SQL Injection

Hans-Petter Halvorsen

Contents

- Cyber Security
- SQL Injection
  - Practical Examples
- How to avoid SQL Injection
  - Practical Examples

Cyber Security

- Cyber Security is the practice of protecting systems, networks, and programs from Digital Attacks
- Cyber Security is the strategy for protecting data systems from attacks where the purpose is to stealing money, personal information, Password, etc.

SQL Injection

- SQL injection is a code injection technique that might destroy your database or expose information, such as passwords, etc.
- SQL injection is one of the most common web hacking techniques.
- A Structured Query Language (SQL) injection occurs when an attacker inserts malicious code into a server that uses SQL and forces the server to reveal information it normally would not.
- An attacker could carry out a SQL injection simply by submitting malicious code into a vulnerable website search box.

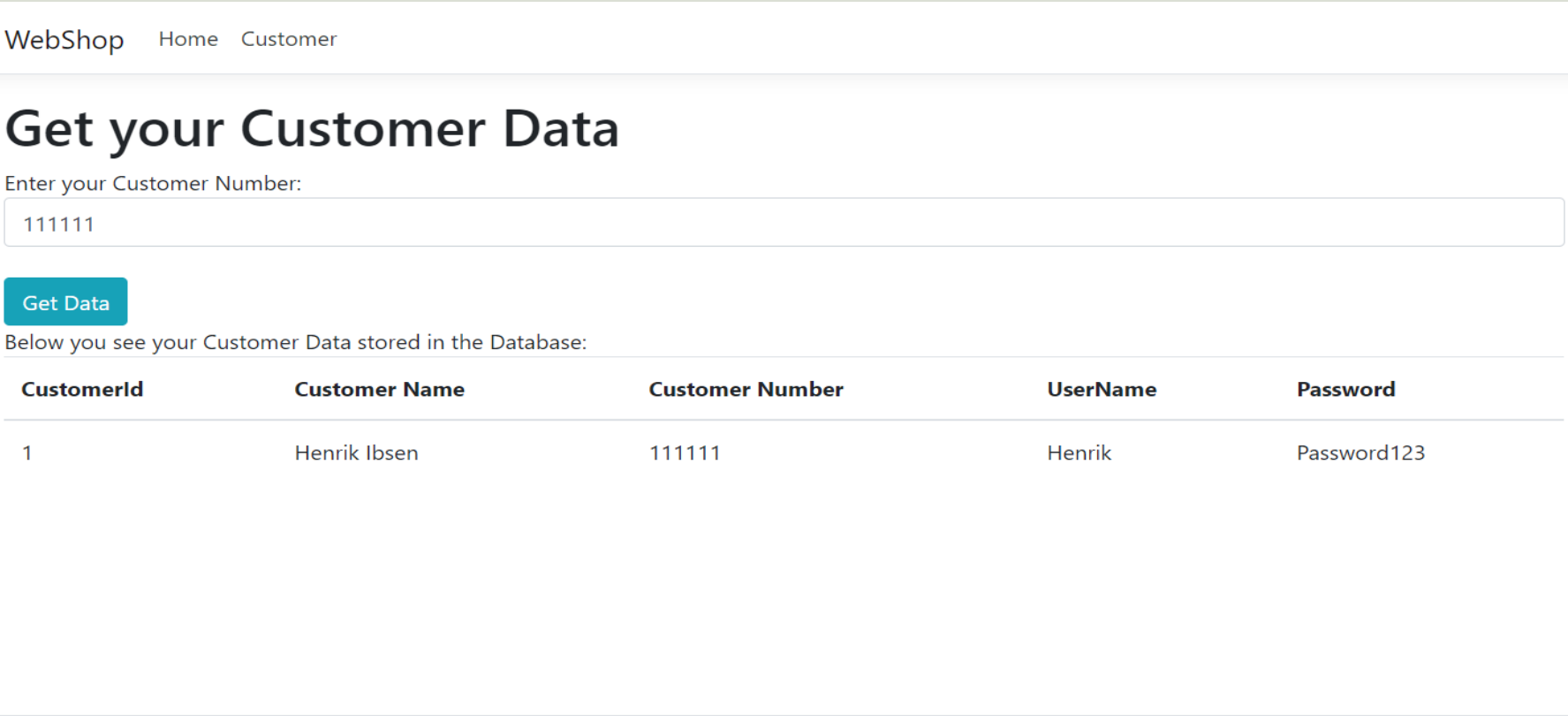
https://www.halvorsen.blog

Practical Examples

ASP.NET Core/C#

Hans-Petter Halvorsen

Web Application Example



C# Examples

Note!

- The Examples are “Dummy” Examples used to prove the concepts of SQL Injection
- The examples provided can be considered as a “proof of concept”
- The sample code is very simplified for clarity and doesn't necessarily represent best practices.

