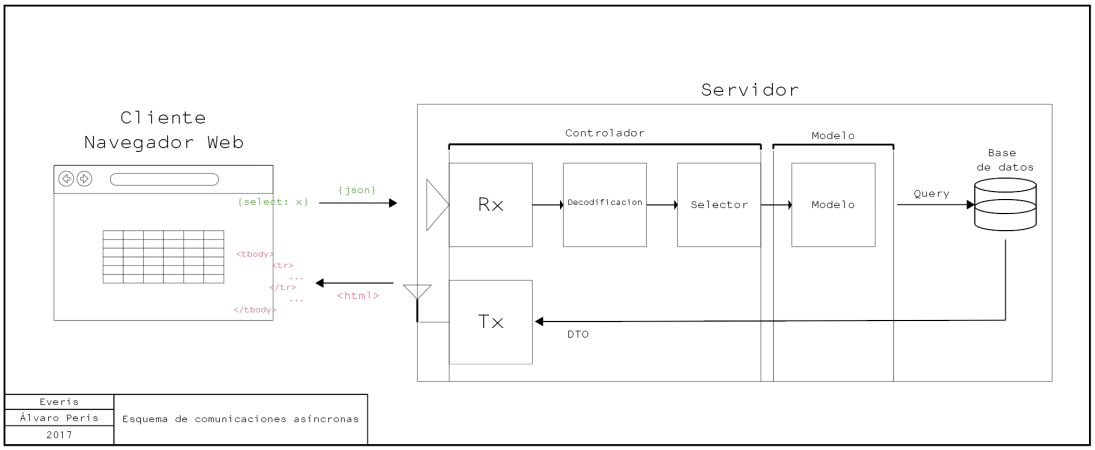
Peticiones AJAX en Spring

Álvaro Peris Zaragozá
EVERIS
CBPCLO

2 de agosto de 2017

1. Esquema de comunicaciones

Este escrito trata de explicar la fraccionación de grandes descargas de información desde el servidor y no saturar al cliente en el intercambio de información.



Realizaremos peticiones asíncronas, donde transmitiremos mensajes del estilo

```
{funcion: "nombreFuncion", parametros: "MisParametros"}
```

- **Rx** : Se trata del receptor encargado de la gestión de la petición *GET* o *POST* del cliente.
- **Decodificador**: Extrae el significado del mensaje en crudo transmitido.
- **Selector:** Se encarga de realizar la petición adecuad a la base de datos.
- **Tx:** Retorna la información al cliente en formato <*html*>

2. Parte de AJAX Cliente

En la parte de *AJAX* realizamos las peticiones peticiones al backend desde el cliente.

Sigue la siguiente estructura:

```
$(document).ready(function(){
        $(".aj").click(function(event) {
                // Llamamos a AJAX
                peticionAJAX();
        });//En click

        function peticionAJAX(){
                $.ajax({
                                type: 'GET',
                    url : 'ajax/' + '{valor1:pepe, valor2:perez, valor3:
                        pee@everis.com}',
                    dataType: 'text',
                    success : function(data) {
```

```
                    $('table').append(data);
                }
            });
             console.log("Llamamos a AJAX");
      }//Fin de la peticion AJAX
});
```

Para facilitar la gestión en el *Spring* emplearemos el método *HTTP GET*.

3. Parte de JAVA Servidor

En la parte de *JAVA* tendremos dos *@RequestMapping* uno donde realizará las peticiones el usuario y otra opaca donde se realizarán las peticiones de *AJAX*.

3.1. Página principal

```
//P GINA PRINCIPAL =========>
        //Petici n dela p gina principal
        @RequestMapping(value= "/principal")
        public String Principal(ModelMap model, HttpServletRequest request) {
                request.setAttribute("enlaces", "Mis Enlaces");
                return "views/ajaxx";
        }
```

3.2. Página de AJAX

En la transferencia de información es habitual emplear XML o JSON. Emplearemos JSON pero será necesario *parsearlo*. Podemos añadir librerías para procesar el JSON, no vienen de serie en JAVA y hay que añadirlas, para no interferir con el paradigma organizativo de *AbsisCloud* implementaremos una pequeña función de procesado decodificadora que retornará un *HashMap*.

```
        //GESTI N DE PETICIONES =========>
        //Petici n que gestiona las recepciones as ncronas de AJAX, se emplea
           de soporte
        // para la p gina principal
        @RequestMapping(value= "/ajax/{valor}", method = RequestMethod.GET)
        @ResponseBody
        public String Ajax(@PathVariable String valor)throws IOException {
                HashMap<String, String> req = Decoder(valor);
                String ret ="<tr><td>"+ req.get("valor1") +"</td><td> "+
                   req.get("valor2") +" </td><td>"+ req.get("valor3")
                   +"</td></tr>";
                return ret;
        }

        //Pasa de un elemento JSON a un HashMap
        public HashMap<String,String> Decoder(String value){
                value = value.substring(1, value.length()-1); //Quitamos los
                   braquets
                String[] keyValuePairs = value.split(",");  //Hacemos un split
                   con coma
                HashMap<String,String> informacion = new HashMap<String,
                   String>();

                for(String pair : keyValuePairs){
                    String[] entry = pair.split(":"); //Hacemos un split por
                       los dos puntos
                    informacion.put(entry[0].trim(), entry[1].trim());
                       //Quitamos los espacios blancos
                }
                return informacion;
        }//End decoder
```

4. Ejemplo de Aplicación: Paginador