Imagen en blanco y negro

Descripción generada automáticamente con confianza baja

Laboratorio 7

Persistencia

Integrantes:

Luisa Fernanda Bermúdez Girón

Karol Daniela Ladino Ladino

Profesor:

Iván Darío Viasus Quintero

Curso:

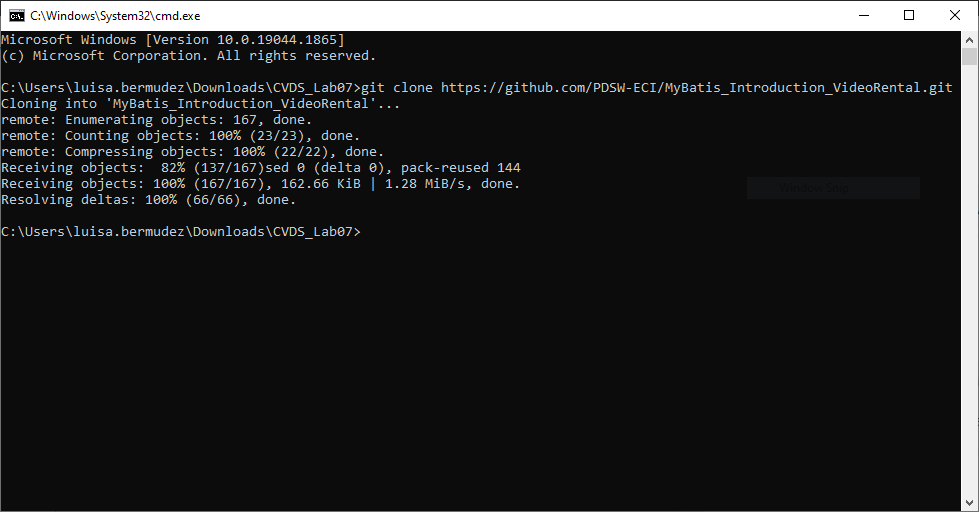
CVDS - 2

Fecha De Entrega:

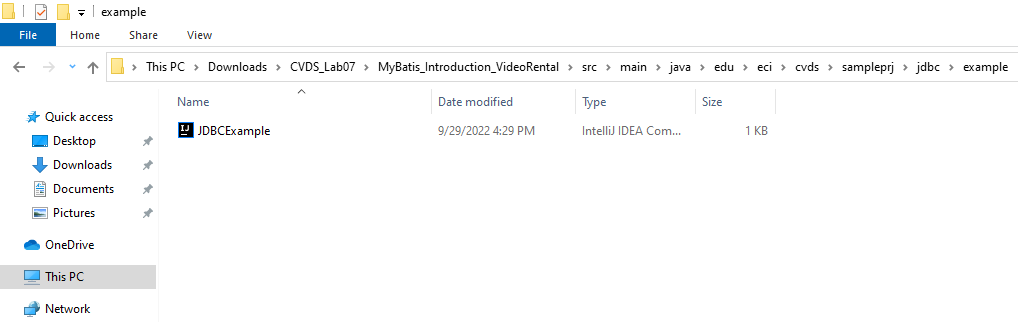
06-10-2022

**SECCIÓN I. - INTRODUCCIÓN A JDBC**

1. Clonar el proyecto MyBatis\_Introduction\_VideoRental de GitHub donde se realizará la implementación completa del laboratorio.



