SPARK JAVA – PARTE I
REST

Objetivos

Spark es un framework Java (utiliza la versión 8) sencillo y muy ligero que permite desarrollar aplicaciones web. El objetivo de este trabajo práctico es utilizar dicho framework para construir microservicios REST (microservicios) de mínima complejidad bajo el concepto arquitectónico Modelo-Vista-Controlador (MVC del inglés), utilizando los conocimientos básicos del lenguaje JAVA que ya posee.
Las actividades correspondientes al taller de Spark, se dividen en tres secciones. Esta sección I tiene con objetivo particular construir los modelos e implementar microservicios REST básicos de forma que pueda ser subida a la nube y ser invocado desde cualquier lugar, con cualquier plataforma y cualquier lenguaje de programación.

Herramientas / Material de Consulta

Para llevar a cabo los ejercicios, se debe:

- Tener instalado Java Development Kit 8 (JDK 8). Se puede instalar en Windows, Mac o Linux y descargar de la página oficial de Oracle
- Tener instalado un entorno de desarrollo (IDE) para Java, tal como Eclipse, IntelliJ o NetBeans. La cátedra sugiere la versión 8.0.2 de NetBeans para Java EE.

Material de consulta:

- Documentación online: http://sparkjava.com/documentation
- Transparencias.
- Paul Clements_ et al - Documenting software architectures: views and beyond-Addison-Wesley (2011). Leer el Anexo A UML - pg 431 para construir los modelos solicitados.

Entrega de Resolución

- El codigo fuente debe ser entregado zipeado por ejercicio, donde el nombre del archivo debe comenzar con el apellido y nombre del alumno y continuar con el práctico que resuelve <apellido-nombre>Practico1. Cada ejercicio debe estar en un directorio distinto con el numero de ejercicio, nombre del directorio ej<nro>
- Entregar las imagenes de los modelos. No mandar el fuente de la herramienta que hayan utilizado. Crear un pdf con todos los ejercicios y dentro de cada ejercicio todos los modelos, pero que sea un SOLO PDF. Nombre del archivo <apellido-nombre>ModelosPractico1

Ejercicios

Crear servicios REST que resuelvan cada consigna. A partir del ejercicio 3 los resultados que sean devueltos en formato JSON.

Ejercicio 1: Dada una palabra ingresada por el usuario, devolver si es palíndromo o no.

Ejercicio 2: Dado el radio de una circunferencia ingresados por el usuario, calcular su perímetro y área.

Ejercicio 3: Registrar en memoria un conjunto de personas, sus hijos y los nietos (utilice listas parametrizadas). Obtener las personas registradas y los hijos y los nietos de alguna de esas personas.

Diseñar el modelo arquitectónico en UML construyendo el modelo de componentes y conectores, y el modelo dinámico entre componentes (diagrama de secuencia).

Ejercicio 4: Devolver los teléfonos para emergencias, según el dato solicitado por el usuario. Por ejemplo, si solicita “policía”, que devuelva “911”. Lo mismo con los bomberos, violencia de género y otros, cargados en memoria durante la ejecución del programa.

Ejercicio 5: Dado un monto en dinero ingresados por el usuario, devolver el valor que tendrá el mismo luego de un plazo indicado por el usuario (ej.: 1 año, 5 meses, etc), de acuerdo con la inflación propuesta por el gobierno Argentino.

Diseñar el modelo arquitectónico en UML construyendo el modelo de componentes y conectores, y el modelo dinámico entre módulos (diagrama de secuencia).

Ejercicio 6: Dado un artículo por código solicitado por el usuario, retornar la descripción, el precio neto, el IVA y el total (neto+IVA) del mismo.

Ejercicio 7: Dado un autor ingresado por el usuario, devolver los libros escritos por él.

Ejercicio 8: Crear un servicio REST que devuelva los resultados de un torneo de fútbol local en que participan 8 equipos. Servicios a desarrollar:
- todos los partidos, jugados y por jugar.

- partidos de fútbol jugados por un equipo dado con el resultado obtenido. Por ejemplo, al consultar sobre “Boca”, se deberá devolver:

[{"local":"River","visitante":"Boca","goleslocal":"2", "golesvisitante":"2","jugado":true}]

- equipos con los que jugó de local. Por ejemplo River:

[{"local":"River","visitante":"Boca","goleslocal":"2", "golesvisitante":"2","jugado":true},
{"local":"River","visitante":"Racing","goleslocal":"0", "golesvisitante":"0","jugado":true}]

- equipos con los que debe jugar de visitante. Por ejemplo Racing:

[{"local":"Aldocivi","visitante":"Racing","goleslocal":"", "golesvisitante":"","jugado":false},
{"local":"Talleres","visitante":"Racing","goleslocal":"", "golesvisitante":"","jugado":false}]

- cantidad de partidos ganados por un equipo
- cantidad de goles convertidos por un equipo

Diseñar el modelo arquitectónico en UML construyendo el modelo de componentes y conectores, y el modelo dinámico entre módulos (diagrama de secuencia).

OPCIONALES

Ejercicio O1: Dada una palabra ingresada por el usuario, devolver su longitud.

Ejercicio O2: Dados dos números ingresados por el usuario, realizar la suma y resta de ambos.

Ejercicio O3: Dada una cantidad de segundos ingresados por el usuario, convertirlos en hora, minutos y segundos.

Ejercicio O4: Un servicio REST que indique dada una palabra cuantas letras, números y símbolos contiene..