Guía para Implementar Servicios REST

1. **Descargar dependencias y herramientas**

- Red Hat Developer Studio: También es posible descargar este stack de desarrollo desde el Marketplace disponible en Eclipse
- Wildfly Application Server: Tras la descarga, realizar la descompresión del mismo en una ruta del sistema.

2. **Descargar esqueleto ejemplo (VideoAndes)**

- Descargar el esqueleto de Ejemplo desde SicuaPlus en la carpeta Contenido/Proyecto/EjemplosYEsqueletos

3. **Descargar Postman**

- Esta aplicación se encuentra disponible como un plugin del navegador Google Chrome/Chromium: https://www.getpostman.com/

4. **Importar el proyecto en Eclipse/Red Hat Developer Studio**
5. **Resolver conflictos relacionados con el Java Build Path y la versión del JRE**: Bajo la ventana de configuración de proyecto, digirse a la opción que permite configurar el Java Build Path, a continuación, editar el `JRE System Library` y seleccionar la versión del Java Runtime Environment instalado en su entorno local.
6. **Enlazar y añadir las credenciales de acceso y conexión a la Base de Datos**: Bajo la carpeta `WebContent/WEB-INF/ConnectionData`, es posible encontrar un archivo denominado `conexion.properties`, por favor, editar el mismo, y en cada una de las entradas que se encuentran *i.e.,* `usuario, clave`, reemplazar por las credenciales de acceso que fueron otorgadas al inicio del semestre.
7. **Definición del servidor, enlace y despliegue de la aplicación:** Tras resolver los conflictos de dependencia relacionados con el JRE, es necesario ejecutar la aplicación y configurar el Servidor de aplicaciones JBoss Wildfly 10, para este fin, al ejecutar el proyecto, es necesario seleccionar la opción `Run on Server`. Tras seleccionar esta opción, es posible seleccionar un nuevo servidor a crear, por favor, seleccionar entre las opciones, `Wildfly 10.x` (Esta opción se encuentra normalmente agrupada bajo la categoría JBoss Community). Seleccionar la versión más reciente que reporte el sistema, a continuación, omitir la siguiente ventana. Posteriormente, cuando aparezca un diálogo que presente un campo de texto llamado: “Home Directory”, proceder a explorar y seleccionar la carpeta que contiene el Servidor de Aplicaciones, definida en la instrucción 1.
8. **Documentación con respecto a la definición y uso de REST** (Opcional): Revisar la definición, ejemplos y la sección *Relationship between URL and HTTP methods* asociados a la entrada en Wikipedia referente a REST
9. **Documentación y tutorial básico relacionado con la implementación de servicios REST en Java** (Opcional): Documento disponible bajo la ruta `Docs/Rest_HelloWorld_Tutorial.pdf`

10. **Explorar la estructura de carpetas y paquetes del archivo, así como el flujo de información**: Tras familiarizarse con el esquema de implementación de servicios REST, así como la distribución y la invocación de los mismos, a través de verbos HTTP, revisar la implementación de un servicio en REST en específico, el intercambio de información a través de Value Objects (VOS) a través del TransactionManager `VideoAndesMaster.java`, así como la implementación y resolución de cada una de las consultas descritas en SQL en cada uno de los DAO (Data Access Object) asociados a cada una de las tablas del esquema de la aplicación. Es necesario notar que el patrón de implementación de un nuevo servicio permanece invariante en el tiempo (Siempre es el mismo)
11. **Realizar pruebas de los servicios REST a través de Postman**: Tras deducir las rutas y servicios que ofrece la aplicación, es posible abrir Postman y realizar solicitudes al servidor (VideoAndes). Es necesario recordar que el servidor Wildfly se encuentra disponible bajo la ruta `localhost:8080`, la aplicación se encuentra disponible bajo la ruta `/VideoAndes`, mientras que los servicios REST se encuentran disponibles bajo la ruta `/rest`. En el presente ejemplo, se desea recuperar todos los registros disponibles en la Video Tienda, como es posible apreciar en la clase `VideoAndesVideosServices.java`, (Disponible bajo el paquete `rest`), ésta implementa y expone los servicios relacionados con la entidad `Video`, bajo la ruta `/videos`, esto implica que la solicitud se encontraría definida bajo la siguiente morfología:

    ```
    GET localhost:8080/VideoAndes/rest/videos
    ```

    Con el fin de enviar información o nuevos objetos a la aplicación *i.e.,* `POST, PUT`, es necesario definir el cuerpo del objeto que el servidor espera al momento de resolver la solicitud respectiva. En Postman, es posible definir un objeto de envío JSON bajo la pestaña `Body`, opción `raw`. Es necesario que el `Content-Type` asociado a la solicitud corresponda a `Application/JSON`, de lo contario, la solicitud sería rechazada por parte del servidor en la medida que éste consume y produce objetos en este formato. Es necesario que los atributos del objeto a enviar, correspondan a aquellos atributos del objeto VOS que el servidor espera, *e.g.,* Si se desea añadir un nuevo video al catálogo de VideoAndes `POST localhost:8080/VideoAndes/rest/videos`, el objeto debe cumplir con la definición del Objeto `vos.Video`, como es posible apreciar a continuación:

    ```
    {
      "id":14,
      "name":"A Clockwork Orange",
      "duration":136
    }
    ```

12. **Modificar el nombre de la aplicación y la ruta de definición para los servicios REST**: Bajo el directorio `WebContent/WEB-INF`, es posible hallar el archivo `web.xml`, el cual define directivas, así como definiciones de direcciones de consumo e identificadores uniformes

de acceso a los recursos web. En el presente contexto, el parámetro `display-name`, permite redefinir el nombre de la aplicación a desplegar en el servidor.

Notas finales con respecto a la arquitectura de la aplicación

Es necesario notar que el Transaction Manager es quién establece y gestiona la conexión a la conexión de la Base de Datos, así mismo, ésta capa se encuentra encargada de iniciar, ejecutar y finalizar una transacción, sujeto a las condiciones ACID. De igual forma, cada DAO debe encontrarse asociado a una tabla o combinación de tablas específica (En el contexto de consultas complejas), cada DAO debe definir las sentencias SQL a ejecutar sobre la base de datos, así como la conversión de las tuplas a los objetos de descripción. Esto implica que la Capa de Servicios, así como el Transaction Manager, no debe definir ninguna consulta SQL, asímismo, la capa de servicios y cada DAO no se encuentra en responsabilidad de gestionar la base de datos. Finalmente, el Transaction Manager o un DAO, no deberá recibir o responder solicitudes enviadas al servidor a través de HTTP.

Recomendaciones y sugerencias

- Durante la implementación del proyecto, se sugiere guardar un archivo compatible con Postman que contenga los casos de prueba asociados a cada servicio implementado en cada iteración, esto resulta necesario con el fin de acelerar el proceso de calificación durante las sustentaciones, así como para evitar posibles malentendidos.
- Es esencial respetar la arquitectura de la aplicación, conforme a la descripción otorgada previamente.

Espero que ésta guía, así como las recomendaciones resulten útiles y permitan resolver sus dudas.