



1



JSP









08/2019	KHOA CNTT - NLU	LẬP TRÌNH WEB	PHAN ĐÌNH LONG	
---------	-----------------	---------------	----------------	---

2



DATABASE



- A **database** is an organized collection of data. It is the collection of schemes, tables, queries, reports, views and other objects.



3

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

3



- A **database management system (DBMS)** is a computer software application that interacts with the user, other applications, and the database itself to capture and analyze data.
- A general-purpose DBMS is designed to allow the **definition, creation, querying, update, and administration** of databases.
- Well-known DBMSs include **MySQL, PostgreSQL**, Microsoft SQL Server, Oracle, Sybase and IBM DB2.



4

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

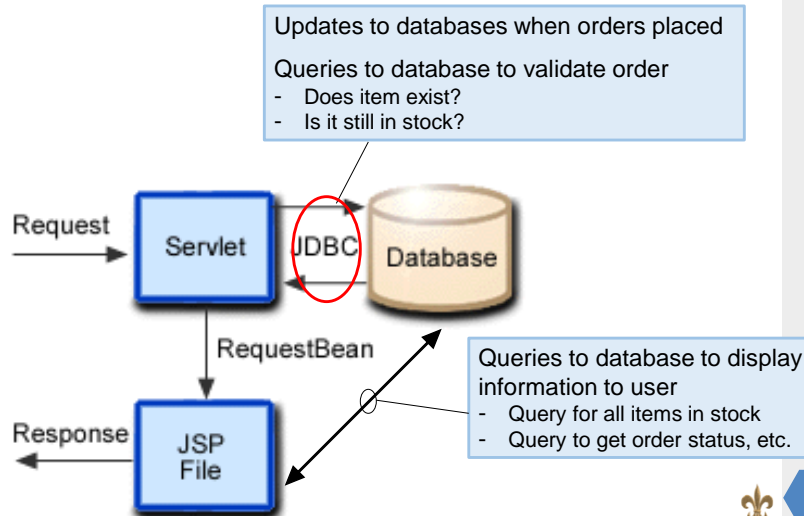
PHAN ĐÌNH LONG

43

4



DATABASE SERVER



08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

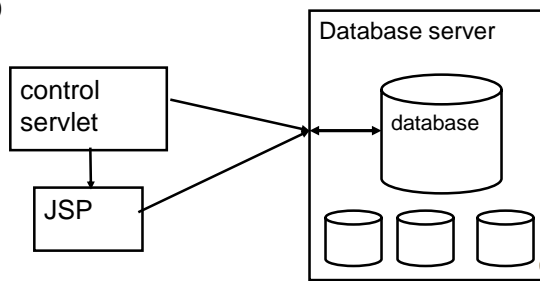
PHAN ĐÌNH LONG

43

5



- Access to database controlled by **database server**
 - Constantly running (like web container)
 - Programs communicate with it Server runs database queries and updates for databases it controls
 - Server handles **security** of database
 - Most are password-controlled
 - Examples:
 - MySQL (free)
 - Oracle
 - MS Server
 - **Not** Access!



08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

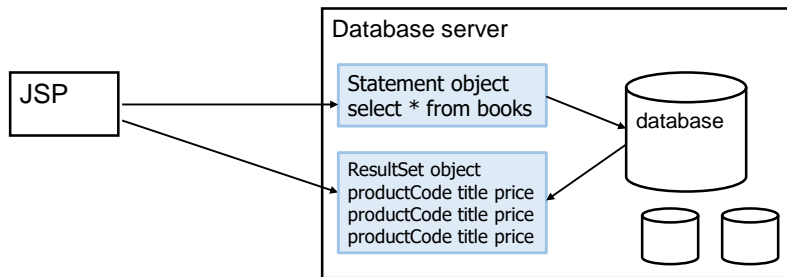
PHAN ĐÌNH LONG

43

6



- Programs create statement objects inside server
- Server executes them on the database
- Server stores results if query performed
- Program may then access those results



08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

7



MYSQL



- The MySQL software delivers a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server.
- MySQL is a database system used on the web.
- MySQL is ideal for both small and large applications
- MySQL uses standard SQL
- MySQL compiles on a number of platforms
- **MySQL is free to download and use**



8

08/2019

KHOA CNTT - NLU


LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

8

Using mysql in Xampp



What is XAMPP?

XAMPP is the most popular PHP development environment

XAMPP is a completely free, easy to install Apache distribution containing MySQL, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.

Download
Click here for other versions

XAMPP for Windows
v5.6.11 (PHP 5.6.11)

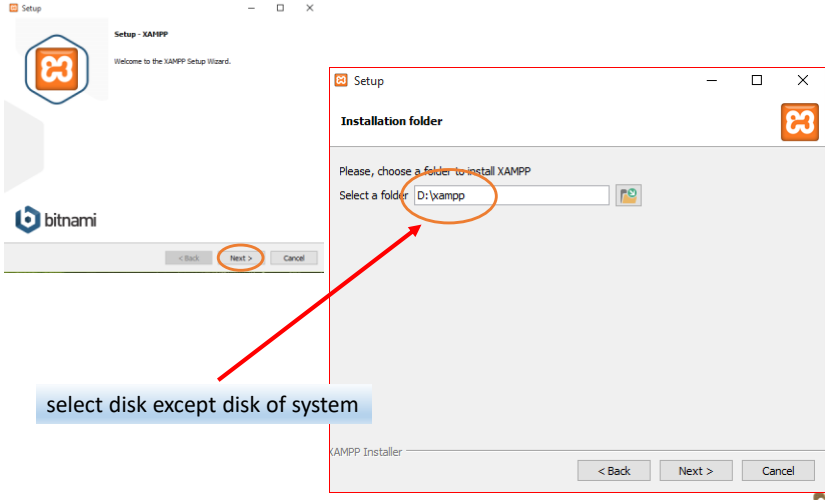
XAMPP for Linux
v5.6.11 (PHP 5.6.11)