1. Descargue el archivo JDBCExample.java y agréguelo en el paquete "edu.eci.cvds.sampleprj.jdbc.example".



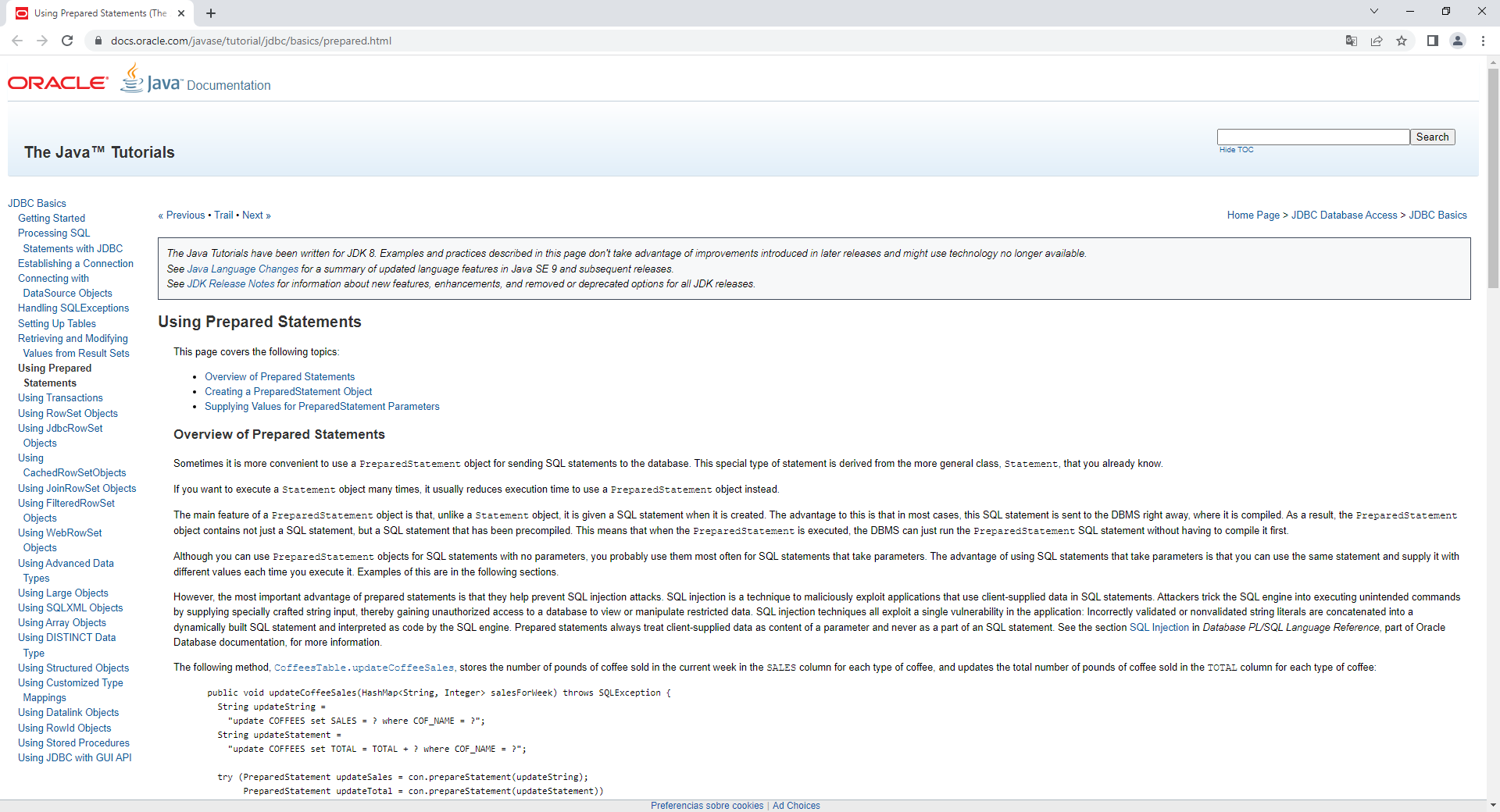
1. Desde esta clase se realizará una conexión a una base de datos MySQL por medio de JDBC y sus "Prepared Statements".

Se reviso la clase que se puso en el proyecto.

