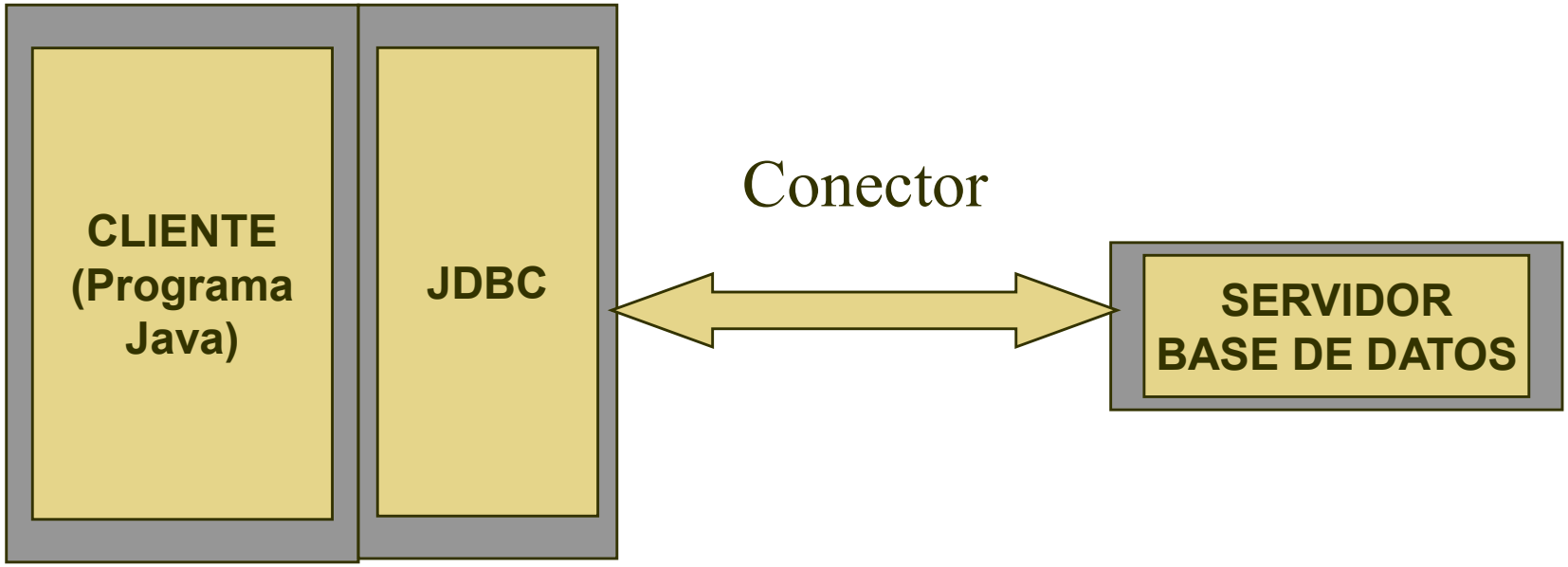
Añadiendo bases de datos


BIG
DATA
ANALYTICS
SOFTWARE
TECHNOLOGIES
INTERNET
SIZE
NEEDED
PETABYTES
PARALLEL
SAN
STORAGE
COMPLEX
DATABASES
RESEARCH
INFORMATION
MANAGEMENT
SOCIAL
LARGE
EVERY
SHARED
TIME
DISK
HUNDREDS
INCLUDE
SYSTEMS
NETWORKS
SEARCH
RECORDS
SETS
EXAMPLES
ABILITY
CURRENT
APPLIED
ARCHIVES
SENSOR
TERABYTES
CONTINUES
CITATION
DIFFICULTY
TARGET
GARTNER
MASSIVELY
BIOLOGICAL
USING
WORKING
TOLERABLE
DISTRIBUTED
CAPTURE
MANAGE
GROW
LARGER
CASE
TOOLS
KEY
WITHIN
PROCESS
DEFINITION
PRACTITIONERS
CAPACITY
TENS
LINKS
MAY

Modelo cliente-servidor para acceder a la BD



Conexión a Bases de Datos: Elementos básicos

- JDBC – API de Java para acceder a bases de datos relacionales
- Driver del gestor de base de datos
- Gestor de base de datos

API JDBC

- Define interfaces Java que son implementadas por los vendedores de los gestores de datos
- Núcleo de los componentes de la API en java.sql

Método básico para trabajar con una BD

1. Cargar el driver JDBC específico de la BD
2. Definir un objeto Connection
3. Definir un objeto Statement
4. Ejecutar consultas para crear o actualizer datos
5. Ejecutar consultas de lectura de datos
6. Cerrar los objetos Statement y Connection