XAMPP for OS X
v5.6.11 (PHP 5.6.11)

08/2019 KHOA CNTT - NLU LẬP TRÌNH WEB PHAN ĐÌNH LONG

9

Note



Installation folder

Please, choose a folder to install XAMPP

Select a folder: D:\xampp

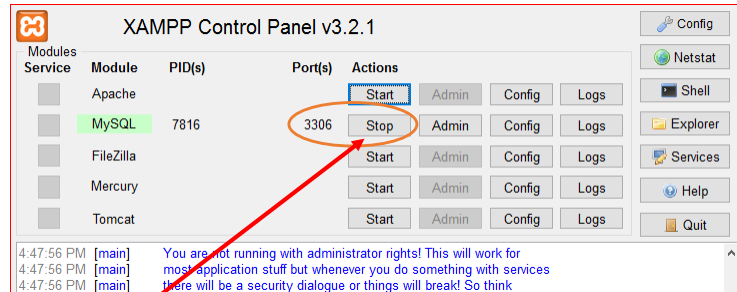
select disk except disk of system

08/2019 KHOA CNTT - NLU LẬP TRÌNH WEB PHAN ĐÌNH LONG

10



Start Mysql server



Start or stop Mysql Server



11

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

11



MySQL Manager tool



- Navicat



12

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

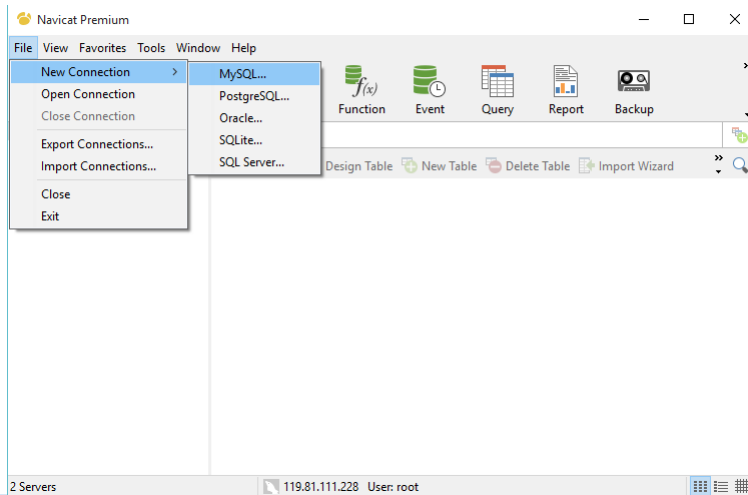
12



Create MySQL connection

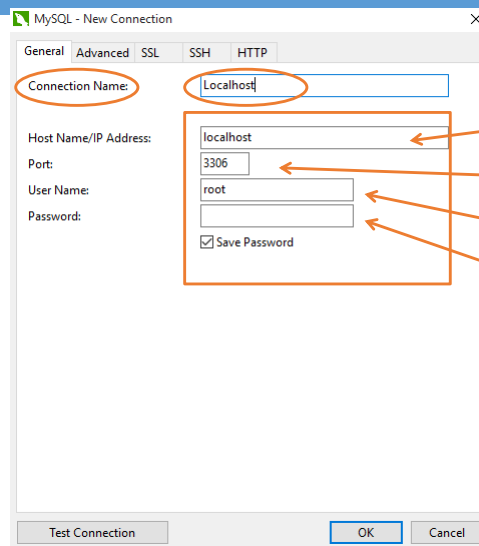


- Before connect to MySQL, we need start MySQL Server



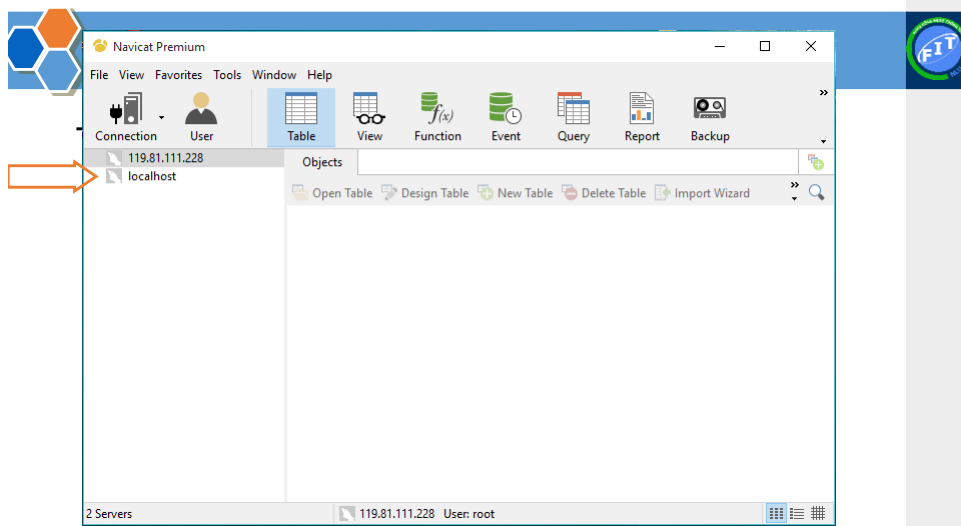
13

13



14

14



15

08/2019

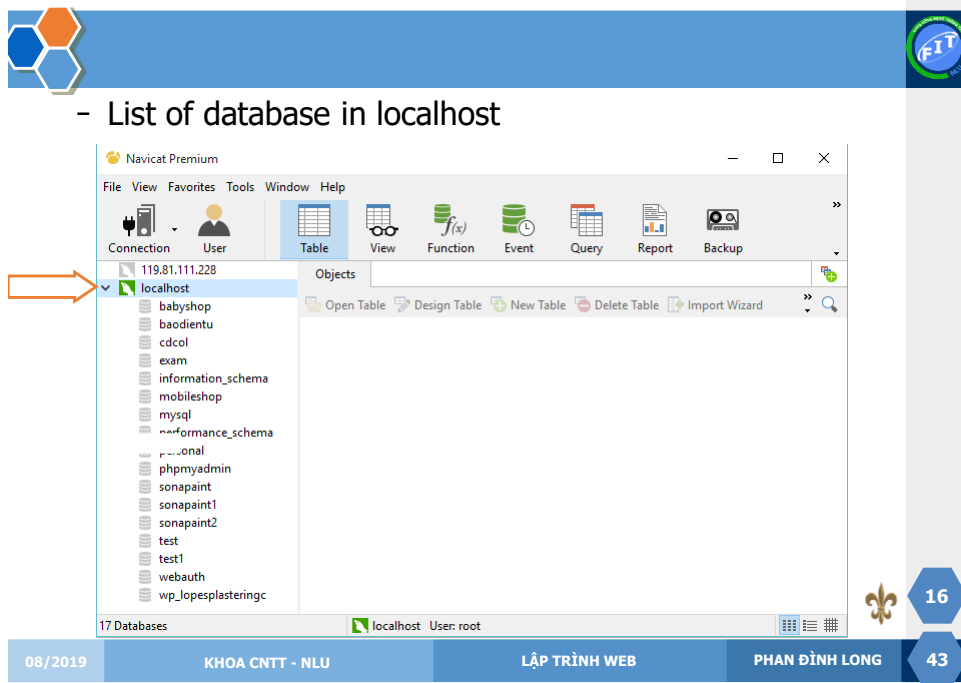
KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