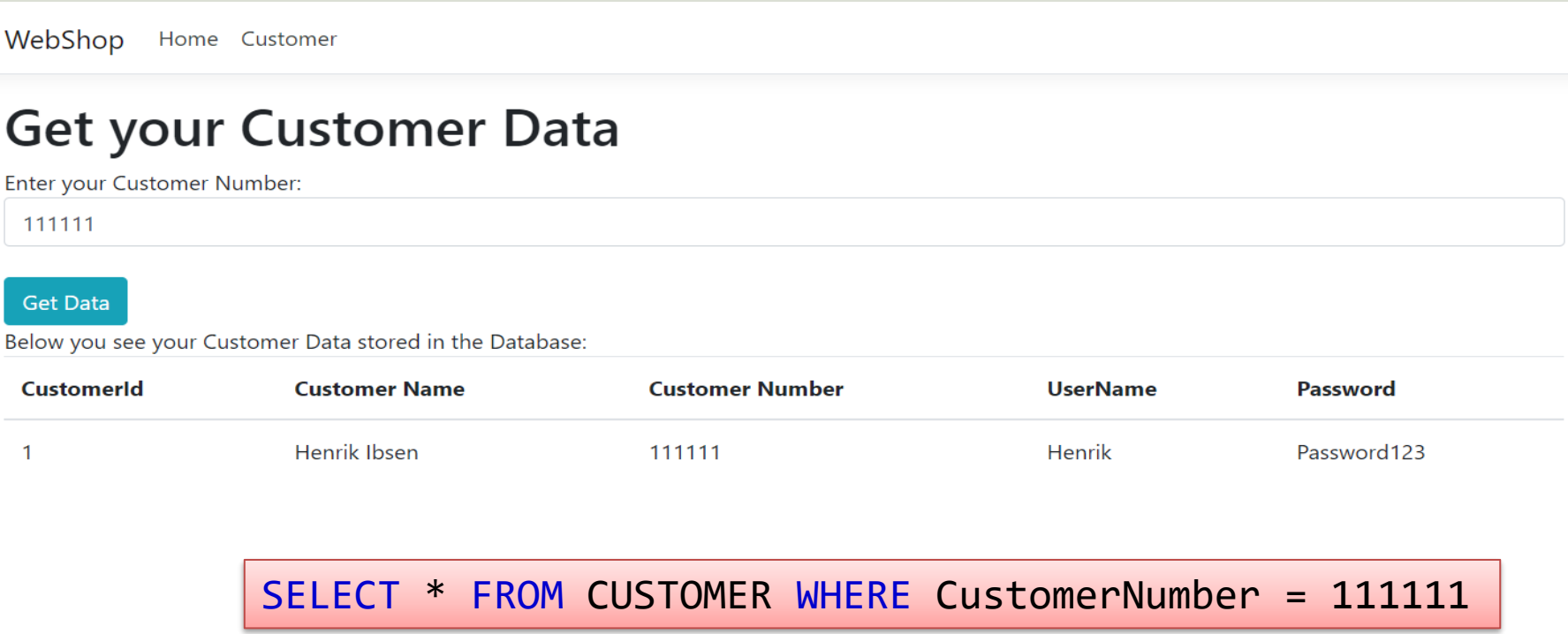
Database

```
CREATE TABLE [CUSTOMER]
(
        [CustomerId] [int] IDENTITY(1, 1) NOT NULL PRIMARY KEY,
        [CustomerName] [varchar](50) NOT NULL UNIQUE,
        [CustomerNumber] [int] NULL,
        [UserName] [varchar](50) NULL,
        [Password] [varchar](50) NULL,
)
GO
```

```
insert into CUSTOMER (CustomerName, CustomerNumber, UserName, Password) values ('Henrik Ibsen', 111111, 'Henrik', 'Password123')
insert into CUSTOMER (CustomerName, CustomerNumber, UserName, Password) values ('Elvis Presly', 222222, 'Elvis', 'Password456')
insert into CUSTOMER (CustomerName, CustomerNumber, UserName, Password) values ('Bob Marley', 333333, 'Bob', 'Password789')
insert into CUSTOMER (CustomerName, CustomerNumber, UserName, Password) values ('Frank  Sinatra', 444444, 'Frank', 'Password101145')
```

