**Preparación**

Comenzamos el proyecto, para esta segunda entrega, la idea es tener lista una versión mínimamente funcional, con las tres aplicaciones consumidas.

Instalamos la plantilla del proyecto en Eclipse y empezamos a trabajar, primera aplicación a integrar: **OMDB**

**OMDB**

La integración es bastante sencilla ya que contamos con un ejemplo muy parecido en la práctica 5.

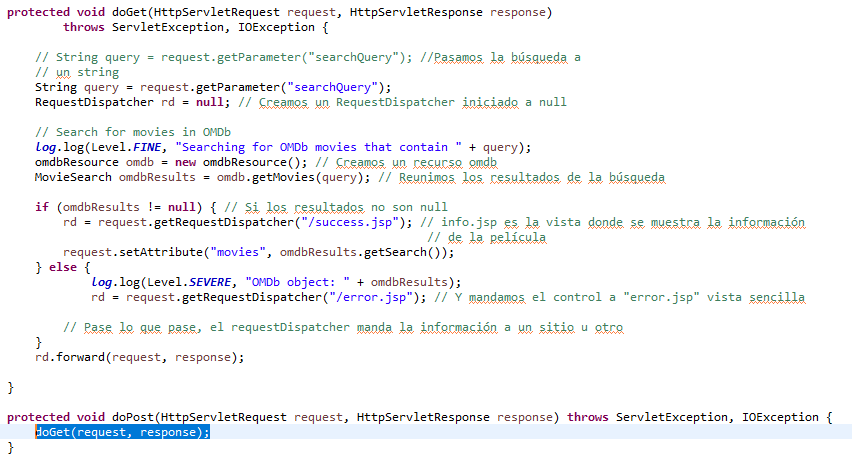
La principal diferencia es el modelo a usar, ya que nosotros queremos obtener más información que la recuperada en dicha práctica, por lo que el modelo principal de película (llamado **Search**) queda de la siguiente manera:

También obtenemos una clase adicional para implementar **Rating** ya que se trata de una lista con dos valores asociados (parecido a un map: *source* y *value*)

Además, tenemos el modelo de búsqueda **MovieSearch** que nos permite navegar a través de los resultados de la búsqueda.

Las vistas actualmente son idénticas a las del ejemplo, solo se han usado para comprobar que el controlador consumía los servicios y mostraba el título, una especie de test.

El controlador es lo único que ha dado problemas, ya que era incapaz de conectar con la vista **success.jsp,** tras muchas vueltas, el error residía en que al contrario que en la práctica, el Servlet creado para actuar de controlador, no contaba con una llamada al método **doGet**, mientras que al copiar la vista **index.html**, el método que pasaba al controlador era un **“post”**, por lo que no llegaban a entenderse. Se puede resolver de dos formas, añadiendo la llamada a **doGet** en el método **doPost** del controlador y/o cambiando el “**post**” por el “**get**” en **index.html**, he optado por la primera, quedando el servlet **searchController** de la siguiente manera.



De este consumo de API queda montar para que obtenga solo un resultado, y sea de este único resultado del que de toda la información. Una forma de implementarlo sería buscar por título con el título exacto o buscar por palabra como ahora, pero crear una vista a la que haría referencia cada resultado con su propia información.

La primera opción es más fácil de implementar y ahorraría trabajo.

La segunda opción lo haría mucho más intuitivo para el cliente.

Una vez acabada esta parte, no habría que implementar nada más de la API de Omdb.

**Youtube**

El consumo de la aplicación de youtube parece bastante más complejo, tengo un resumen básico de sus propiedades, parámetros y métodos que tengo que transcribir todavía, esta noche me pondré a ello.

Voy a empezar con lo que creo que es el modelo de búsqueda.

En la siguiente dirección se encuentra una “guía” del recurso **Search Result**, y acontinuación otra dirección donde se pueden hacer llamadas de prueba a la API para obtener el JSON de ejemplo.

<https://developers.google.com/youtube/v3/docs/search> (Guia del recurso Search)

<https://developers.google.com/youtube/v3/docs/search/list?apix_params=%7B%22part%22%3A%22id%22%7D> (Guía y pruebas de API de list Search)

Obtengo un JSON del consumo de **Search Result** con

Part = id (luego podemos indicar que id.videoId para solo obtener videos)

Type = Video (necesario para obtener únicamente videos)

videoEmbeddable = true (muestra solo los videos que se pueden insertar)

Obtengo un JSON del consumo de **Video** con:

Part = Player (Obtenemos datos para insertar un video)

Id = id (usamos como filtro del video su id)

Obtenemos así dos modelos:

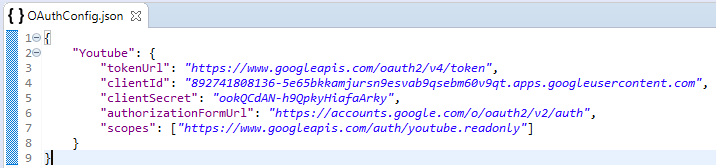
* + - **youtubeSearch**
      * **Id**
      * **Item**
      * **PageInfo**
      * **YoutubeSearchId**
    - **youtubeVideo**
      * **Item**
      * **PageInfo**
      * **Player**
      * **VideoPlayerById**

La idea es:

Mediante **index.html** obtener el parámetro de búsqueda, OMDB ya está configurado para obtener todo lo que queremos, y youtube, mediante ese mismo parámetro, obtendrá primero la id del video mediante **YoutubeSearch** (recibe un parámetro de búsqueda y devuelve la id del video) y con esa id, insertar el vídeo en la vista mediante **YoutubeVideo** ( recibe un id de un video y devuelve el código embed del propio video, sabemos que el video tendrá código embed porque SearchResultId solo devuelve videos **videoEmbeddable = true** )

Continuamos con los controladores, un primer vistazo del recurso es este, falta probar si funciona correctamente.

Adjunto un nuevo archivo con la configuración JSON en la carpeta *src/main/resources*

**