PROGRAMACION ORIENTADA A OBJETOS II
PROGRAMA 2

Para este programa, deberá completar los servicios REST que se comenzaron a desarrollar para la práctica 9, los cuales deben poder ser usados por alguna implementación de la interface DaoEscolares del Programa 1, es decir, los servicios REST a generar, deben proveer la información requerida por los métodos indicados en la interface DAOEscolares (no es necesario todos los métodos de los DAOs individuales del Programa 1).

Entonces, tomando en cuenta de la práctica que el URL Base de los servicios a proporcionar es http://localhost:8080/RESTControlEscolar-XXXXXXXX/servicios (donde XXXXXXXX se sustituye por su matrícula, el programa 2 deberá contener un servicio REST por cada una de las tablas (estado, municipio, estudiante, profesor, cargaestudiante, cargaprofesor, periodoescolar, carrera y materia) que componen a la base de datos controlescolar, cada uno representado por la ruta indicada más adelante.

INSTRUCCIONES PARA EL PROYECTO DEL PROGRAMA 2 (Y PRÁCTICA 9)

Paso 1. Clone el repositorio usando la técnica que le convenga (ya sea usando Intellij IDEA o desde la terminal de comandos usando **git clone**). Los detalles están en el archivo README.md

Paso 2. Con la guía del instructor, se estará demostrando lo que hay que realizar para el Programa 2, completando específicamente lo relacionado a los servicios REST para las tablas municipio y estudiante (la calificación de la Práctica 9 es en base a los servicios de estas dos tablas).

Paso 3. Una vez completado lo relacionado a la Práctica 9, podrá continuar de manera independiente lo restante. El mismo proyecto se seguirá usando para completar el código necesario para manejar el resto de las tablas.

NOTA IMPORTANTE: Por el momento las únicas pruebas completas son las que tienen que ver con lo que se va a completar para la práctica 9 (están dentro de la sección test en el paquete poo2.prac09.main). Para lo que completarán para el Programa 2, en los siguientes días se les estarán proporcionando los archivos de pruebas. Cada vez que haya nuevas clases de prueba deberás:

a) Agregarlas a la sección test en el paquete poo2.progs.main
b) Incluirlas haciendo un commit
c) Antes de que hagan un push (al completar la implementación de algún método y después de hacer el commit respectivo), **deben** hacer un pull antes de hacer el push (por si el repositorio remoto tiene incluidas nuevas pruebas), de otra manera les marcará error al hacer el push

MODIFICACIÓN DE LOS ARCHIVOS PROPORCIONADOS (EN CUANTO A CONFIGURACIÓN)

Una vez que tengas el repositorio local, el trabajo principal consiste en agregar ciertas anotaciones a las clases de entidad para que la API de Java Empresarial pueda acceder a las tablas (y hacer ciertas validaciones) y crear las clases para proporcionar los servicios REST descritos en la siguiente sección. Asegúrate de una vez creados los servicios REST, poner en el método agregaRecursosREST de la clase AplicacionConfig código para agregar las clases que representan a los servicios REST al Set que recibe como argumento.

Antes de hacer los cambios en el código haga los pasos indicados en la sección CONFIGURACION INICIAL del archivo README.md

TRABAJO A REALIZAR EN CUANTO A CÓDIGO

De acuerdo a lo indicado para el Programa 1, la interface ***DaoEscolares*** contiene los siguientes métodos:

1. **public List<Carrera> obtenCarreras()**
2. **public boolean agregaCarrera(Carrera c)**
3. **public boolean eliminaCarrera(String clave)**
4. **public boolean modificaCarrera(Carrera c)**

5. **public List<Entidad> obtenEstados()**

6. **public List<Municipio> obtenMunicipios(long idEstado)**

7. **public List<Materia> obtenMaterias()**
8. **public boolean agregaMateria(Materia m)**
9. **public boolean eliminaMateria(String clave)**
10. **public boolean modificaMateria(Materia m)**

Los métodos en verde son métodos que no corresponden a ninguno de los 5 métodos que ya pueden realizar los servicios REST por heredar de RESTAbstracto