Examples #1

Example

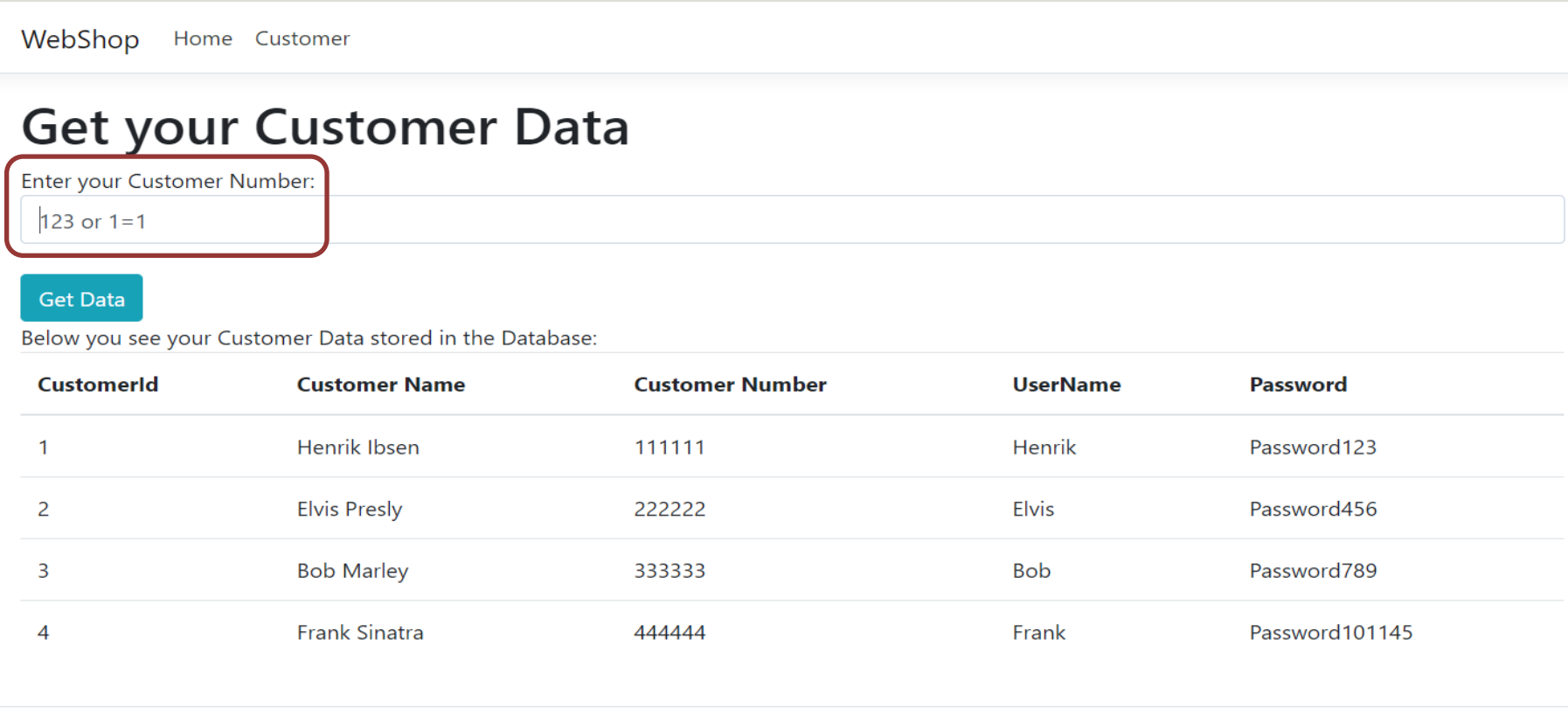

WebShop
Home
Customer
Get your Customer Data
Enter your Customer Number:
111111
Get Data
Below you see your Customer Data stored in the Database:
CustomerId
Customer Name
Customer Number
UserName
Password
1
Henrik Ibsen
111111
Henrik
Password123
SELECT * FROM CUSTOMER WHERE CustomerNumber = 111111
© 2021 - WebShop - Privacy

Code

```
public class Customer
{
    2 references
    public int CustomerId { get; set; }
    2 references
    public string CustomerName { get; set; }
    2 references
    public string CustomerNumber { get; set; }
    2 references
    public string UserName { get; set; }
    2 references
    public string Password { get; set; }

    1 reference
    public List<Customer> GetCustomerData(string customerNumber)
    {
        string connectionString = "DATA SOURCE=LOCALHOST\\SQLEXPRESS;DATABASE=WEBSHOP;Integrated Security = True";

        List<Customer> customerDataList = new List<Customer>();

        SqlConnection con = new SqlConnection(connectionString);

        string sqlQuery = "select * from CUSTOMER where CustomerNumber=" + customerNumber;

        con.Open();

        SqlCommand cmd = new SqlCommand(sqlQuery, con);

        SqlDataReader dr = cmd.ExecuteReader();

        if (dr != null)
        {
            while (dr.Read())
            {
                Customer customerData = new Customer();

                customerData.CustomerId = Convert.ToInt32(dr["CustomerId"]);
                customerData.CustomerName = dr["CustomerName"].ToString();
                customerData.CustomerNumber = dr["CustomerNumber"].ToString();
                customerData.UserName = dr["UserName"].ToString();
                customerData.Password = dr["Password"].ToString();

                customerDataList.Add(customerData);
            }
        }
        return customerDataList;
    }
}
```

```
SELECT * FROM CUSTOMER WHERE CustomerNumber = 111111
```