15



16

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

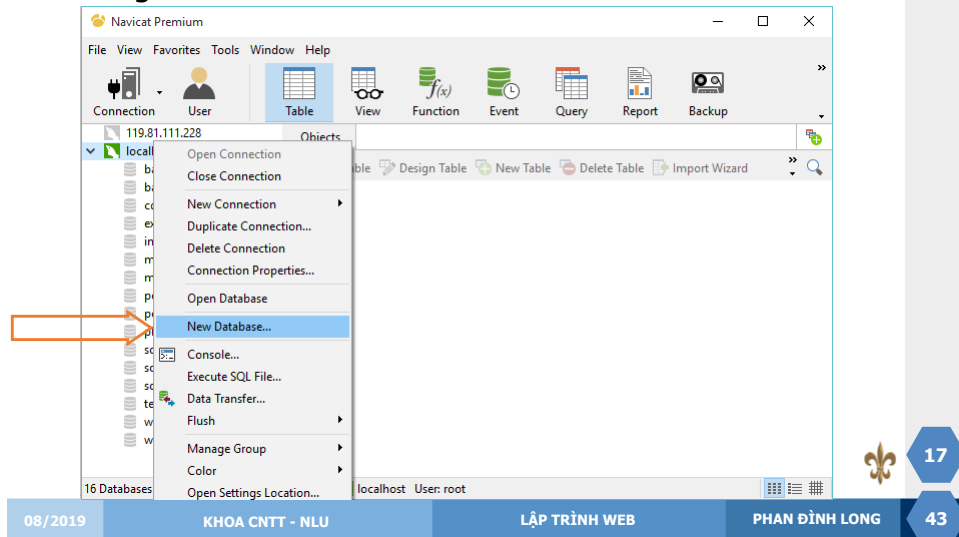
16



New Database



- Right click to localhost -> choose New Database



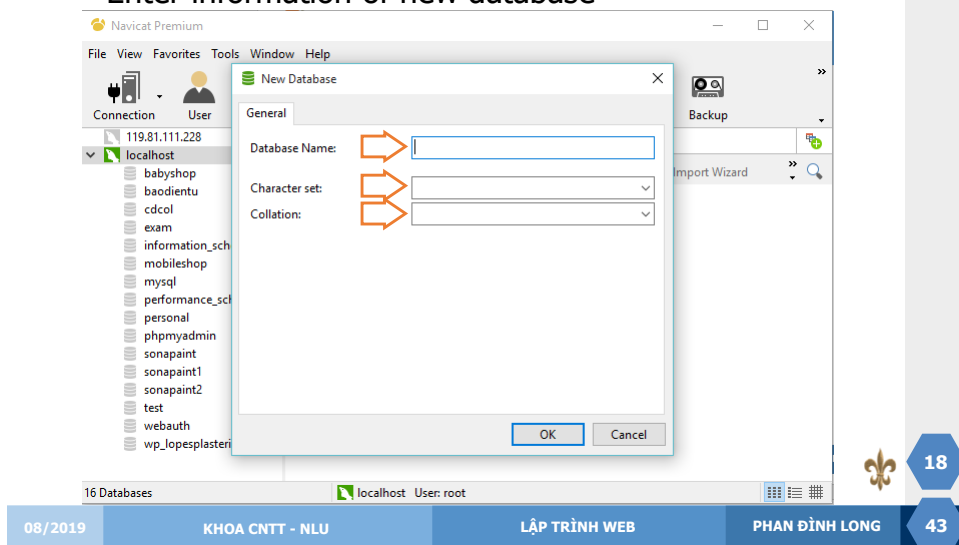
17



New Database

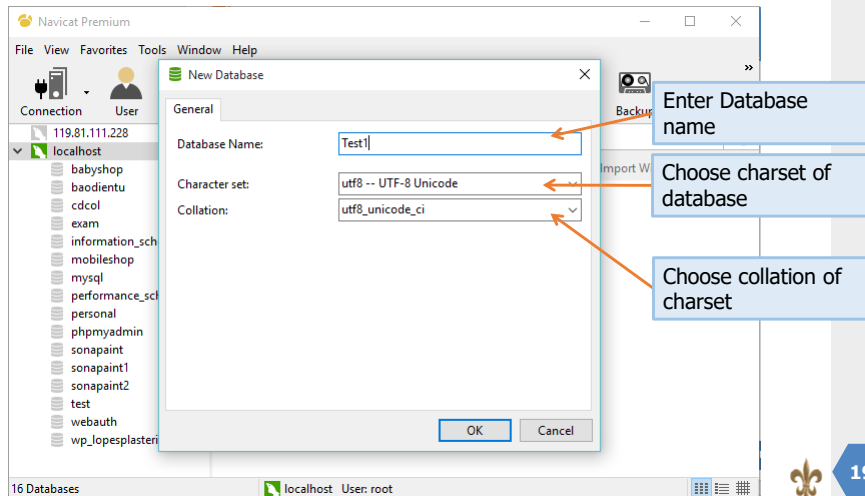


- Enter information of new database



18

New Database



19

08/2019

KHOA CNTT - NLU

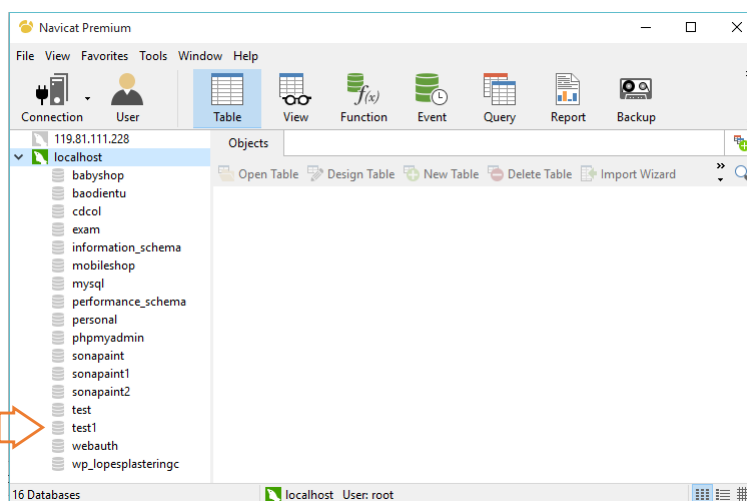
LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

19

New Database



20

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

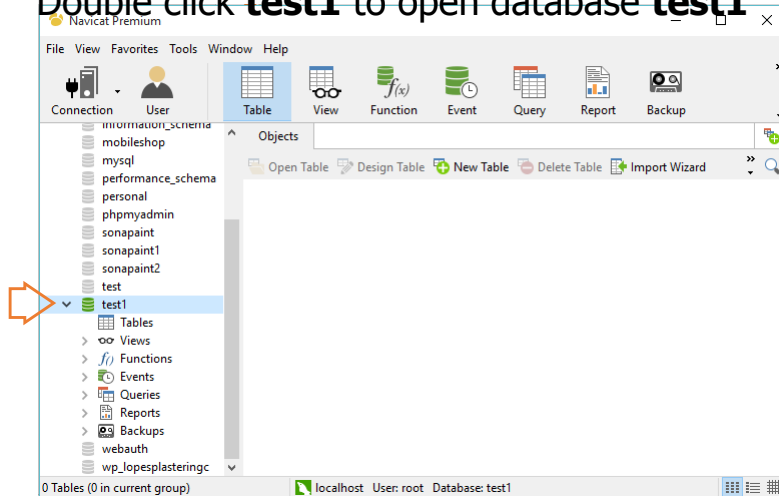
PHAN ĐÌNH LONG

43

20



Double click **test1** to open database **test1**



21

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

21



SQL review



- SQL (Structured Query Language) is a special purpose programming language designed for managing data held in a relational database management system.
- The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control



22

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

22



Select



- Syntax:
 - SELECT field FROM table WHERE condition;
- EX:
 - SELECT ID, NAME, SALARY FROM CUSTOMERS;

Data in table					Data after query		
ID	NAME	AGE	ADDRESS	SALARY	ID	NAME	SALARY
1	Ramesh	32	Ahmedabad	2000.00	1	Ramesh	2000.00
2	Khilan	25	Delhi	1500.00	2	Khilan	1500.00
3	kaushik	23	Kota	2000.00	3	kaushik	2000.00
4	Chaitali	25	Mumbai	6500.00	4	Chaitali	6500.00
5	Hardik	27	Bhopal	8500.00	5	Hardik	8500.00
6	Komal	22	MP	4500.00	6	Komal	4500.00
7	Muffy	24	Indore	10000.00	7	Muffy	10000.00

23

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

23



Insert



- Syntax:
 - INSERT INTO **TABLE** (list column) VALUES (list value);
- or**
- INSERT INTO **TABLE** VALUES (list value);
- EX:
 - INSERT INTO **CUSTOMERS** (**ID,NAME,AGE, ADDRESS,SALARY**) **VALUES** (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);
