บทที่ 3

JDBC

(Java Database Connectivity)

ธีระยุทธ ทองเครือ

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยขอนแก่น

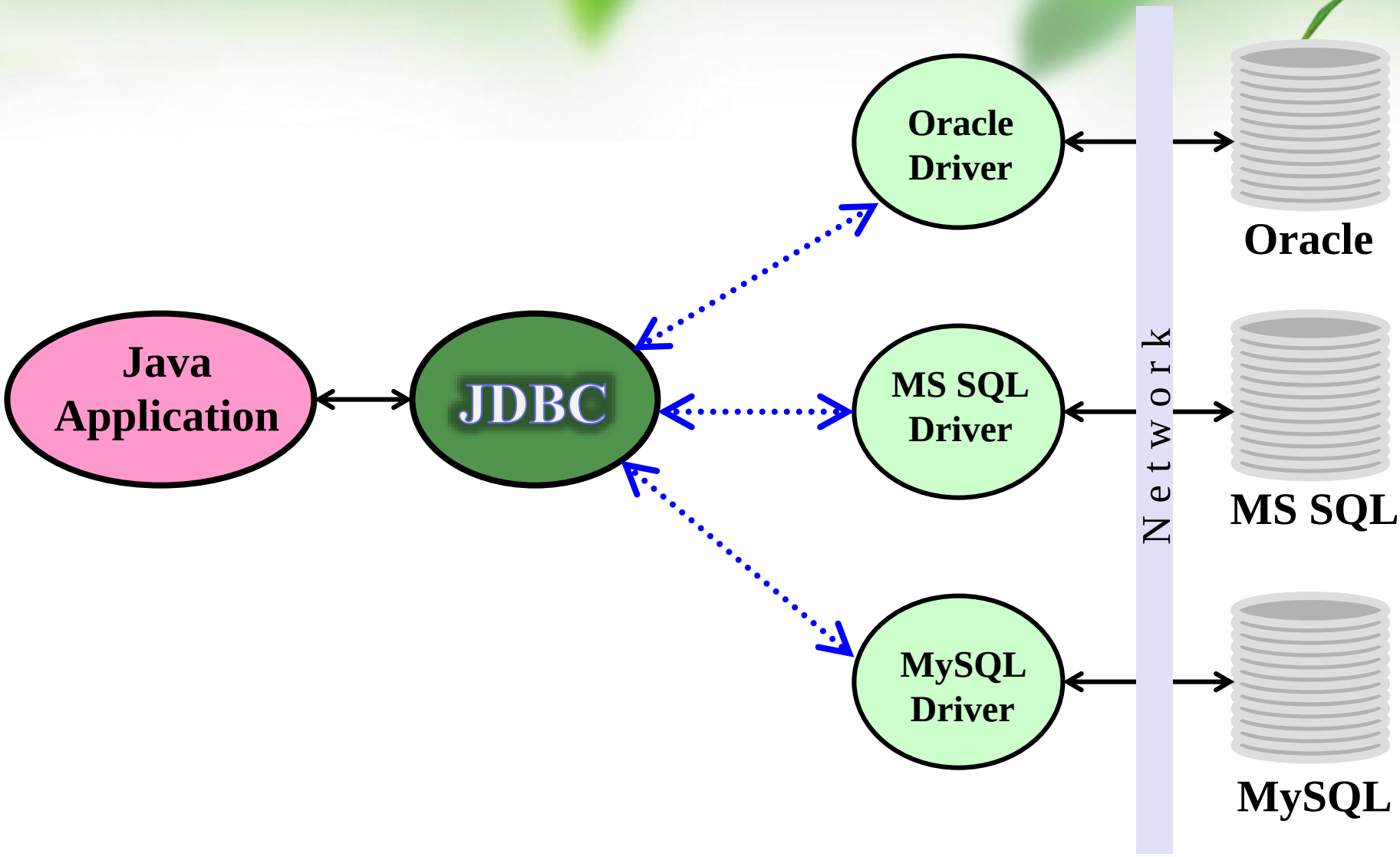
แนะนำ JDBC

- JDBC คือ API มาตรฐานสำหรับใช้ในการติดต่อฐานข้อมูลในภาษาจาวา

- JDBC ประกอบด้วยเมธอด ที่ใช้สำหรับเข้าถึงฐานข้อมูล
  - สร้างฐานข้อมูล
  - Query ข้อมูล
  - insert/update/delete

