Práctica 1 AngularJS

Diagramas de arquitectura para contextualizar los ejercicios:

https://drive.google.com/file/d/0B_7-fxyOFEa7dEtUU2k2d1BvME0/view?usp=sharing

Diagrama de arquitectura AngularJS detallado en el lado izquierdo. La aplicación HTML 5 y AngularJS se ejecuta en el navegador. En el lado derecho aparece el servidor Spring desplegado en una plataforma en la nube. Ambos componentes, la capa cliente HTML5 y AngularJS, y la capa servidora Spring se comunican mediante un API REST (HTTP para el transporte y JSON para el formato de los datos):

https://drive.google.com/file/d/0B_7-fxyOFEa7Z0dlZ1BkaGxPTTQ/view?usp=sharing

Podéis desarrollar los primeros 6 ejercicios con el siguiente servidor HTTP ultraligero:

https://www.npmjs.com/package/serve

A partir del ejercicio 7 trabajaréis con el servidor Spring. Tenéis que almacenar vuestros ficheros HTML (con los scripts JavaScript/AngularJS) en la carpeta webapp de la aplicación JHipster (proyecto generado AngularJS/Spring). Después, tenéis que copiar los primeros seis ejercicios también en la carpeta webapp para tenerlos todos en un mismo proyecto (y repositorio de Github).

Tenéis que tener vuestro proyecto AngularJS/Spring con todos los ficheros (desde el ejercicio 1 hasta el final) que habéis programado de manera manual para implementar estos ejercicios, perfectamente subido a Github. Para la corrección de los ejercicios os pediré que pongáis en marcha el servidor y os haré preguntas sobre la implementación que habéis desarrollado. En el mismo repositorio de Github podéis almacenar en una carpeta un fichero PDF con la respuesta a las preguntas teóricas. Me tenéis que entregar la URL de vuestro repositorio de Github

Para desarrollar, solventar errores de programación y entender mejor tus aplicaciones AngularJS se recomienda el siguiente complemento para el navegador: http://ng-inspector.org/

Para desarrollar esta práctica se recomienda leer con detenimiento el siguiente tutorial público y de libre distribución: http://www.w3schools.com/angular/

También lo podéis complementar con la siguiente documentación oficial:

https://docs.angularjs.org/api

1. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_intro_bind

Desarrolla una página web equivalente que permita introducir los campos de un jugador de baloncesto (te puedes basar en los atributos que hemos utilizado con Spring), y muestre en tiempo real el valor introducido. Explica, de manera muy resumida, el mecanismo que utiliza AngularJS para sincronizar los elementos de la interfaz de usuario HTML (explica la directiva específica que se utiliza en este ejemplo).

2. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_intro_controller

Extiende el ejercicio número 1, de manera que se incluya un controlador que gestione los datos de la vista. El controlador ha de proporcionar unos datos iniciales por defecto que se mostrarán en la vista. Cuando el usuario actualiza los elementos de la vista (los elementos de la interfaz de usuario en HTML), ¿se actualizan automáticamente los datos en el controlador? Justifica tu respuesta.

3. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_expressions_objects

Utilizando la directiva ng-init, inicializa un objeto jugador de baloncesto con sus diversos atributos. Posteriormente, muestra sus atributos en diversos elementos HTML mediante la directiva {{}}. Explica en qué componentes de AngularJS se debe manipular el DOM: directivas, controladores, vistas o servicios. Justifica tu respuesta.

4. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_repeat_object

Inicializa un array para almacenar diversos jugadores de baloncesto. Posteriormente, utiliza la directiva ng-repeat para mostrar todos los jugadores de baloncesto con sus diversos atributos. Explica cómo funciona la gestión de scopes con la directiva ng-repeat.

5. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_filters_orderby

Inicializa un array que almacene diversos jugadores de baloncesto en un controlador. Posteriormente, utiliza la directiva ng-repeat para mostrar todos los jugadores. Utiliza también un filtro, para mostrar los jugadores ordenando mediante varios atributos (por ejemplo, nombre, total canastas, total asistencias, país, etc.). La idea es que puedes experimentar con

diversos atributos para comprobar que la ordenación funciona correctamente con todos los tipos de datos. Justifica en qué casos es preferible realizar la ordenación en el cliente mediante AngularJS, y en qué casos es preferible realizar la ordenación en el servidor mediante Spring. Explica ventajas e inconvenientes de cada enfoque.

6. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_scope_sync

Repaso sobre la sincronización automática que realiza angular entre la vista, el controlador y el modelo. Experimenta con el siguiente código de ejemplo, comprobando la sincronización que se efectúa con los atributos de un jugador de baloncesto.

7. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_customers_json

Desarrolla una aplicación similar que obtenga un listado de jugadores de baloncesto de un API REST Spring. En primer lugar, debes programar el API REST Spring que devuelva el listado de jugadores de baloncesto, tal y como hemos hecho hasta ahora. Debes probar que el API REST funciona correctamente mediante una herramienta tipo:

https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo

Una vez que hayas comprobado que el API REST funciona correctamente, puedes consumir el listado de jugadores desde angular, tal y como enseña el código de ejemplo. Puedes guardar tu fichero HTML en la carpeta webapp de JHipster.

Demuestra que sabes depurar correctamente la aplicación incluyendo capturas de pantalla del depurador en el navegador (por ejemplo Google Chrome), y también capturas del depurador en el Idea (para depurar el servidor basado en Spring).

8. **Ejercicio opcional:** Realiza una aplicación similar, con la diferencia de que ahora el listado de jugadores se deberá cargar con la información de jugadores de un equipo determinado. Puedes utilizar un select HTML de AngularJS (https://docs.angularjs.org/api/ng/directive/select). La URL del API REST ha de ser la siguiente:

/api/equipos/{id}/jugadores

Importante: utilizar directamente 127.0.0.1 o similar, en lugar de localhost (es posible que os dé problemas en Windows).

9. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_tables_simple

Extiende el ejercicio número 7 para obtener el listado de jugadores del API REST Spring

En este ejercicio, debes mostrar la información de los jugadores en formato tabla, especificando una columna para cada atributo del jugador.

10. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_tables_css

Mejora la tabla del ejercicio anterior utilizando estilos CSS. Te puedes basar en el código de ejemplo proporcionado, y puedes proponer tus propios estilos.

11. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_tables_even

En este ejercicio, debes utilizar la nueva directiva ng-if para aplicar diversos estilos a las columnas en función de que la fila sea par o impar.

12. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_html_show_if

Extiende el ejercicio número 11 de manera que sólo se muestren los jugadores que hayan conseguido 50 o más canastas.

13. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_filters_input

Extiende el ejercicio número 11, de modo que sea posible filtrar los jugadores por cualquiera de sus atributos, tal y como se enseña en la documentación oficial de AngularJS:

https://docs.angularjs.org/api/ng/filter/filter

Consultar el ejemplo al final de la página y hacer clic en “edit in punkle”

14. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_filters_orderby_click

Extiende el ejercicio número 11, de manera que sea posible ordenar por diversos atributos de la clase jugador.

15. Ejercicio basado en el siguiente código de ejemplo:

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_validate_email

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_validate_show

http://www.w3schools.com/angular/tryit.asp?filename=try_ng_validate_classes_form

Desarrolla un formulario para dar de alta un jugador, tal y como se especifica en el código de ejemplo. Para este ejercicio, no es preciso que realmente se comunique con el API REST de Spring para crear el jugador en la base de datos. De manera opcional, para subir nota puedes implementar la creación realista comunicándote con el API REST de Spring investigando el código de JHipster.