Example


WebShop
Home
Customer
Get your Customer Data
Enter your Customer Number:
123 or 1=1
Get Data
Below you see your Customer Data stored in the Database:
CustomerId
Customer Name
Customer Number
UserName
Password
1
Henrik Ibsen
111111
Henrik
Password123
2
Elvis Presly
222222
Elvis
Password456
3
Bob Marley
333333
Bob
Password789
4
Frank Sinatra
444444
Frank
Password101145
© 2021 - WebShop - Privacy

Example

"OR 1=1" is always TRUE

WebShop Home Customer

123 or 1=1

Get your Customer Data

Enter your Customer Number:

123 or 1=1

Customer Data for ALL Customers are exposed!!

Get Data

Below you see your Customer Data stored in the Database:

CustomerId	Customer Name	Customer Number	UserName	Password
1	Henrik Ibsen	111111	Henrik	Password123
2	Elvis Presly	222222	Elvis	Password456
3	Bob Marley	333333	Bob	Password789
4	Frank Sinatra	444444	Frank	Password101145

```
SELECT * FROM CUSTOMER WHERE CustomerNumber = 123 OR 1=1
```

Code

```
public class Customer
{
    2 references
    public int CustomerId { get; set; }
    2 references
    public string CustomerName { get; set; }
    2 references
    public string CustomerNumber { get; set; }
    2 references
    public string UserName { get; set; }
    2 references
    public string Password { get; set; }

    1 reference
    public List<Customer> GetCustomerData(string customerNumber)
    {
        string connectionString = "DATA SOURCE=LOCALHOST\\SQLEXPRESS;DATABASE=WEBSHOP;Integrated Security = True";

        List<Customer> customerDataList = new List<Customer>();

        SqlConnection con = new SqlConnection(connectionString);

        string sqlQuery = "select * from CUSTOMER where CustomerNumber=" + customerNumber;

        con.Open();

        SqlCommand cmd = new SqlCommand(sqlQuery, con);

        SqlDataReader dr = cmd.ExecuteReader();

        if (dr != null)
        {
            while (dr.Read())
            {
                Customer customerData = new Customer();

                customerData.CustomerId = Convert.ToInt32(dr["CustomerId"]);
                customerData.CustomerName = dr["CustomerName"].ToString();
                customerData.CustomerNumber = dr["CustomerNumber"].ToString();
                customerData.UserName = dr["UserName"].ToString();
                customerData.Password = dr["Password"].ToString();

                customerDataList.Add(customerData);
            }
        }
        return customerDataList;
    }
}
```