- JDBC API อยู่ใน package ชื่อ java.sql (อยู่ใน Java SE)

สถาปัตยกรรม JDBC



JDBC Driver

- JDBC Driver คือ API ที่ผู้ผลิตซอฟท์แวร์ฐานข้อมูล พัฒนาขึ้นตามข้อกำหนดของ JDBC เพื่อสนับสนุนการเชื่อมต่อกับฐานข้อมูลของตนเอง

- JDBC Driver จะอยู่ในรูปแบบของไฟล์ที่บีบอัดในนามสกุล .jar เพื่อให้นักพัฒนานำไปเก็บไว้ในแอปพลิเคชัน

JDBC Driver

- MySQL

  https://dev.mysql.com/downloads/connector/j/

- Oracle

  http://www.oracle.com/technetwork/database/features/jdbc/

- Microsoft SQL Server

  https://msdn.microsoft.com/en-us/sqlserver/aa937724.aspx

ขั้นตอนการติดต่อฐานข้อมูลด้วย JDBC

1. โหลดคลาส JDBC Driver

2. กำหนด URL สำหรับติดต่อกับฐานข้อมูล

3. สร้าง Connection

4. สร้าง Statement Object

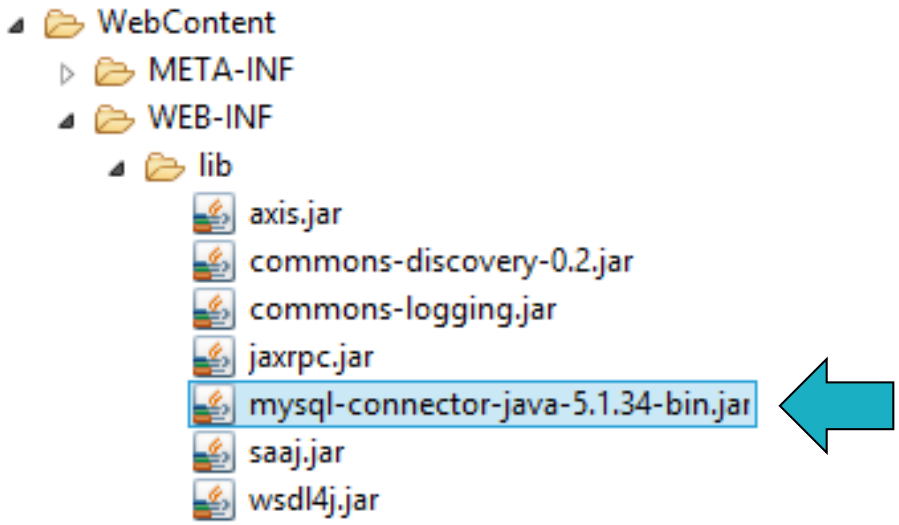
5. ส่งคำสั่ง SQL ไปยังฐานข้อมูล

6. อ่านผลลัพธ์ที่ฐานข้อมูลส่งกลับ

7. ปิด Connection

1. โหลดคลาส JDBC Driver

- คลาส JDBC Driver จะอยู่ในไฟล์ .jar ของฐานข้อมูลที่ต้องการติดต่อ (ในที่นี้คือ MySQL) เก็บไว้ใน WebContent\WEB-INF\lib\



ชื่อคลาส JDBC Driver ของแต่ละฐานข้อมูล

- MySQL

```
Class.forName("com.mysql.jdbc.Driver");
```

- Oracle

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

- Microsoft SQL Server

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
```

ตัวอย่าง

```
public class TestDB {

  public static void main(String[] args) {

    try {
        Class.forName("com.mysql.jdbc.Driver");
        ...