 - INSERT INTO **CUSTOMERS** **VALUES** (2, 'Muffy', 24, 'Indore', 10000.00);

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Muffy	24	Indore	10000.00



24

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

24



Update



- Syntax:
 - UPDATE table SET [data change] WHERE [condition];
- Ex: UPDATE CUSTOMERS SET ADDRESS = 'Pune' WHERE ID = 6;

Before update					After update				
ID	NAME	AGE	ADDRESS	SALARY	ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00	1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00	2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00	3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00	4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00	5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00	6	Komal	22	Pune	4500.00
7	Muffy	24	Indore	10000.00	7	Muffy	24	Indore	10000.00



25

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

25



Delete



- Syntax:
 - DELETE FROM **table** WHERE [condition];
- Ex:
 - DELETE FROM **CUSTOMERS** WHERE ID = 6;

Before delete					After delete				
ID	NAME	AGE	ADDRESS	SALARY	ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00	1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00	2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00	3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00	4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00	5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00	7	Muffy	24	Indore	10000.00
7	Muffy	24	Indore	10000.00					



26

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

26



JDBC



- **JDBC** is a Java database connectivity technology (Java Standard Edition platform) from Oracle Corporation. This technology is an API for the Java programming language that defines how a client may access a database.
- JDBC is oriented towards relational databases.
- JDBC provides methods for querying and updating data in a database.