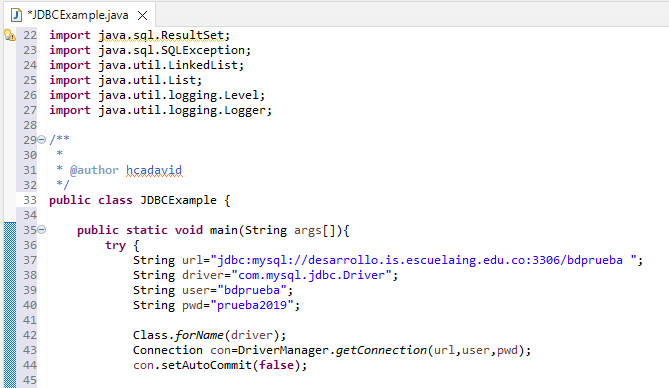
1. En un motor de base de datos SQL se tiene un esquema con el siguiente modelo de base de datos (para registrar pedidos sobre productos):



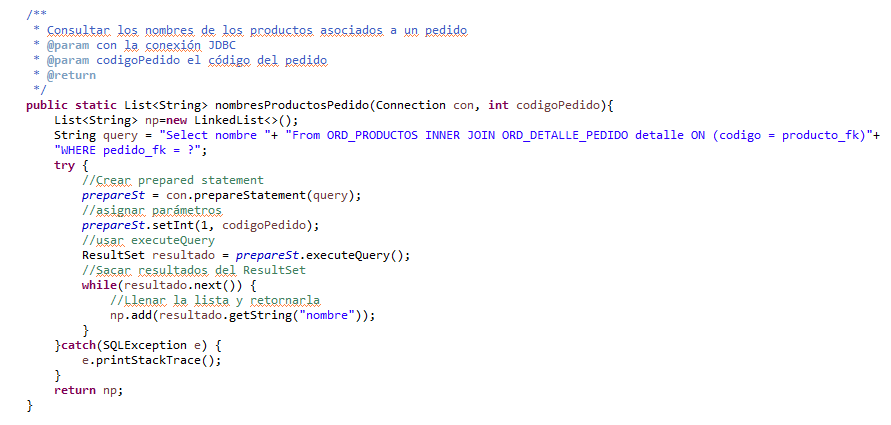
1. Revise la documentación de PreparedStatement del API JDBC



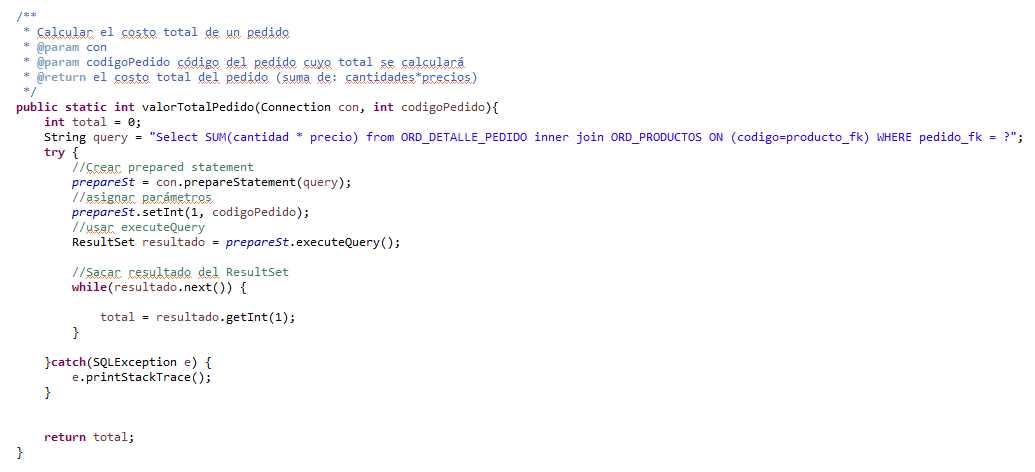
1. En la clase JDBCExample juste los parámetros de conexión a la base de datos con los datos reales:



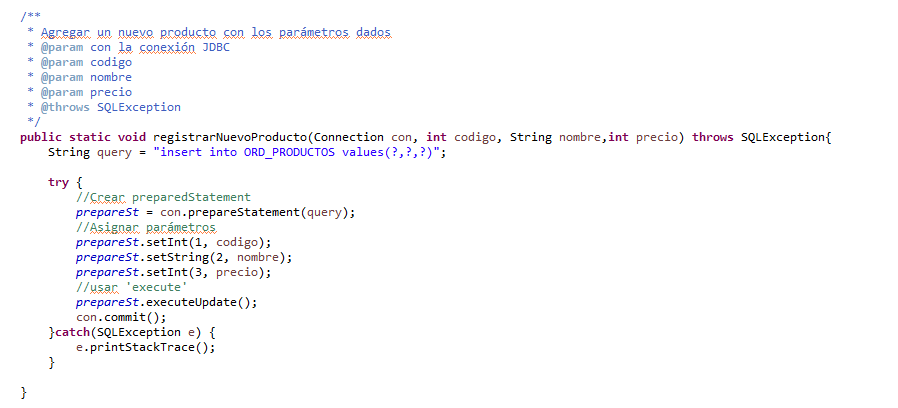
1. Implemente las operaciones faltantes:
2. nombresProductosPedido



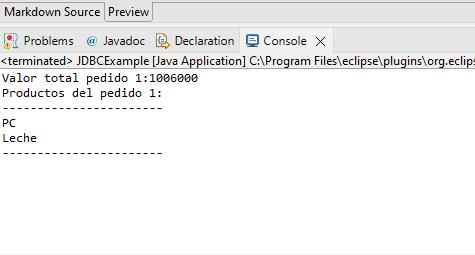
1. valorTotalPedido - El resultado final lo debe retornar la base de datos, no se deben hacer operaciones en memoria.



1. registrarNuevoProducto - Use su código de estudiante para evitar colisiones.



1. Verifique por medio de un cliente SQL, que la información retornada por el programa coincide con la que se encuentra almacenada en base de datos.

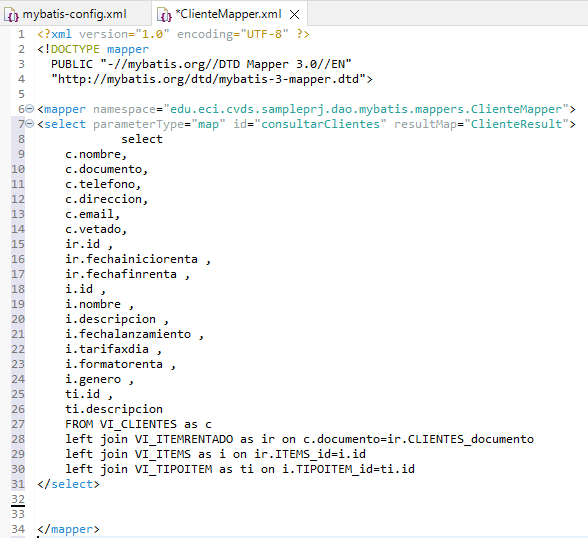


**Parte I (Para entregar en clase)**

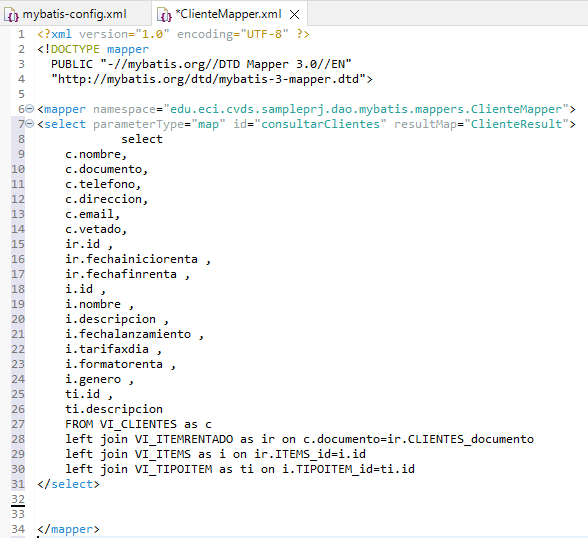
1. Ubique los archivos de configuración para producción de MyBATIS (mybatis-config.xml). Éste está en la ruta donde normalmente se ubican los archivos de configuración de aplicaciones montadas en Maven (src/main/resources). Edítelos y agregue en éste, después de la sección <settings> los siguientes 'typeAliases':