    } catch (ClassNotFoundException e) {
        System.err.println("Error loading driver: " + e);
        System.exit(0);
    }

    System.out.println("Loaded JDBC driver successfully");
  }

}
```

2. กำหนด URL สำหรับติดต่อกับฐานข้อมูล

❖ URL สำหรับติดต่อกับฐานข้อมูล เป็น String ทั่วไป ซึ่งมีรูปแบบดังนี้

```
String dbURL = "jdbc:mysql://localhost/blueshop";
```

ชื่อ Protocol

ชื่อ domain หรือ ip ของ database server

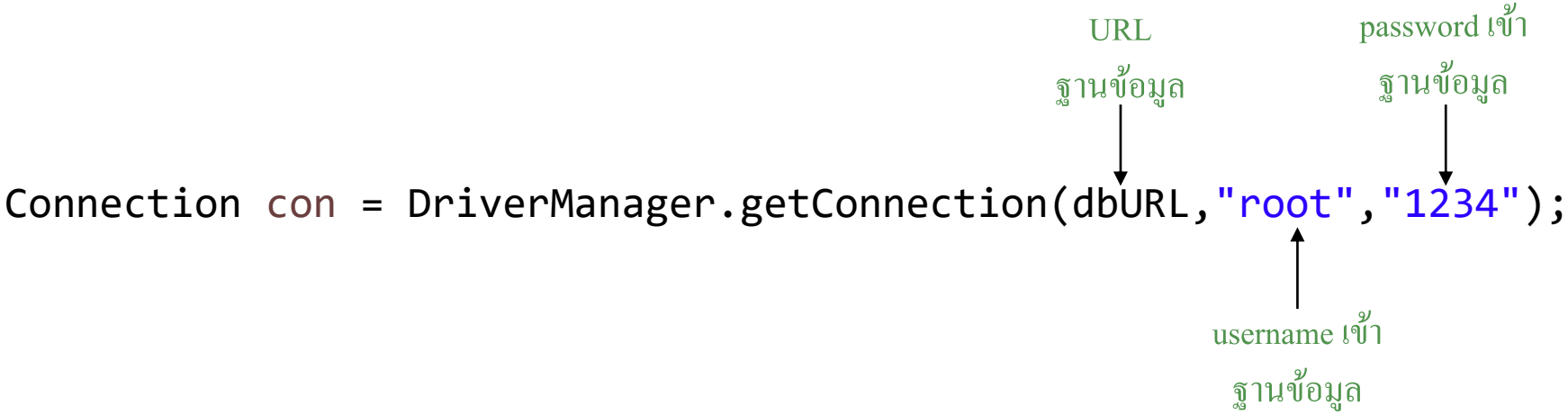
ชื่อฐานข้อมูล

❖ URL สำหรับติดต่อกับฐานข้อมูล ที่ระบุว่าให้มีการเข้ารหัสเป็นชุดอักขระที่ต้องการ

```
"jdbc:mysql://localhost/blueshop?characterEncoding=utf-8";
```

3. สร้าง Connection

- ประกาศ object ของคลาส Connection สำหรับเก็บการเชื่อมต่อ
- เรียกเมธอด getConnection( ) จากคลาส DriverManager ซึ่งมีอาร์กิวเมนต์ดังนี้



ตัวอย่าง

```
public class TestDB {

  public static void main(String[] args) {

    try {
        Class.forName("com.mysql.jdbc.Driver");
        String dbURL = "jdbc:mysql://localhost/blueshop?characterEncoding=utf-8";
        Connection con = DriverManager.getConnection(dbURL, "root", "");

    } catch (ClassNotFoundException e) {
        System.err.println("Error loading driver: " + e);
        System.exit(0);
    } catch (SQLException e) {
        System.err.println("Error database connection: " + e);
        System.exit(0);
     }