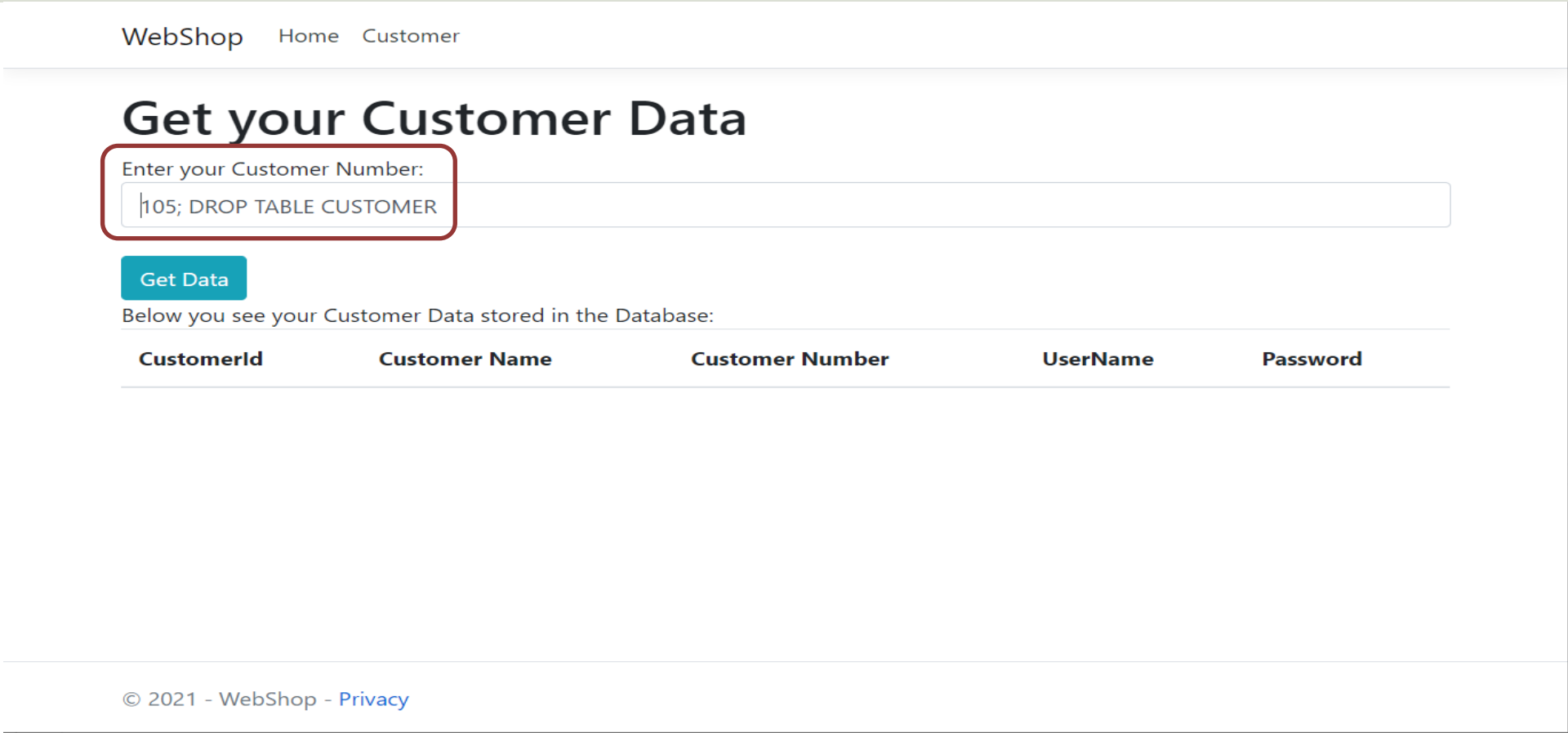
```
SELECT * FROM CUSTOMER WHERE CustomerNumber = 123 OR 1=1
```

The SQL above is valid and will return ALL rows from the "CUSTOMER" table, since **OR 1=1** is always TRUE.