1. Lo primero que va a hacer es configurar un 'mapper' que permita que el framework reconstruya todos los objetos Cliente con sus detalles (ItemsRentados). Para hacer más eficiente la reconstrucción, la misma se realizará a partir de una sola sentencia SQL que relaciona los Clientes, sus Items Rentados, Los Items asociados a la renta, y el tipo de item. Ejecute esta sentencia en un cliente SQL (en las estaciones Linux está instalado EMMA), y revise qué nombre se le está asignando a cada columna del resultado:



1. Abra el archivo XML en el cual se definirán los parámetros para que MyBatis genere el 'mapper' de Cliente (ClienteMapper.xml). Ahora, mapee un elemento de tipo <select> al método 'consultarClientes':

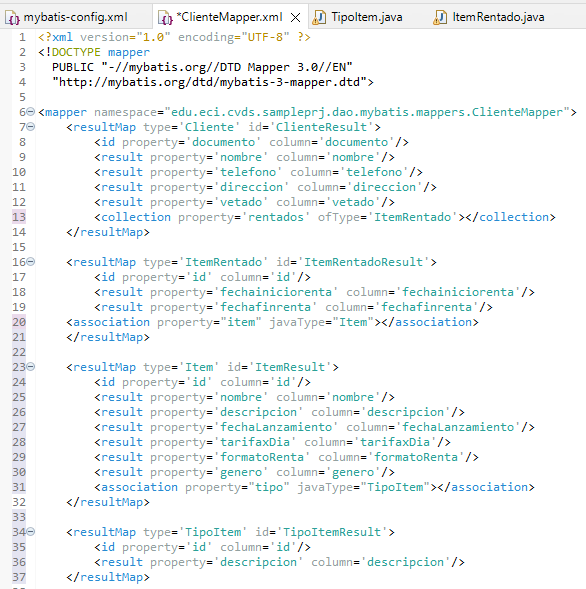


1. Note que el mapeo hecho anteriormente, se indica que los detalles de a qué atributo corresponde cada columna del resultado de la consulta están en un 'resultMap' llamado "ClienteResult". En el XML del mapeo agregue un elemento de tipo <resultMap>, en el cual se defina, para una entidad(clase) en particular, a qué columnas estarán asociadas cada una de sus propiedades (recuerde que propiedad != atributo). La siguiente es un ejemplo del uso de la sintaxis de <resultMap> para la clase Maestro, la cual tiene una relación 'uno a muchos' con la clase DetalleUno y una relación 'uno a uno' con la clase DetalleDos, y donde -a la vez-, DetalleUno tiene una relación 'uno-a-uno- con DetalleDos:



Como observa, Para cada propiedad de la clase se agregará un elemento de tipo <result>, el cual, en la propiedad 'property' indicará el nombre de la propiedad, y en la columna 'column' indicará el nombre de la columna de su tabla correspondiente (en la que se hará persistente). En caso de que la columna sea una llave primaria, en lugar de 'result' se usará un elemento de tipo 'id'. Cuando la clase tiene una relación de composición con otra, se agrega un elemento de tipo <association>.Finalmente, observe que si la clase tiene un atributo de tipo colección (List, Set, etc), se agregará un elemento de tipo <collection>, indicando (en la propiedad 'ofType') de qué tipo son los elementos de la colección. En cuanto al indentificador del 'resultMap', como convención se suele utilizar el nombre del tipo de dato concatenado con 'Result' como sufijo.