27

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

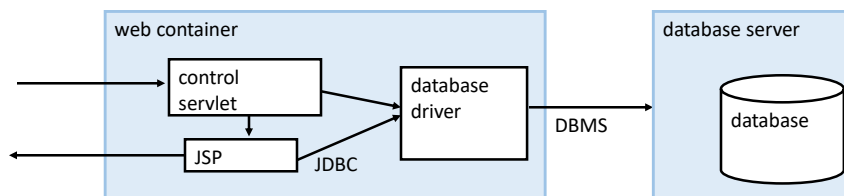
27



Database Drivers



- Database server does not understand JDBC commands
- Only understands its own DBMS protocols
 - Each server has its own DBMS
- Need a database driver to perform translation
 - Obtain from database server provider
 - Install in Java libraries



28

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

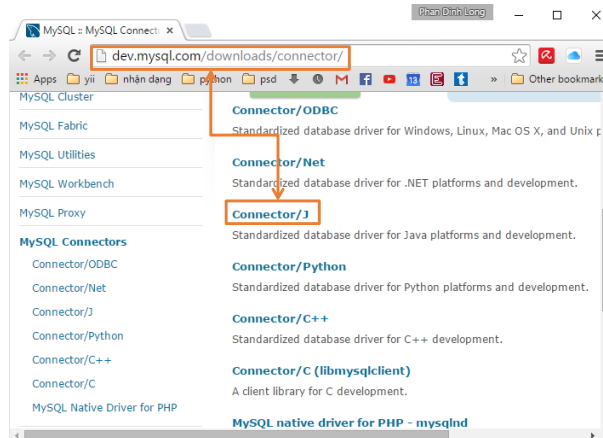
28



MySQL JDBC DRIVER



- Go to link: <http://dev.mysql.com/downloads/connector/> and down load **connector j**



29

08/2019

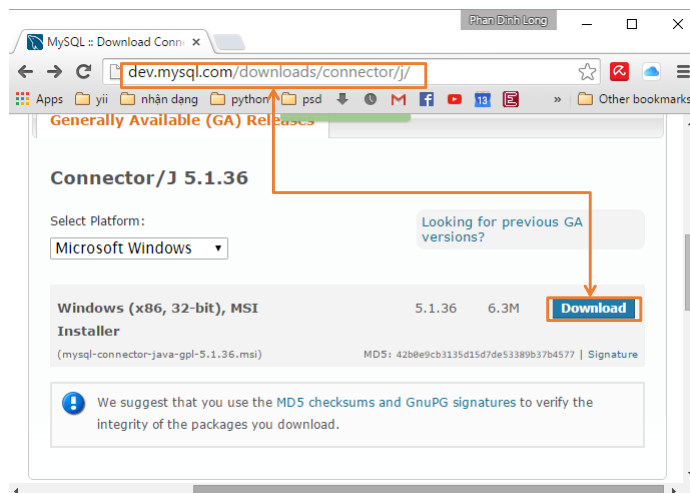
KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