11. **public List<Periodoescolar> obtenPeriodos()**
12. **public boolean agregaPeriodo(Periodoescolar p)**
13. **public boolean eliminaPeriodo(long id)**
14. **public boolean modificaPeriodo(Periodoescolar p)**

15. **public List<Profesor> obtenProfesores()**
16. **public boolean agregaProfesor(Profesor p)**
17. **public boolean eliminaProfesor(String rfc)**
18. **public boolean modificaProfesor(Profesor p)**

19. **public List<DetalleCargaProfesor> obtenCargaProfesor(String rfc)**
20. **public boolean agregaCargaProfesor(Cargaprofesor c)**
21. **public boolean eliminaCargaProfesor(long idCarga)**

22. **public List<Estudiante> obtenEstudiantes()**
23. **public boolean agregaEstudiante(Estudiante e)**
24. **public boolean eliminaEstudiante(String matricula)**
25. **public boolean modificaEstudiante(Estudiante e)**

26. **public List<DetalleCargaEstudiante> obtenCargaEstudiante(String matricula)**
27. **public boolean agregaCargaEstudiante(Cargaestudiante c)**
28. **public boolean eliminaCargaEstudiante(long idCarga)**

Para todos estos métodos debe crear servicios REST que realicen el trabajo correspondiente.

Los métodos 1 a 4 deberían estar en un servicio REST de ruta **carrera**. El método 5 debería estar en un servicio REST de ruta **estado**. El método 6 debería estar en un servicio REST de ruta **municipio**. Los métodos 7 a 10 deberían estar en un servicio REST de ruta **materia**. Los métodos 11 a 14 deberían estar en un servicio REST de ruta **periodoescolar**. Los métodos 15 a 18 deberían estar en un servicio REST de ruta **profesor**. Los métodos 19 a 21 deberían estar en un servicio REST de ruta **cargaprofesor**. Los métodos 22 a 25 deberían estar en un servicio REST de ruta **estudiante**. Los métodos 26 a 28 deberían estar en un servicio REST de ruta **cargaestudiante**.

Las clases de entidad a generar deberán estar en el paquete poo2.progs.entidades y los servicios en el paquete poo2.progs.servicios.

Algunos de estos métodos ya tienen prácticamente la funcionalidad deseada, puesto que los servicios REST heredan de la clase RESTAbstracto, mientras que otros requieren de la generación de código más elaborado para realizar el trabajo que se requiere. La ruta específica que debe tener el servicio REST asociado con los métodos de la interface DAOEscolares son los siguientes (urlbase es el URL Base de los servicios REST, indicado ya previamente):

Método de la Interface DaoEscolares	**Ruta del Servicio REST**	**Método HTTP a usar**
obtenCarreras	urlbase/carrera	GET
agregaCarrera	urlbase/carrera	POST
modificaCarrera	urlbase/carrera/{clave}	PUT
eliminaCarrera	urlbase/carrera/{clave}	DELETE
obtenEstados	urlbase/estado	GET
obtenMunicipios	urlbase/municipio/{idestado}	GET
obtenMaterias	urlbase/materia	GET
agregaMateria	urlbase/materia	POST
modificaMateria	urlbase/materia/{clave}	PUT
eliminaMateria	urlbase/materia/{clave}	DELETE
obtenPeriodos	urlbase/periodo	GET
agregaPeriodo	urlbase/periodo	POST
modificaPeriodo	urlbase/periodo/{id}	PUT
eliminaPeriodo	urlbase/periodo/{id}	DELETE
obtenProfesores	urlbase/profesor	GET
agregaProfesor	urlbase/profesor	POST
modificaProfesor	urlbase/profesor/{rfc}	PUT
eliminaProfesor	urlbase/profesor/{rfc}	DELETE
obtenCargaProfesor	urlbase/cargaprofesor/{rfc}	GET
agregaCargaProfesor	urlbase/cargaprofesor	POST
eliminaCargaProfesor	urlbase/cargaprofesor/{idcarga}	DELETE

obtenEstudiantes	urlbase/estudiante	GET
agregaEstudiante	urlbase/estudiante	POST
modificaEstudiante	urlbase/estudiante/{matricula}	PUT
eliminaEstudiante	urlbase/estudiante/{matricula}	DELETE
obtenCargaEstudiante	urlbase/cargaestudiante/{matricula}	GET
agregaCargaEstudiante	urlbase/cargaestudiante	POST
eliminaCargaEstudiante	urlbase/cargaestudiante/{idcarga}	DELETE