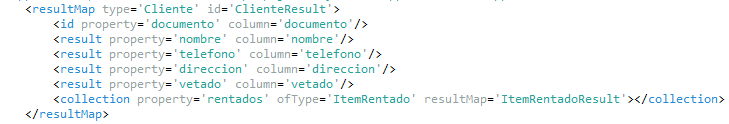
Teniendo en cuenta lo anterior, haga cuatro 'resultMap': uno para la clase Cliente, otro para la clase ItemRentado, otro para la clase Item, y otro para la clase TipoItem.



1. Una vez haya hecho lo anterior, es necesario que en el elemento <collection> del maestro se agregue una propiedad que indique cual es el 'resultMap' a través del cual se podrá 'mapear' los elementos contenidos en dicha colección. Para el ejemplo anterior, como la colección contiene elementos de tipo 'Detalle', se agregará el elemento **resultMap** con el identificador del 'resultMap' de Detalle:



Teniendo en cuenta lo anterior, haga los ajustes correspondientes en la configuración para el caso del modelo de Alquiler de películas.

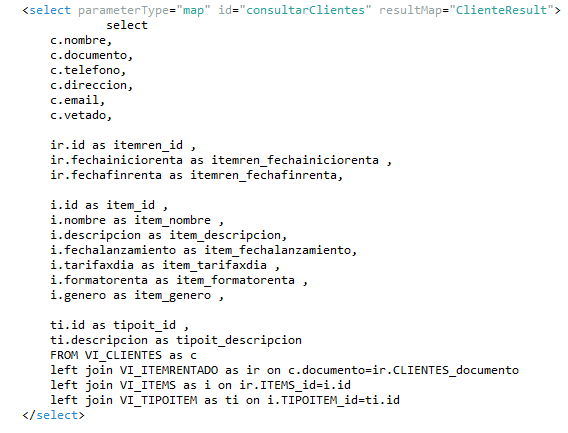


1. Si intenta utilizar el 'mapper' tal como está hasta ahora, se puede presentar un problema: ¿qué pasa si las tablas a las que se les hace JOIN tienen nombres de columnas iguales?... Con esto MyBatis no tendría manera de saber a qué atributos corresponde cada una de las columnas. Para resolver esto, si usted hace un query que haga JOIN entre dos o más tablas, siempre ponga un 'alias' con un prefijo el query. Por ejemplo, si se tiene



Se debería cambiar a:





Y posteriormente, en la 'colección' o en la 'asociación' correspondiente en el 'resultMap', indicar que las propiedades asociadas a ésta serán aquellas que tengan un determinado prefijo:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

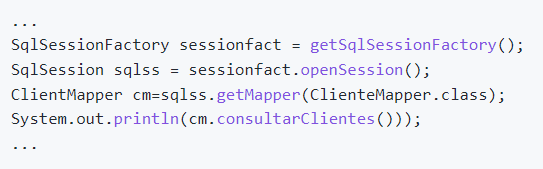
Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. Use el programa de prueba suministrado (MyBatisExample) para probar cómo a través del 'mapper' generado por MyBatis, se puede consultar un Cliente.



Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

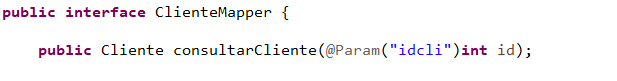
**Parte II (para el Miércoles)**

1. Configure en el XML correspondiente, la operación consultarCliente(int id) del 'mapper' ClienteMapper.

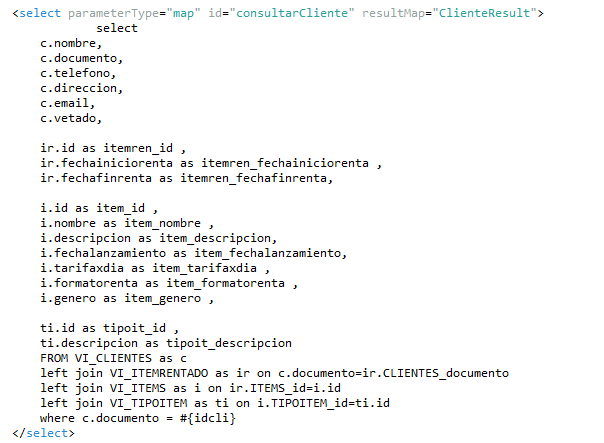
En este caso, a diferencia del método anterior (cargar todos), el método asociado al 'mapper' tiene parámetros que se deben usar en la sentencia SQL. Es decir, el parámetro 'id' de *public Cliente consultarCliente(int id);* se debe usar en el WHERE de su correspondiente sentencia SQL. Para hacer esto tenga en cuenta:

* + Agregue la anotación @Param a dicho parámetro, asociando a ésta el nombre con el que se referirá en la sentencia SQL:





* Al XML (<select>, <insert>, etc) asociado al método del mapper, agregue la propiedad *parameterType="map"* .
* Una vez hecho esto, podrá hacer referencia dentro de la sentencia SQL a este parámetro a través de: #{idcli}



1. Verifique el funcionamiento haciendo una consulta a través del 'mapper' desde MyBatisExample.
2. Configure en el XML correspondiente, la operación agregarItemRentadoACliente. Verifique el funcionamiento haciendo una consulta a través del 'mapper' desde MyBatisExample.

Texto

Descripción generada automáticamente

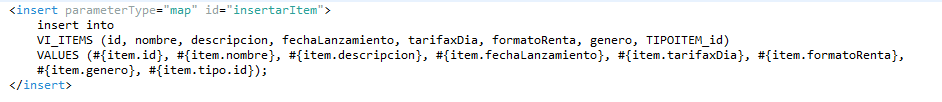
Texto

Descripción generada automáticamente

1. Configure en el XML correspondiente (en este caso ItemMapper.xml) la operación 'insertarItem(Item it). Para este tenga en cuenta:
   * Al igual que en los dos casos anteriores, el query estará basado en los parámetros ingresados (en este caso, un objeto Item). En este caso, al hacer uso de la anotación @Param, la consulta SQL se podrá componer con los atributos de dicho objeto. Por ejemplo, si al parámetro se le da como nombre ("item"): **insertarItem(@Param("item")Item it)**, en el query se podría usar #{item.id}, #{item.nombre}, #{item.descripcion}, etc. Verifique el funcionamiento haciendo una consulta a través del 'mapper' desde MyBatisExample.

Un conjunto de letras blancas en un fondo blanco

Descripción generada automáticamente con confianza media



1. Configure en el XML correspondiente (de nuevo en ItemMapper.xml) las operaciones 'consultarItem(int it) y 'consultarItems()' de ItemMapper. En este caso, tenga adicionalmente en cuenta:

* Para poder configurar dichas operaciones, se necesita el 'resultMap' definido en ClientMapper. Para evitar tener CODIGO REPETIDO, mueva el resultMap *ItemResult* de ClienteMapper.xml a ItemMapper.xml. Luego, como dentro de ClienteMapper el resultMap *ItemRentadoResult* requiere del resultMap antes movido, haga referencia al mismo usando como referencia absoluta en 'namespace' de ItemMapper.xml:

|  |
| --- |
| <resultMap type='ItemRentado' id="ItemRentadoResult">  <association ...resultMap='edu.eci.cvds.sampleprj.dao.mybatis.mappers.ItemMapper.ItemResult'></association>  </resultMap> |

ClienteMapper



Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

ItemMapper

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Imagen que contiene Texto

Descripción generada automáticamente

TipoItemMapper

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**url repositorio:** <https://github.com/LuisaGiron/ivn>