29



30

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

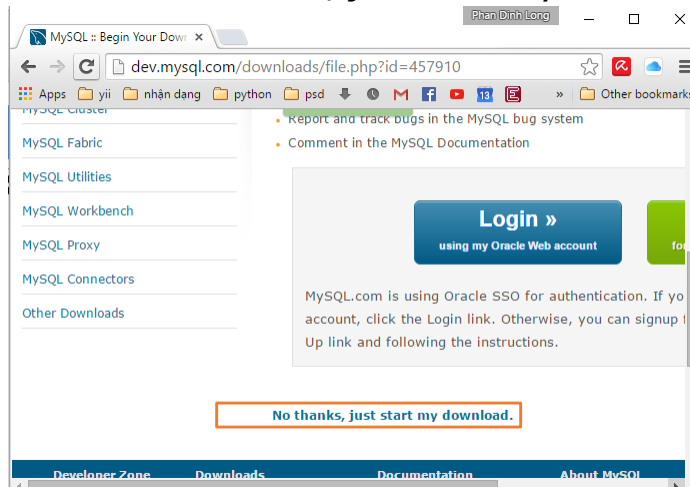
PHAN ĐÌNH LONG

43

30



- Choose no thanks, just start my download



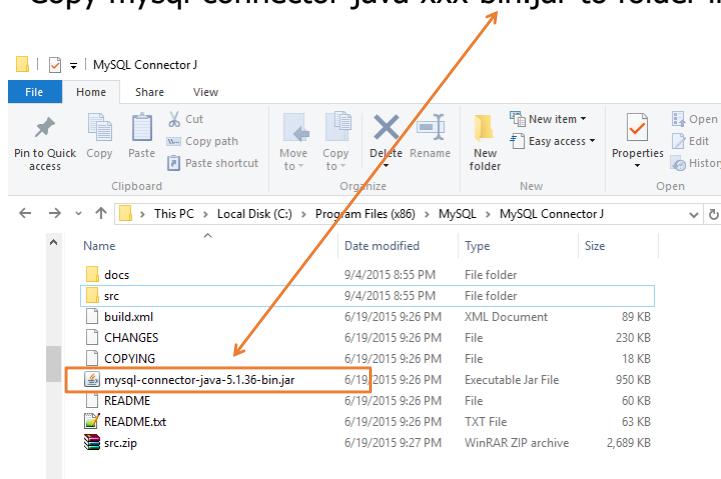
31



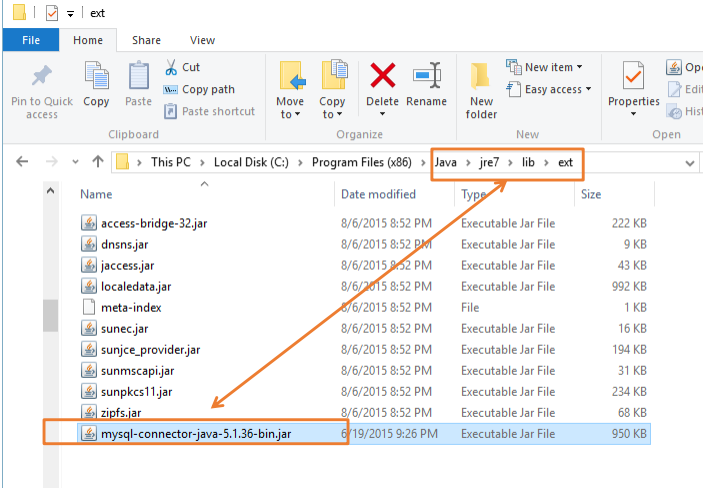
Install MySQL lib



- Copy mysql-connector-java-xxx-bin.jar to folder lib



32



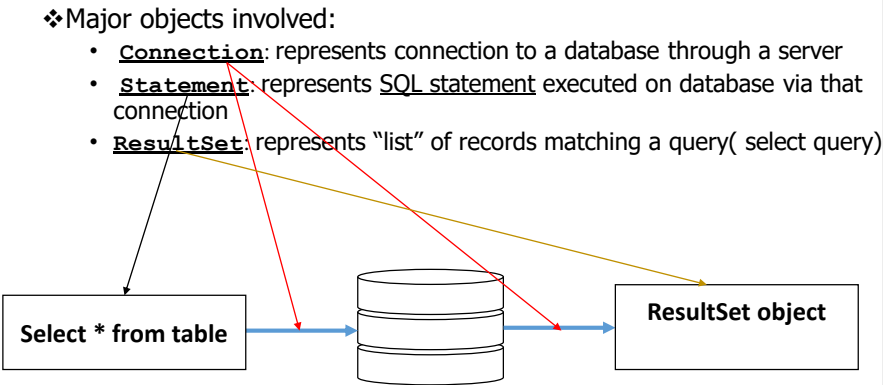
File Explorer window showing the path: This PC > Local Disk (C:) > Program Files (x86) > Java > jre7 > lib > ext. The file 'mysql-connector-java-5.1.36-bin.jar' is highlighted. An orange arrow points from the file name to the 'lib' folder in the address bar.

Name	Date modified	Type	Size
access-bridge-32.jar	8/6/2015 8:52 PM	Executable Jar File	222 KB
dnsns.jar	8/6/2015 8:52 PM	Executable Jar File	9 KB
jaccess.jar	8/6/2015 8:52 PM	Executable Jar File	43 KB
localedata.jar	8/6/2015 8:52 PM	Executable Jar File	992 KB
meta-index	8/6/2015 8:52 PM	File	1 KB
sunec.jar	8/6/2015 8:52 PM	Executable Jar File	16 KB
sunjce_provider.jar	8/6/2015 8:52 PM	Executable Jar File	194 KB
sunmscapi.jar	8/6/2015 8:52 PM	Executable Jar File	31 KB
sunpkcs11.jar	8/6/2015 8:52 PM	Executable Jar File	234 KB
zipfs.jar	8/6/2015 8:52 PM	Executable Jar File	68 KB
mysql-connector-java-5.1.36-bin.jar	8/19/2015 9:26 PM	Executable Jar File	950 KB

33

08/2019 KHOA CNTT - NLU LẬP TRÌNH WEB PHAN ĐÌNH LONG 43

33



JDBC Connection

- ❖ Major objects involved:
 - Connection: represents connection to a database through a server
 - Statement: represents SQL statement executed on database via that connection
 - ResultSet: represents "list" of records matching a query(select query)

Diagram illustrating the flow: Select * from table → Database → ResultSet object.

34

08/2019 KHOA CNTT - NLU LẬP TRÌNH WEB PHAN ĐÌNH LONG 43

34



Create connection



- Load driver
 - `Class.forName("driver name")`
- Create Connection Object to open connection to database

```
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Connection"%>
<%
Class.forName("com.mysql.jdbc.Driver");
Connection conn=DriverManager.getConnection(
    "jdbc:mysql://localhost/test", "root", "");
%>
```

Load driver mysql

Create Connection Object



35

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

35



- To connect database server we need:
 - Type of server (MySQL, oracle, MS SQL,...)
 - Driver for database server.
 - Url to access to server.
 - Username and password to access to server.
- Base URL:
 - Syntax: `jdbc:server type:url of server/database name`
 - Ex: `jdbc:mysql://localhost/test`



36

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

36



Exception Handling in JDBC



- Any database-related statement may throw an **SQLException**
 - Your code must put in try/catch block
 - May also need to catch other exceptions
 - **ClassNotFoundException** for missing database driver

```
<%
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conn = DriverManager.getConnection(
            "jdbc:mysql://localhost/test", "root", "");
    } catch (ClassNotFoundException e1) {
        // NO DRIVER
    } catch (SQLException e2){
        // CONNECTION ERROR
    }
%>
```

37

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

37



Insert data



- Create statement:
 - Statement s=conn.createStatement();
- Execute query: using method executeUpdate()

```
String username= ...
String password= ...
Class.forName("com.mysql.jdbc.Driver");
Connection conn = DriverManager.getConnection(
    "jdbc:mysql://localhost/test", "root", "");
Statement s=conn.createStatement();
s.executeUpdate("INSERT INTO user(username,password)
    VALUES("+username+", "+password+"");
```