Paso 1: Cargar el driver

```
/**
    *  Comprueba si el driver de conexion JDBC de MySQL esta instalado
    */
   protected void testDriver() throws Exception
   {
         try
            {
                  Class.forName ( "org.gjt.mm.mysql.Driver" );
                  System.out.println ( "Encontrado el driver de MySQL" );
            }
         catch
            (java.lang.ClassNotFoundException e)
            {
                  System.out.println("MySQL JDBC Driver no encontrado ... ");
                  throw (e);
            }
   }
```

Paso 2: Definir conexion

```
/**
    *  Devuelve conexion a MySQL
    *
    */
   protected Connection obtenerConexion (String host, String database)throws
         Exception
   {
         String url = "";
         try
            {
                  url = "jdbc:mysql://" + host + "/" + database;
                Connection con = DriverManager.getConnection(url);
                  System.out.println("Conexion establecida con " + url + "...");
                  return con;
            }
         catch (java.sql.SQLException e)
            {
                  System.out.println("Conexion NO establecida con " + url);
                  throw (e);
            }
   }
```

Paso 3 y 4: Definir y ejecutar consultas de creación/actualización

```
/**
   *  Este metodo ejecuta una sentencia de actualizacion sobre la base de
      datos
   *  @param con database connection
   *  @param sqlStatement SQL DDL or DML statement to execute
   */
  protected void ejecutarOperacion (Connection con, String sqlStatement)
      throws Exception
  {
      try
         {
                Statement s = con.createStatement();
                s.execute (sqlStatement);
                s.close ( );
         }
      catch (SQLException e)
         {
                System.out.println ("Error ejecutando sentencia SQL");
                throw (e);
         }
```

Paso 5: Ejecutar consultas de lectura

```
/**
    *  Este metodo ejecuta una sentencia de seleccion/consulta y muestra el resultado
    *  @param con database connection
    *  @param sqlStatement SQL SELECT statement to execute
    */
   protected void ejecutarConsulta(Connection con, String sqlStatement)
         throws Exception
   {
         try
            {
                  Statement s = con.createStatement ( );
                  ResultSet rs = s.executeQuery( sqlStatement );
                  while ( rs.next() )
                     {
                              String id = (rs.getObject("id").toString());
                              String text = (rs.getObject("text").toString());
                              System.out.println ( "Registro encontrado : " + id + " " + text );
                     }
                  rs.close ( );
            }
         catch (SQLException e)
            {
                  System.out.println ( "Error ejecutando la sentencia SQL" );
                  throw (e);
            }
   }
```

Código principal

```
  /* Primero, comprobar si el driver de MySQL esta instalado */
testDriver ();

/* Segundo, obtener conexion a la base de datos */
Connection con = obtenerConexion (host, database);

/* Tercero, creamos una tabla */
ejecutarOperacion(con, "create table test (id int not null,text varchar(20))" );

/* Cuarto, insertamos datos */
ejecutarOperacion(con,"insert into test (id,text) values (1,'primer valor')");
ejecutarOperacion(con,"insert into test (id,text) values (2,'segundo valor')");
ejecutarOperacion(con,"insert into test (id,text) values (3,'tercer valor')");

/* Quinto, consultamos datos y los visualizamos. */
/* Por este motivo (visualizacion) utilizamos un metodo diferente */
ejecutarConsulta(con, "select * from test");

/* Sexto, eliminamos la tabla */
ejecutarOperacion (con, "drop table test" );

/* Por ultimo cerramos la conexion con la base de datos */
con.close();
```

TAREA: Vamos a modificar la aplicación **despliegueTomcat2.war** para guardar datos en una base de datos. Sigue los siguientes pasos:

1. Abre tu BD y crea una nueva base de datos con una tabla de nombre **usuario** que contenga dos atributos: “nombre” y “password”.
2. A continuación, crea una nueva carpeta en el entorno de desarrollo con nombre **despliegueTomcatBD** para desarrollar tu aplicación. Puedes copiar la carpeta de la aplicación despliegueTomcat2 con todos sus ficheros, y cambiar el nombre de la misma.
3. A continuación, modifica el servlet para conectar con la BD y enviar una consulta para guardar los parámetros “nombre” y “password” enviados desde el formulario.
4. Compila el servlet.
5. Modifica la URL del atributo *action* del formulario de la página index.html para que encuentre el Servlet en la nueva aplicación: /**despliegueTomcatBD/login.**
6. Copia el driver JDBC de tu gestor en la carpeta /lib de tu aplicación.
7. Crea el fichero .war y despliégalo en tu entorno de producción.