     System.out.println("Database connection successfully");
  }
}
```

4. สร้าง Statement Object

- Statement Object ใช้ในการส่งคำสั่ง SQL ไปยังฐานข้อมูล
- สร้างขึ้นโดยการเรียกเมธอด createStatement( ) จาก Connection Object

```
Statement statement = con.createStatement();
```

5. ส่งคำสั่ง SQL ไปยังฐานข้อมูล

- สร้าง String ในรูปแบบภาษา SQL
- เรียกเมธอด executeQuery() จาก Statement Object โดยส่งอาร์กิวเมนต์ที่เป็นสตริงภาษา SQL
- ประกาศตัวแปรของคลาส ResultSet เพื่อรับค่าผลลัพธ์

```
ResultSet resultSet = statement.executeQuery("SELECT * FROM product");
```

Object สำหรับเก็บผลลัพธ์ที่ฐานข้อมูลส่งกลับ

String ที่อยู่ในรูปคำสั่ง SQL

ตัวอย่าง

```
public class TestDB {

  public static void main(String[] args) {

    try {
        Class.forName("com.mysql.jdbc.Driver");
        String dbURL = "jdbc:mysql://localhost/blueshop?characterEncoding=utf-8";
        Connection con = DriverManager.getConnection(dbURL, "root", "");
        Statement statement = con.createStatement();
        ResultSet resultSet = statement.executeQuery("SELECT * FROM product");

    } catch (ClassNotFoundException e) {
        System.err.println("Error loading driver: " + e);
        System.exit(0);
    } catch (SQLException e) {
        System.err.println("Error database connection: " + e);
        System.exit(0);
     }
  }
}
```

5. อ่านผลลัพธ์ที่ฐานข้อมูลส่งกลับ

- ผลลัพธ์ที่ได้จากฐานข้อมูลจะถูกเก็บลง ResultSet Object
- สามารถโหลดข้อมูลทีละแถวด้วยเมธอด next( )
- เข้าถึงข้อมูลในแต่ละคอลัมน์ด้วยเมธอด getXxxx( )

```
ResultSet resultSet = statement.executeQuery("SELECT * FROM product");

while (resultSet.next()) {
    int pid = resultSet.getInt("pid");
    String pname = resultSet.getString("pname");
    String pdetail = resultSet.getString("pdetail");
    int price = resultSet.getInt("price");

    System.out.println(pid + "," + pname + ","
                              + pdetail + "," + price);
}
```

ดึงข้อมูลออกมาเป็น integer

ดึงข้อมูลของแถวปัจจุบัน คอลัมน์ชื่อ pname

ดึงข้อมูลออกมาเป็น String

6. การปิด Connection

- เรียกเมธอด close() จาก Connection Object เพื่อหยุดการเชื่อมต่อกับฐานข้อมูล

- การปิด Connection ควรทำทุกครั้ง เพื่อเพิ่มประสิทธิภาพการทำงานกับฐานข้อมูล

ตัวอย่าง

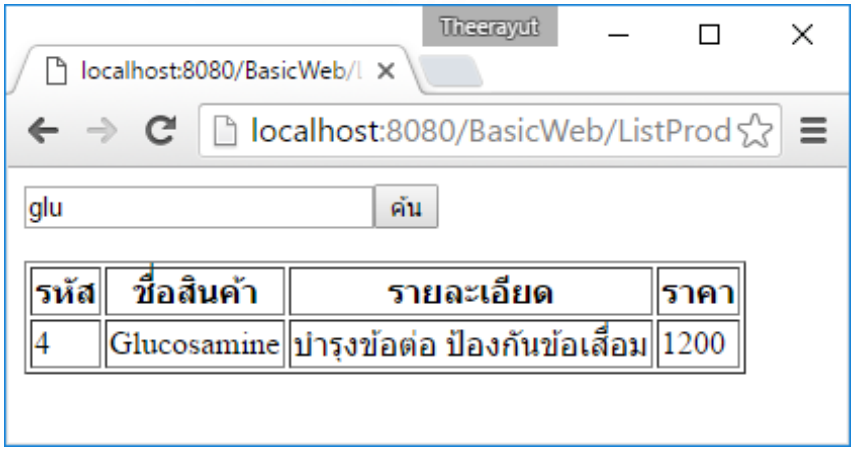
```
public class TestDB {
  public static void main(String[] args) {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        String dbURL = "jdbc:mysql://localhost/blueshop?characterEncoding=utf-8";
        Connection con = DriverManager.getConnection(dbURL, "root", "");
        Statement statement = con.createStatement();
        ResultSet resultSet = statement.executeQuery("SELECT * FROM product");
        while (resultSet.next())...
                ...