Para que los servicios REST funcionen correctamente, es necesario que las clases de entidad correspondiente tengan las anotaciones necesarias, durante la práctica 9 se estará agregando las anotaciones a 3 de ellas (Estudiante, Municipio y Cargaestudiante), las cuales pueden usar de referencia para completar las restantes. Para las clases extendidas (DetalleCargaProfesor y DetalleCargaEstudiante solo es necesario poner la anotación @XmlRootElement a la clase (dado que esas no representan a una tabla de la base de datos).

Al igual que para los servicios realizados en la práctica 9, las operaciones para agregar, modificar o eliminar deberán primero validar si es posible realizar la operación solicitada, de acuerdo con las reglas indicadas en el Programa 1, si no es posible, regresará "false".

ESPECIFICACIONES IMPORTANTES QUE DEBEN CUMPLIR LAS CLASES DE ENTIDAD

1. Las clases de entidad deben tener las siguientes anotaciones:
    a. `@Entity` que indica que esta clase representa una tabla en alguna base de datos
    b. `@Table(name="nombretabla")` que indica explícitamente el nombre de la tabla que representa la clase
    c. `@XmlRootElement` que representa que la información de esta clase puede ser convertida a un formato XML (uno de los dos comúnmente usados en la transferencia de información entre cliente y servicio REST)
2. Los atributos de la clases de entidad deberán tener anotaciones que especifican información asociada con la columna correspondiente en la tabla:
    a. `@Column(name = "nombrecolumna")` para indicar a que columna está relacionado el atributo
    b. @Id si es que el atributo está relacionado a la llave primaria de la tabla
    c. `@Basic(optional = false)` si es que el campo en la tabla es obligatorio
    d. `@NotNull` en caso de que la columna correspondiente tenga el atributo NOT NULL y sea representado en Java con una clase (no con un tipo primitivo)
    e. Se pueden usar anotaciones para solicitar que JPA haga ciertas validaciones, por ejemplo, una longitud mínimo y/o una máxima usando `@Size(min = LONGMIN, max = LONGMAX)`, no es necesario poner min y max, se ponen solo los deseados
    f. Se puede especificar que cumpla con alguna expresión regular usando @Pattern(regexp =EXPRESIONREGULAR, message=MENSAJEERROR), donde EXPRESIONREGULAR es la expresión regular que se requiere cumpla el valor a guardar y message contiene el mensaje a regresar en la excepción que se genera si el campo no cumple con la expresión regular

ESPECIFICACIONES IMPORTANTES QUE DEBEN CUMPLIR LOS SERVICIOS REST

1. Las clases que representan a los servicios REST deben tener un constructor vacío que manda llamar al constructor de la clase padre pasándole como referencia la clase de entidad con la que se trabajará.

2. Deben tener un atributo privado de tipo **EntityManager** que tendrá la anotación siguiente: **`@PersistenceContext(unitName = "default")`**
3. El método **getEntityManager** (que debe implementar) regresará simplemente el objeto **EntityManager** que tiene como atributo
4. Solo implementará los métodos que requiera hacer para la tabla para la cual está haciendo los servicios REST con las anotaciones adecuadas (GET, POST, PUT, DELETE, si consume información de que tipo, si genera información de que tipo y las rutas asociadas)

CON RESPECTO A LAS PRUEBAS EN GITHUB

Para este programa y práctica, las pruebas que se realizarán en Github requerirán que el servidor Glassfish este corriendo en su computadora y el servidor MySQL que se estará usando será el de su computadora. Para que estas pruebas puedan por tanto funcionar, se requiere de instalar un software que permita a Github comunicarse con su servidor Glassfish y servidor MySQL de manera remota. El software que usaremos será ngrok y en las sesiones de la semana del 22 al 26 de marzo se demostrará como configurar ngrok y como ejecutarlo antes de hacer un push al repositorio remoto (que tenga como fin ver cómo funcionan las pruebas en Github). Esto implicará estar modificando un archivo de configuración (datosmysql.properties) antes de hacer cada push donde ya se quiera estar haciendo pruebas en el repositorio remoto. Mas detalles en el archivo README.md