38

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

38



Udata data



- Using like insert data.

```
String id= ...
String productName= ...
Class.forName("com.mysql.jdbc.Driver");
Connection conn = DriverManager.getConnection(
    "jdbc:mysql://localhost/test", "root", "");
Statement s=conn.createStatement();
s.executeUpdate("UPDATE product set name = '"+productName+"'
where id = "+id);
```



39

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

39



Select data



- Create statement:
 - Statement s=conn.createStatement();
- Execute query: using method executeUpdate()
 - Using ResultSet object to get data return.

```
String listProduct= ...
Class.forName("com.mysql.jdbc.Driver");
Connection conn = DriverManager.getConnection(
    "jdbc:mysql://localhost/test", "root", "");
Statement s=conn.createStatement();
ResultSet rs=s.executeQuery("SELECT * FROM product where
listProduct = "+listProduct);
```



40

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

40



Reading ResultSets



- How to access:
 - Only access current record.
 - Move next record to continues read ResultSets.
- Syntax:
 - Next record: using `next()` to move next record. Method return a value Boolean, true if can read next record, false if can't read next record.

```
while(ResultSet.next()){
    // read data here
}
```



41

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

41



- Read data:
 - Read by column index:
DATATYPE value = `ResultSet.getType(int column_index)`.
 - Read by column name:
DATATYPE value = `ResultSet.getType(int column_name)`.

Specify type data is to be read in as
 varChar → `getString`
 int → `getInt`
 double → `getDouble`



42

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

42




```

try {
    String listProduct= ...
    Class.forName("com.mysql.jdbc.Driver");
    Connection conn = DriverManager.getConnection(
        "jdbc:mysql://localhost/test", "root", "");
    Statement s=conn.createStatement();
    ResultSet rs=s.executeQuery("SELECT * FROM product
        where listProduct = "+listProduct);
    while(rs.next()){
        rs.getString(1);//read by column index
        rs.getString("name");//read by column name
    }

} catch (ClassNotFoundException e1) {
    // NO DRIVER
} catch (SQLException e2){
    // CONNECTION ERROR
}

```

43

08/2019
KHOA CNTT - NLU
LẬP TRÌNH WEB
PHAN ĐÌNH LONG

43

43



Efficient and Safe Database

44



Problem



- We have a product. Now quantity of this product is 100, and 2 client modify data with a processor.
- Processor working with client 1:
 - Client 1 load quantity of this product and view, quantity of product is 100.
- Processor working with client 2:
 - Client 2 load quantity of this product, too. And quantity still is 100.
 - Client 2 set quantity reduced 1 product. (quantity = 99)
- Processor swap back client 1:
 - Client 1 Client 2 set quantity reduced 1 product(→ quantity = 99)
- Expected results: **98**



45

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

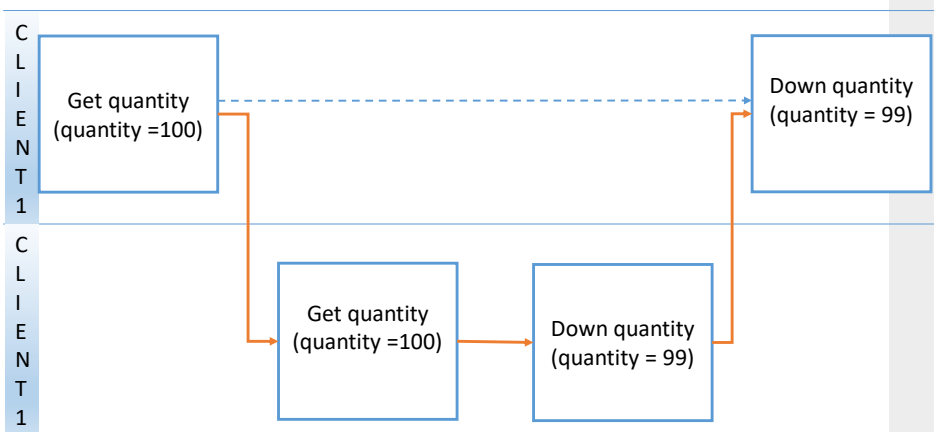
PHAN ĐÌNH LONG

43

45



Why quantity product is 97



46

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

46



Solution



- ❖ Can declare sections of code to be synchronized
 - Only one thread may execute it at a time
 - Another thread cannot start the code until the first has finished it

❖ Syntax: **synchronized**(object) { code }

Only one thread at a time should be able to execute this **code** on this **object**



47

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

47



```
Statement s;
synchronized(s){
    try{
        s=conn.createStatement();
        ResultSet rs=s.executeQuery("SELECT quantity FROM
                                   product where idproduct = "+id);
        rs.next() ;
        int quantity=rs.getInt("quantity");
        quantity-=1;
        s.executeUpdate("UPDATE product SET quantity = "
                        +quantity+" WHERE idproduct = "+id);
    }catch(SQLException e){
        //BAD SQL
    }
}
```



48

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

48



Problem



- We have a login form with 2 input
 - Username
 - Password
- And SQL to login:
 - `SELECT * FROM user where username = "username value" and password = "password value"`
 - Ex: `SELECT * FROM user where username = "ti" and password = "123456"`



49

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

49



- What happen when user input username is: "or "1"="1"
- ❖ SQL query now: `SELECT * FROM user where username = ""or "1"="1" and password = "password value"`
- ➔ **Query always true**



50

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

50



Solution → PreparedStatement



- **Prepared statement** is a feature used to execute the same or similar database statements repeatedly with high efficiency.
 - **Prepare:** The statement template is created by the application and sent to the database management system (DBMS). Certain values are left unspecified, called parameters, placeholders or bind variables (labelled "?" below):
 - **INSERT INTO PRODUCT (name, price) VALUES (?,?)**



51

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

51



- The DBMS **parses, compiles**, and performs query optimization on the statement template, and **stores the result without executing it**.
- **Execute:** At a later time, the application supplies (or binds) values for the parameters, and the DBMS executes the statement (possibly returning a result). The application may execute the statement as many times as it wants with different values.