        con.close();

    } catch (ClassNotFoundException e) {
        System.err.println("Error loading driver: " + e);
        System.exit(0);
    } catch (SQLException e) {
        System.err.println("Error database connection: " + e);
        System.exit(0);
     }
  }
}
```

กิจกรรม

- จงสร้าง Servlet ที่ส่ง response ในรูปแบบ HTML ซึ่งภายในมีเนื้อหาที่แสดงข้อมูลทั้งหมดจากฐานข้อมูล blueshop ในตาราง product โดยใช้แท็ก <table> ในการจัดรูปแบบ

Assignment#3

- จากกิจกรรมก่อนหน้านี้ จงสร้างแบบฟอร์มค้นหา ซึ่งอยู่เหนือตารางรายการสินค้า โดยช่องค้นหานั้นจะค้นหาตามชื่อของสินค้า



- จากหน้าเว็บข้างต้นจงสร้าง Link ไปยัง Servlet ใหม่ เพื่อทำหน้าที่ Export ข้อมูลจากผลลัพธ์ในการค้นหาให้อยู่ในรูปแบบไฟล์ product.csv
  - หมายเหตุ การทำให้ผู้ใช้ save เป็นชื่อและนามสกุลที่ต้องการได้นั้นจะต้องกำหนดหลังจาก set content type ดังนี้

```
response.setHeader("Content-Disposition", "attachment;filename='product.csv'");
```

การติดตั้ง MySQL Server

- แตกไฟล์ mysql-xxx-winx64.zip ไว้ที่ Drive C:\

- หลังจากนั้นเปิด Command Line ที่ C:\mysql-xxx-winx64\bin พิมพ์คำสั่งดังนี้

```
mysqld-debug --initialize-insecure --user=mysql
```

- Start MySQL Server ด้วยคำสั่ง

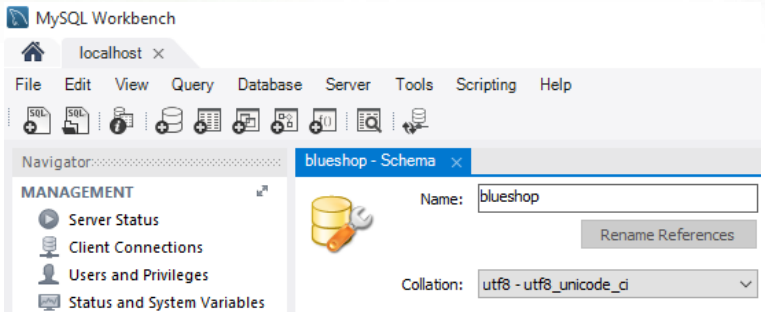
```
mysqld-debug
```

การติดตั้งโปรแกรม MySQL Workbench

- แตกไฟล์ mysql-workbench-community-XXX-winx64-noinstall.zip ใน Drive C:\
- เข้าไปยังโฟลเดอร์ C:\MySQL Workbench XXX CE (winx64)
- เปิดโปรแกรม โดยการ double click ที่ไฟล์ MySQLWorkbench.exe
- หากโปรแกรมไม่ขึ้นให้ติดตั้งโปรแกรมเพิ่มเติม
  - Microsoft .NET Framework 4 Client Profile
  - Visual C++ Redistributable for Visual Studio 2013

สร้างฐานข้อมูลบน MySQL Server

❖ สร้างฐานข้อมูลใหม่ชื่อ blueshop โดยคลิกที่ปุ่ม  หลังจากนั้นคลิกปุ่ม Apply



❖ เลือกฐานข้อมูล blueshop ที่สร้างขึ้นโดย Double click ที่ชื่อ blueshop

❖ เปิดหน้าต่างพิมพ์คำสั่ง SQL โดยคลิกที่ปุ่ม 