PORCENTAJE DE CALIFICACION DE CADA ELEMENTO PARA PRACTICA 9:

La calificación de cada servicio es el siguiente:

Método de la Interface DAOEscolares	Ruta del Servicio REST	Puntos
obtenMunicipios	urlbase/municipio/{idestado}	20
obtenEstudiantes	urlbase/estudiante	5
agregaEstudiante	urlbase/estudiante	25
modificaEstudiante	urlbase/estudiante/{matricula}	25
eliminaEstudiante	urlbase/estudiante/{matricula}	25

TOTAL DE PUNTOS: 100

PORCENTAJE DE CALIFICACION DE CADA ELEMENTO PARA PROGRAMA 2:

La calificación de cada servicio es el siguiente:

Método de la Interface DAOEscolares	Ruta del Servicio REST	Puntos
obtenCarreras	urlbase/carrera	1
agregaCarrera	urlbase/carrera	5
modificaCarrera	urlbase/carrera/{clave}	5
eliminaCarrera	urlbase/carrera/{clave}	5
obtenEstados	urlbase/estado	1
obtenMunicipios	urlbase/municipio/{idestado}	1
obtenMaterias	urlbase/materia	1
agregaMateria	urlbase/materia	5
modificaMateria	urlbase/materia/{clave}	5

eliminaMateria	urlbase/materia/{clave}	5
obtenPeriodos	urlbase/periodo	1
agregaPeriodo	urlbase/periodo	3.25
modificaPeriodo	urlbase/periodo/{id}	3.25
eliminaPeriodo	urlbase/periodo/{id}	5
obtenProfesores	urlbase/profesor	1
agregaProfesor	urlbase/profesor	5
modificaProfesor	urlbase/profesor/{rfc}	5
eliminaProfesor	urlbase/profesor/{rfc}	5
obtenCargaProfesor	urlbase/cargaprofesor/{rfc}	10
agregaCargaProfesor	urlbase/cargaprofesor	6
eliminaCargaProfesor	urlbase/cargaprofesor/{idcarga}	1
obtenEstudiantes	urlbase/estudiante	0.5
agregaEstudiante	urlbase/estudiante	1
modificaEstudiante	urlbase/estudiante/{matricula}	1
eliminaEstudiante	urlbase/estudiante/{matricula}	1
obtenCargaEstudiante	urlbase/cargaestudiante/{matricula}	10
agregaCargaEstudiante	urlbase/cargaestudiante	6
eliminaCargaEstudiante	urlbase/cargaestudiante/{idcarga}	1

TOTAL DE PUNTOS: 100

PUNTOS EXTRA PARA PROGRAMA 2: 10%

- Si durante el desarrollo del programa 2 realizan commits atómicos (es decir, que representan el cambio de una unidad de código, en este caso un método) que describan de manera adecuada el cambio realizado obtendrán 10 puntos extra

RECUERDEN QUE LOS TRABAJOS DE PROGRAMACION NO SON REEMPLAZABLES LO CUAL SIGNIFICA QUE SI NO SON ENTREGADOS NO HABRA FORMA DE RECUPERAR EL PORCENTAJE DE LA CALIFICACION FINAL QUE LES CORRESPONDE.

La fecha y hora límite para hacer el push final es a las 23:59 hrs del 18 de abril del 2021. NO SE ACEPTARÁN PROGRAMAS ENVIADOS POR CORREO ELECTRÓNICO. El nombre que utilice para las clases, atributos y métodos deben seguir las convenciones mencionadas en clase, si usa identificadores tales como xxx, xya, o similares el programa se le asignará una calificación reprobatoria. Finalmente, si se reciben programas copiados, todas las materias que así lo entreguen recibirán calificación reprobatoria **EN LA MATERIA, NO SOLAMENTE EN EL PROGRAMA**.