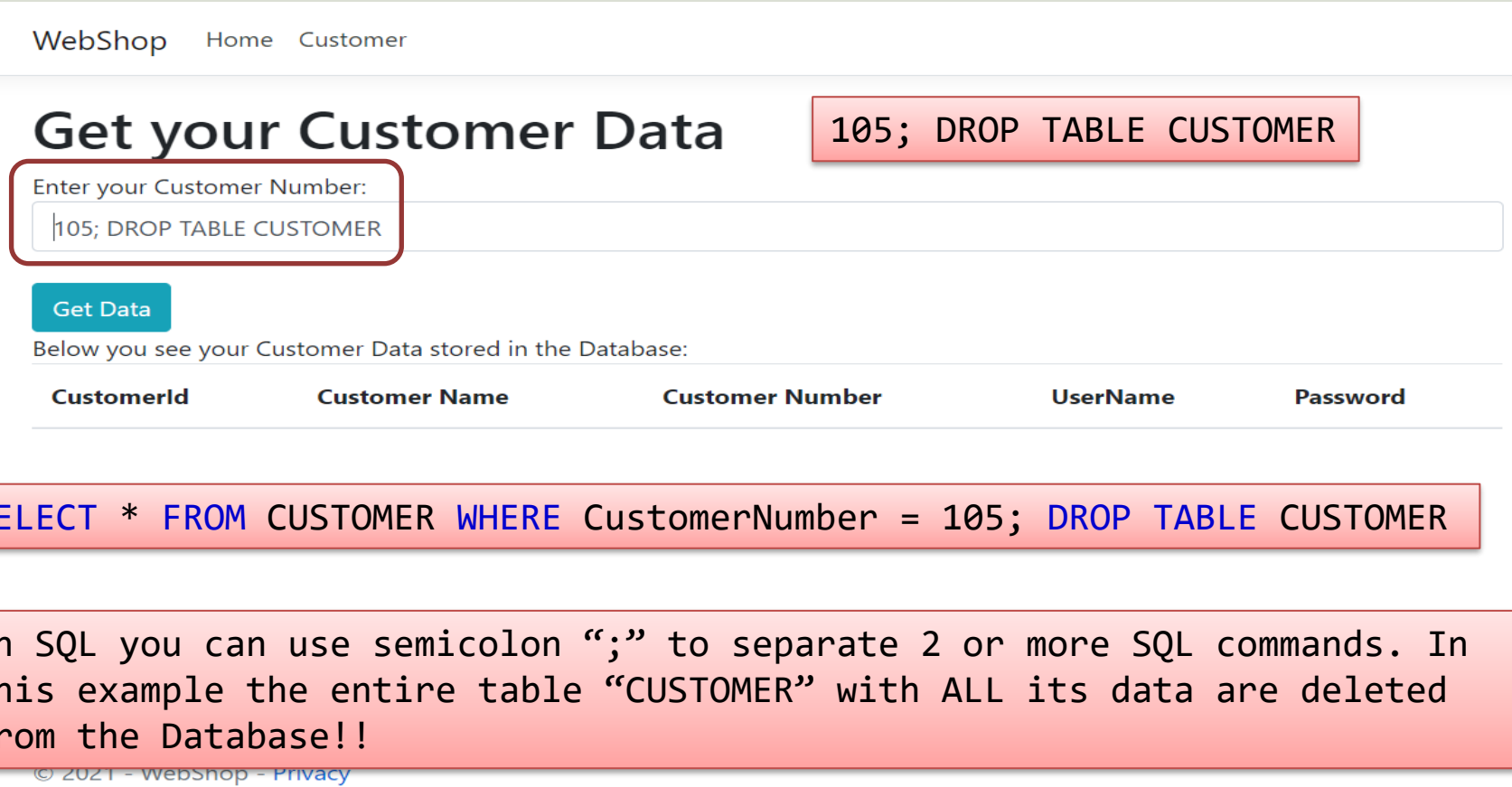
Examples #2

Hans-Petter Halvorsen

Deleting Data!!



Deleting Data!!



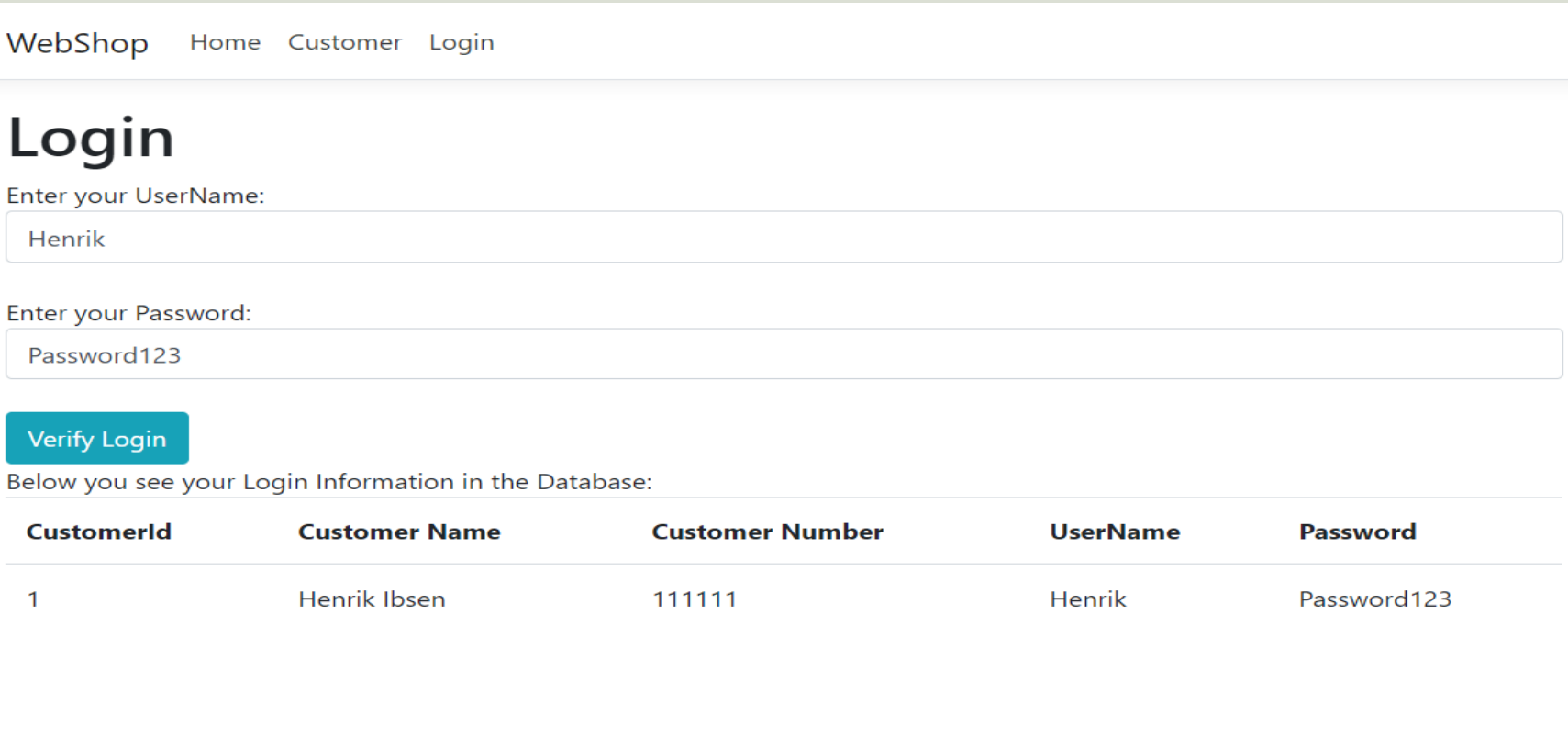

```
SELECT * FROM CUSTOMER WHERE CustomerNumber = 105; DROP TABLE CUSTOMER
```