```
CREATE TABLE product (
    pid INT NOT NULL AUTO_INCREMENT,
    pname  VARCHAR(50),
    pdetail  VARCHAR(200),
    price INT,
    PRIMARY KEY (pid)
);
```

เพิ่มข้อมูลลงไปในตาราง

❖ ใช้คำสั่ง SQL ดังนี้

```
INSERT INTO product (pname, pdetail, price) VALUES
('Centrum', 'วิตามินรวมจาก A ถึง Zinc', 350),
('Caltrate', 'บำรุงกระดูก เสริมวิตามินดี', 760),
('Ester-C', 'วิตามินซี 500 mg ไม่กัดกระเพาะ', 500),
('Glucosamine', 'บำรุงข้อต่อ ป้องกันข้อเสื่อม', 1200);
```

การ Insert/Update/Delete

- การใช้คำสั่ง SQL ประเภทดึงข้อมูล จะใช้เมธอด executeQuery( ) แต่การใช้คำสั่ง SQL ที่เป็นการ Insert/Update/Delete จะใช้เมธอด executeUpdate( )

คำสั่ง SQL ประเภทดึงข้อมูล

```
ResultSet resultSet = statement.executeQuery("SELECT * FROM product");
```

Object สำหรับเก็บผลลัพธ์ที่ฐานข้อมูลส่งกลับ

String ที่อยู่ในรูปคำสั่ง SQL ประเภทดึงข้อมูล

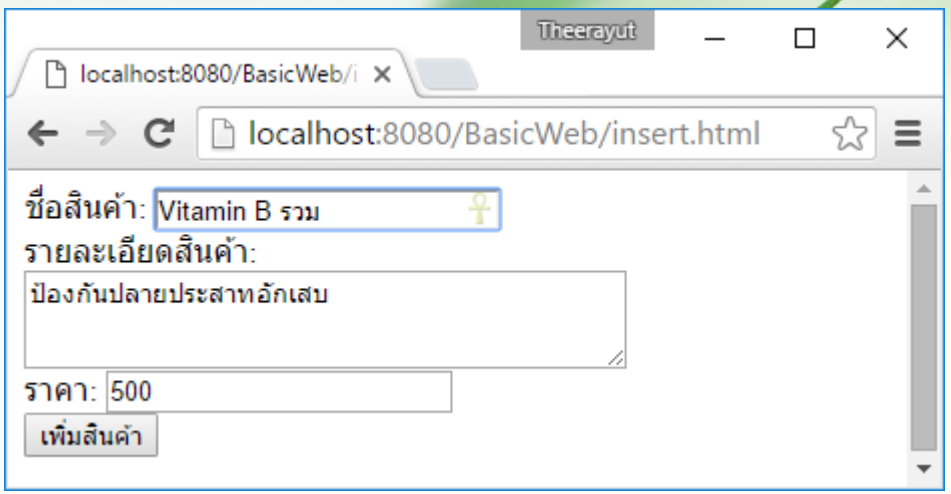
คำสั่ง SQL Insert/Update/Delete

```
int row = statement.executeUpdate("UPDATE product SET price = 1000 WHERE pid = 3");
```

เก็บจำนวนแถวที่มีผลกระทบ

String ที่อยู่ในรูปคำสั่ง SQL ประเภท Insert/Update/Delete

ตัวอย่างแบบฟอร์มเพิ่มข้อมูลสินค้า




