



AUT07_01 Introducción a JAX-RS

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

- 1 Crear proyecto Maven > Java EE 7. El nombre del proyecto puede ser SakilaRs
- 2 Crear un recurso JAX-RS que se mapeará sobre el componente de URL **actors**

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

En mi caso Actor o ActorVo (vo de value object-). 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.

2.2 Crear una clase POJO para el recurso REST

Ejemplo: ActorRest, en un packageservices.rest

2.3 Anotar la clase con @Path (“**actors**”)

(os pedirá que configuréis REST según Java EE 6)

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

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

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

2.5.1 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>
```

2.5.2 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 devolvemos un objeto fijo. Por ejemplo: `return new ActorVo("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"/>	• Juan López
Apellidos <input type="text"/>	• Pepe Reina
<input type="button" value="Ajax"/>	• 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();
        }
    });
});
```

4. Sustituir los resultados mock(fijos) por llamadas a los DAO que teníamos en Sakila.



5. Corrección de BUG al serializar a JSON con GLASSFISH 4.1.1

https://bugs.eclipse.org/bugs/show_bug.cgi?id=463169

5.1 Pasos de solución

1. En el enlace anterior descargar el MANIFEST.MF corregido (está en el attachment titulado: manually edited manifest, added missing packages (35.94 KB, application/octet-stream))
2. Hacer una copia de seguridad del JAR **org.eclipse.persistence.moxy.jar**

Está en C:\Program Files\glassfish-4.1.1\glassfish\modules

El nuevo nombre podría ser: org.eclipse.persistence.moxy.jar.**original_20170129**

NUNCA dejar la extensión .jar en la copia
3. Copiar el JAR incorrecto org.eclipse.persistence.moxy.jar a una carpeta de trabajo, para modificar su MANIFEST

En mi caso lo copio a C:\Temp
4. Abrir el JAR con 7zip, por ejemplo, y entrar en la carpeta META-INF
5. En esta carpeta sustituir el fichero MANIFEST.MD por el corregido en el parche
6. Cerrar 7zip y copiar de nuevo el JAR corregido a C:\Program Files\glassfish-4.1.1\glassfish\modules

6. Corrección de BUG cuando se usa GLASSFISH 4.1.1 e Hibernate 5

<https://github.com/payara/Payara/issues/554>

<http://stackoverflow.com/questions/34813782/hibernate-5-glassfish-4-1-1-java-lang-nosuchmethoderror-org-jboss-logging-lo>

6.1 Pasos de solución

1. Parar GlassFish
2. Descargar la última versión de **jboss-logging.jar**. Está en Maven en:

<https://mvnrepository.com/artifact/org.jboss.logging/jboss-logging>
3. Hacer una copia de seguridad del JAR actual que está en C:\Program Files\glassfish-4.1.1\glassfish\modules

El nombre podría ser: jboss-logging.jar.**original_20170129**

NUNCA dejar la extensión .jar en la copia
4. Copiar el nuevo JAR **jboss-logging-3.3.0.Final.jar** a la carpeta:

C:\Program Files\glassfish-4.1.1\glassfish\modules
5. Arrancar Glassfish