En el enlace a Github, en las líneas 75 a 77 se encuentra la solución para deshabilitar CSRF:

https://github.com/alfredorueda/jhipster_public_apis/blob/af4a9d2c74719d994d713f98cf88da72a7d60100/src/main/java/com/mycompany/myapp/config/SecurityConfiguration.java#L76

Solución más completa y realista con la protección CSRF activa:

https://github.com/alfredorueda/jhipster_public_apis/commit/6fc04bde047c280c3077b4c56ffd134c9a691e91

https://github.com/alfredorueda/jhipster_public_apis/blob/master/src/main/webapp/ej15.html

En este caso, es preciso que el ejercicio 15 utilice el mismo nombre de módulo que JHipster, para que se ejecute el código de AngularJS específico para temas de autentificación (y para ello, obviamente, es preciso incluir los scripts que implementan autentificación).

Información sobre CSRF:

https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet

Es un tema de seguridad avanzado y no se va a exigir en el examen.

16. Ejercicio basado en la solución propuesta para el ejercicio 15:

https://github.com/alfredorueda/jhipster_public_apis/blob/master/src/main/webapp/ej15.html

Para comprender esta implementación os explicaré en clase el siguiente servicio de AngularJS para comunicarse con un API REST:

https://docs.angularjs.org/api/ngResource/service/$resource

En este ejercicio, tenéis que realizar el mismo procedimiento para crear un nuevo equipo.

17. Ahora que has sido capaz de crear un nuevo equipo, desarrolla un script con HTML y AngularJS para consultar un jugador, teniendo en cuenta que el jugador ha de estar vinculado a un equipo. Es decir, debe de existir la asociación entre el jugador y el equipo. Una vez que se muestra la información sobre el jugador al usuario, ha de ser posible modificar los datos del jugador incluido el equipo al que pertenece. El formulario debe de incluir un botón para actualizar los datos del jugador. Te puedes basar en el ejercicio número 16, y también en el código de JHipster para actualizar una entidad:

https://github.com/alfredorueda/jhipster_public_apis/blob/af4a9d2c74719d994d713f98cf88da72a7d60100/src/main/webapp/scripts/app/entities/jugador/jugador-dialog.html

https://github.com/alfredorueda/jhipster_public_apis/blob/af4a9d2c74719d994d713f98cf88da72a7d60100/src/main/webapp/scripts/app/entities/jugador/jugador-dialog.controller.js

https://github.com/alfredorueda/jhipster_public_apis/blob/af4a9d2c74719d994d713f98cf88da72a7d60100/src/main/webapp/scripts/components/entities/jugador/jugador.service.js

Nota: Utiliza los ficheros equivalentes de un proyecto AngularJS y Spring, generado con JHipster en el que exista la asociación entre jugador y equipo.

Ya estás muy cerca de ser capaz de comprender una capa cliente AngularJS realista a nivel empresarial como, por ejemplo, la que genera JHipster.

El siguiente paso es entender el sistema de rutas para navegar por la interfaz de usuario:

https://github.com/angular-ui/ui-router/wiki

Anotaciones para el profesor para los próximos ejercicios:

- Documentación oficial de AngularJS para explicar la comunicación avanzada con el API REST de Spring tal y como lo lleva a cabo JHipster:

https://docs.angularjs.org/api/ngResource/service/$resource

Este servicio está basado en un servicio de más bajo nivel:

https://docs.angularjs.org/api/ng/service/$http

Tiene un ejemplo muy bueno que se puede experimentar online.

- Documentación para entender el sistema avanzado de rutas en AngularJS que utiliza JHipster :

https://github.com/angular-ui/ui-router/wiki

http://www.webcodegeeks.com/javascript/angular-js/angularjs-ui-router-example/

https://jsfiddle.net/r9mbtp3c/

http://joelhooks.com/blog/2013/07/22/the-basics-of-using-ui-router-with-angularjs/

https://scotch.io/tutorials/angular-routing-using-ui-router

http://www.sitepoint.com/creating-stateful-modals-angularjs-angular-ui-router/