52

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

52



Using preparedStatement



- Define preparedStatement:
 - `PreparedStatement pstmt=null;`
- Create preparedStatement:
 - `pstmt=conn.prepareStatement("SELECT quantity FROM product where idproduct = ?");`
- Set value to SQL:
 - Syntax: using method `setTYPE(index, value);`

`pstmt.setInt(1, id);`



53

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

53



```

pstmt=conn.prepareStatement("SELECT quantity FROM
                             product where idproduct = ?");
pstmt.setInt(1, id);
ResultSet rs=pstmt.executeQuery();
rs.next() ;
int quantity=rs.getInt("quantity");
quantity-=1;
pstmt=conn.prepareStatement("UPDATE product SET quantity
                             = ? WHERE idproduct = ?");
pstmt.setInt(1, quantity);
pstmt.setInt(2, id);
pstmt.executeUpdate();

```



54

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

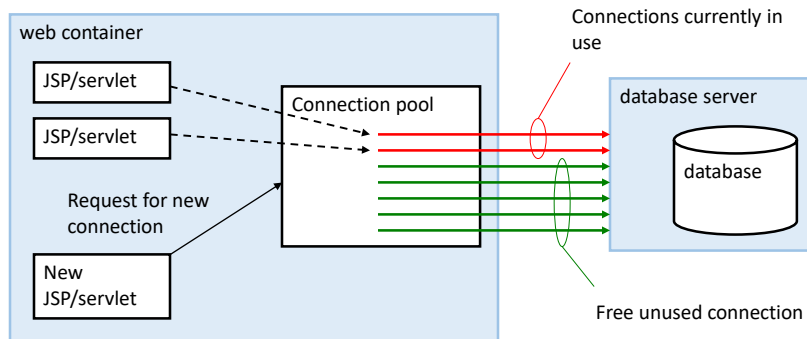
54



Connection Pooling



- When connection requested:
 - Get unused connection from pool



55

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

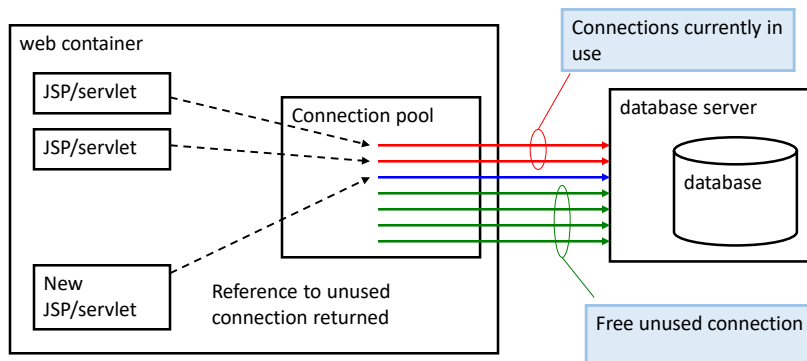
55



Connection Pooling



- When connection requested:
 - Connection used by servlet/JSP



56

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

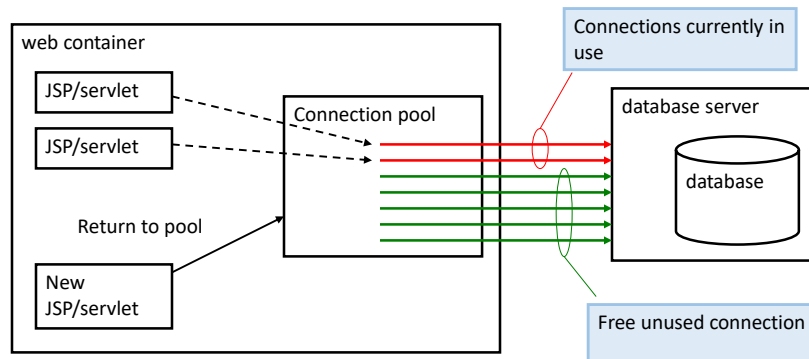
56



Connection Pooling



- When finished, JSP/servlet returns the connection back to the pool
 - Now free for use by another



08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

57

43

57



Connection Pooling



- Unlike prepared statement, no built in Java methods/classes
 - Write your own
 - <http://java.sun.com/developer/onlineTraining/Programming/JDCBook/conpool.html>
 - Third party classes
 - `dbConnectionBroker`, etc.
 - Build components directly into `web.xml/context.xml` files
 - Page 466 of text
 - Not well supported by Tomcat



58

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

58



Connection Pooling



- Usually static object
 - Automatically constructs connections first time `getConnection` called
- Usually provide following methods:
 - `ConnectionPool.getInstance()`
 - `freeConnection()`

- Example:

```
Connection connection = ConnectionPool.getInstance();
// Code that creates statement, executes queries, etc.
connection.freeConnection();
```



59

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

59



Connection Pooling



- Required parameters:
 - Driver name
 - `"com.mysql.jdbc.Driver"`
 - Url, name, and password
 - `"jdbc:mysql://localhost/bookstore", "root", "sesame"`
 - Number of initial connections to create
 - Usually a few hundred to a few thousand
 - Timeout for idle connections
 - Time after which idle connections are returned to pool automatically
 - Important to prevent pool running out!

Necessary so connection pool can connect to database



60

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

60



DEMO



61

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

61



Q & A



62

08/2019

KHOA CNTT - NLU

LẬP TRÌNH WEB

PHAN ĐÌNH LONG

43

62