```
<html>
<head>
<meta charset="UTF-8">
</head>

<body>

<form action="AddProduct" method="post">
    ชื่อสินค้า: <input type="text" name="pname"><br>
    รายละเอียดสินค้า: <br>
   <textarea name="pdetail" rows="3" cols="40"></textarea><br>
    ราคา: <input type="number" name="price"><br>
   <input type="submit" value="เพิ่มสินค้า">
</form>

</body>
</html>
```

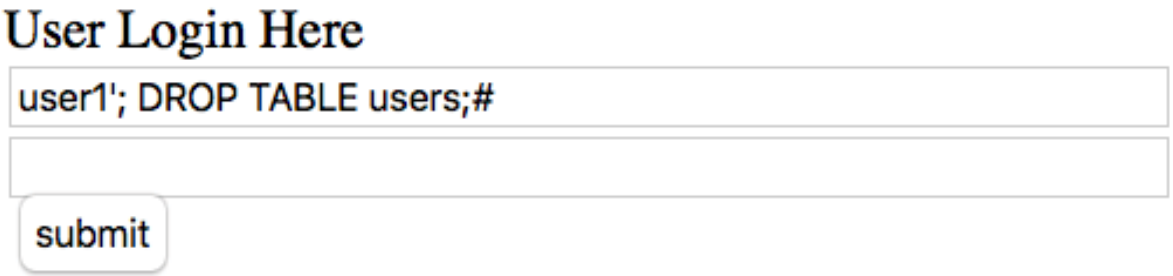
Servlet สำหรับ insert ข้อมูล

```
protected void service(...) throws ServletException, IOException {

response.setCharacterEncoding("utf-8");
response.setContentType("text/html; charset=utf-8");
PrintWriter out = response.getWriter();

try {
 Class.forName("com.mysql.jdbc.Driver");
 String dbURL = "jdbc:mysql://localhost/blueshop?characterEncoding=utf-8";
 Connection con = DriverManager.getConnection(dbURL, "root", "");

 String pname = new String(request.getParameter("pname").getBytes("ISO8859_1"),"utf-8");
 String pdetail = new String(request.getParameter("pdetail").getBytes("ISO8859_1"),"utf-8");
 String price = request.getParameter("price");

 Statement statement = con.createStatement();
 statement.executeUpdate("INSERT INTO product (pname, pdetail, price) VALUES ("
                         + "'" + pname + "', " + "'" + pdetail + "', " + price + ")");
 out.println("เพิ่มสินค้าสำเร็จ");

} catch (...)
}
}
```

การใช้ PreparedStatement

❖ การใช้ Statement Object มีข้อเสียคือ

- หากมีการเรียกเมธอด createStatement( ) ที่อยู่ในรูปแบบเดียวกันจำนวนมาก เช่น การวนลูป insert จะทำให้ประสิทธิภาพการทำงานลดลง
- อาจเกิด SQL Injection ด้วยการส่งคำสั่ง SQL แทรกมากับ input เช่น




```
SELECT * FROM users WHERE username = 'user1'; DROP TABLE
users;#' AND password = ''
```

ที่มา: เว็บไซต์ http://blog.wisered.com, การป้องกัน SQL Injection เว็บไซต์

การใช้ PreparedStatement

❖ การใช้ PreparedStatement ทดแทน Statement จะทำได้ดังนี้

การใช้ Statement Object

```
Statement statement = con.createStatement();
statement.executeUpdate("INSERT INTO product
                    (pname, pdetail, price) VALUES ("
                    + "'" + pname + "', " + "'" + pdetail + "', " + price + ")");
```

การใช้ PreparedStatement Object

```
PreparedStatement pStatement = con.prepareStatement("INSERT INTO product
                                (pname, pdetail, price) VALUES (?, ?, ?)");
pStatement.setString(1, pname);
pStatement.setString(2, pdetail);
pStatement.setInt(3, Integer.parseInt(price));
pStatement.executeUpdate();
```