



AUT08_01 Introducción a JAX-RS

1. Crear una aplicación Maven > Java EE 7, con servicios JAX-RS que serán consumidos por un cliente JQuery/Json

- 1 El fichero de subida es AUT08_01_Apellido1Apellido2Nombre.zip
- 2 Crear proyecto Maven > Java EE 7, Wildfly. El nombre del proyecto **ActorsRs_ApellidoNombre**
- 3 El contexto de la aplicación será ActorsRs_AN (A Inicial del apellido, N Inicial del Nombre)
- 4 Crear un recurso JAX-RS que se mapeará sobre el componente de URL **actors (@Path)**

4.1 Cread una clase POJO para almacenar los datos del actor (vale una que tengáis ya hecha)

En mi caso Actor Que sea JavaBean!!!

Nota: Los Beans son tipos especiales de POJOS. Hay algunas restricciones en POJO para que puedan ser Beans.

1. Todos los JavaBeans son POJO pero no todos los POJOs son JavaBeans.
2. Serializables, es decir, deben implementar la interfaz Serializable. Aún algunos POJOs que no implementan interfaz Serializable se llaman POJOs porque Serializable es una interfaz de marcadores y por lo tanto no de mucha carga.
3. Los campos deben ser privados. Esto es para proporcionar el control completo en los campos.
4. Los campos deben tener getters o setters o ambos.
5. Un constructor no-arg puede existir en un bean.
6. Se accede a los campos sólo por constructor o getter o setters.

4.2 Crear una clase POJO para el recurso REST

Ejemplo: ActorRest, en un packageservices.rest

4.3 Anotar la clase con @Path (“actors”)

(os pedirá que configuréis REST según Java EE 6)

4.4 Cread un método read o similar, que devuelva la lista completa de actores de tipo.

Para la primera prueba devolved unos objetos fijos en una lista tipo String.

4.5 Probad con petición GET a la URL y luego probad con JQuery AJAX (montón de ejemplos en StackOverflow)

4.5.1 Para añadir JQuery de forma muy simple, creamos un index.html y enlazamos en el HEAD la URL remota de JQuery:

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js"></script>
```



- 4.5.2 En el <body> de nuestra página, al final, añadimos un bloque de <SCRIPT> y una función JQuery en el document.ready que llamará al método JAX-RS

Nuestra función simplemente invocará al método JAX-RS de nuestro recurso, obteniendo la lista de actores. Con esa lista generará los elementos de una que tenemos en la página index.html. Esta , en el body de nuestra index.html, tendría la pinta:

```
<ul id="actorList" style="float: right"></ul>
```

El código Javascript

```
<script type="text/javascript">

    function refreshActorList () {

        var custList = $('#actorList');
        custList.empty();

        $.ajax({
            "url": "webresources/actor/",
            "type": "get",
            "dataType": "json",
            "success": function (actors) {
                //console.log(actors);

                $.each(actors, function (i, actor) {
                    var li = $('<li/>')
                        .addClass('ui-menu-item')
                        .attr('role', 'menuitem')
                        .appendTo(custList);

                    var a = $('<a/>')
                        .addClass('ui-all')
                        .text(actor.firstName + ' ' + actor.lastName)
                        .appendTo(li);

                });
            }
        });

        $(function () {
            refreshActorList();
        });
    }

</script>
```



2. Añadir un nuevo método al recurso JAX-RS actors, que en vez de devolver todos los actores, devuelva uno sólo, pasando su actorId por la URL.

1. Para hacer la prueba, simplemente invocamos la URL a través del navegador. Para la primera prueba devolved un objeto fijo. Por ejemplo: `return new Actor("Paco", "Martínez")`
2. Anotad el método para GET, que coja el parámetro y que produzca JSON

3. Crear un nuevo método que permita recibir un actor vía petición POST desde Javascript

En este caso vamos a utilizar un formulario en la misma página index.html. Cuando demos al botón submit del formulario se enviaría el dato y debería refrescarse la lista de actores

Actores

Nombre	<input type="text"/>
Apellidos	<input type="text"/>
Ajax	<input type="button" value="Submit"/>

- Juan López
- Pepe Reina
- gfd gdfg
- Félix Fuente
- Suzzane Smith
- cxzz czxc
- Luis Olivier
- Luis Olivier
- Kyle MacLahan
- Pepe Suárez
- Fernando Salazar
- Fernando Salazar

3.1 Ejemplo de Javascript, asociado al click del botón de un formulario

```
$("#cmdSubmit").click(function (event) {  
    var actor = {  
        firstName: $("#txtNombre").val(),  
        lastName: $("#txtApellidos").val()  
    };  
  
    var request = $.ajax({  
        url: "webresources/actor/",  
        type: "POST",  
        contentType: 'application/json',  
        data: JSON.stringify(actor),  
        dataType: "json",  
        success: function (data, textStatus, jqXHR) {  
            refreshActorList();  
        }  
    });  
});
```