In SQL you can use semicolon “;” to separate 2 or more SQL commands. In this example the entire table “CUSTOMER” with ALL its data are deleted from the Database!!

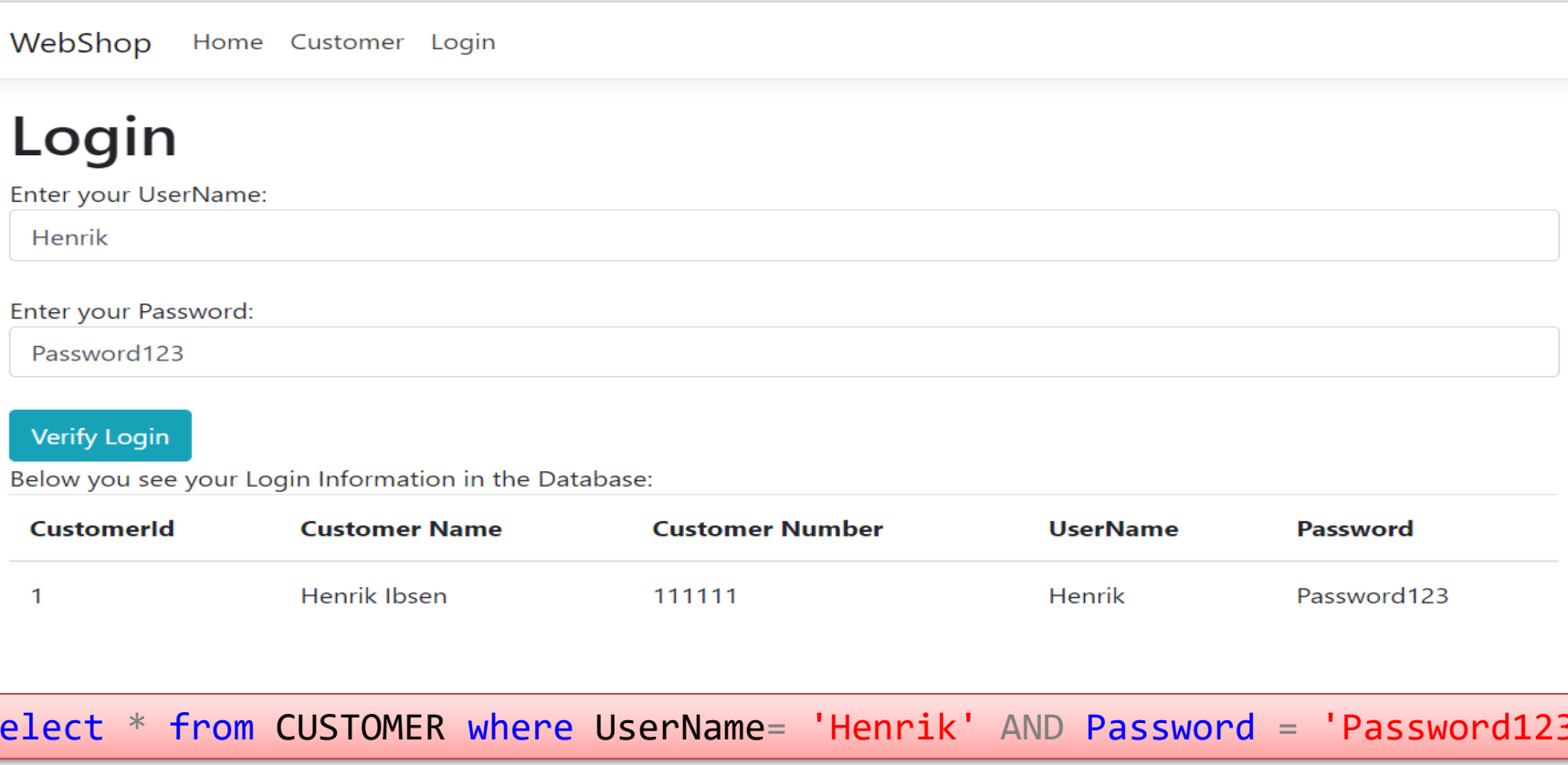
Examples #3

Hans-Petter Halvorsen

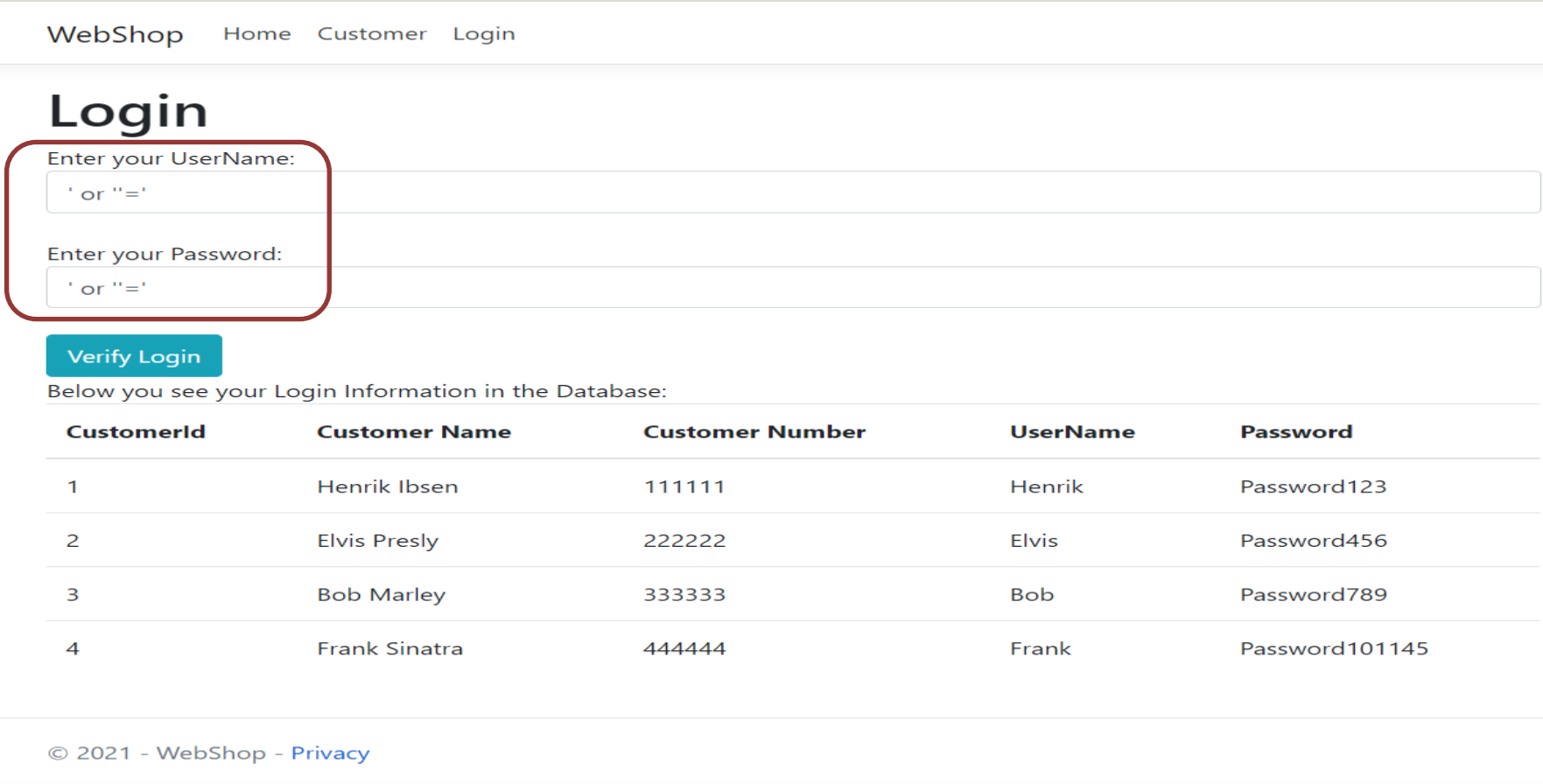
Login Example



Login Example



Login Example



Login Example

"OR ''='' " is always TRUE

WebShop Home Customer Login

Login

Enter your UserName:

' or ''='

or ''='

Enter your Password:

' or ''='

or ''='

Customer Data for ALL Customers are exposed!!

Verify Login

Below you see your Login Information in the Database:

CustomerId	Customer Name	Customer Number	UserName	Password
1	Henrik Ibsen	111111	Henrik	Password123
2	Elvis Presly	222222	Elvis	Password456
3	Bob Marley	333333	Bob	Password789
4	Frank Sinatra	444444	Frank	Password101145

```
select * from CUSTOMER where UserName='' or ''='' AND Password ='' or ''=''
```

https://www.halvorsen.blog

Avoid SQL Injection

Practical ASP.NET Core/C# Examples

Hans-Petter Halvorsen

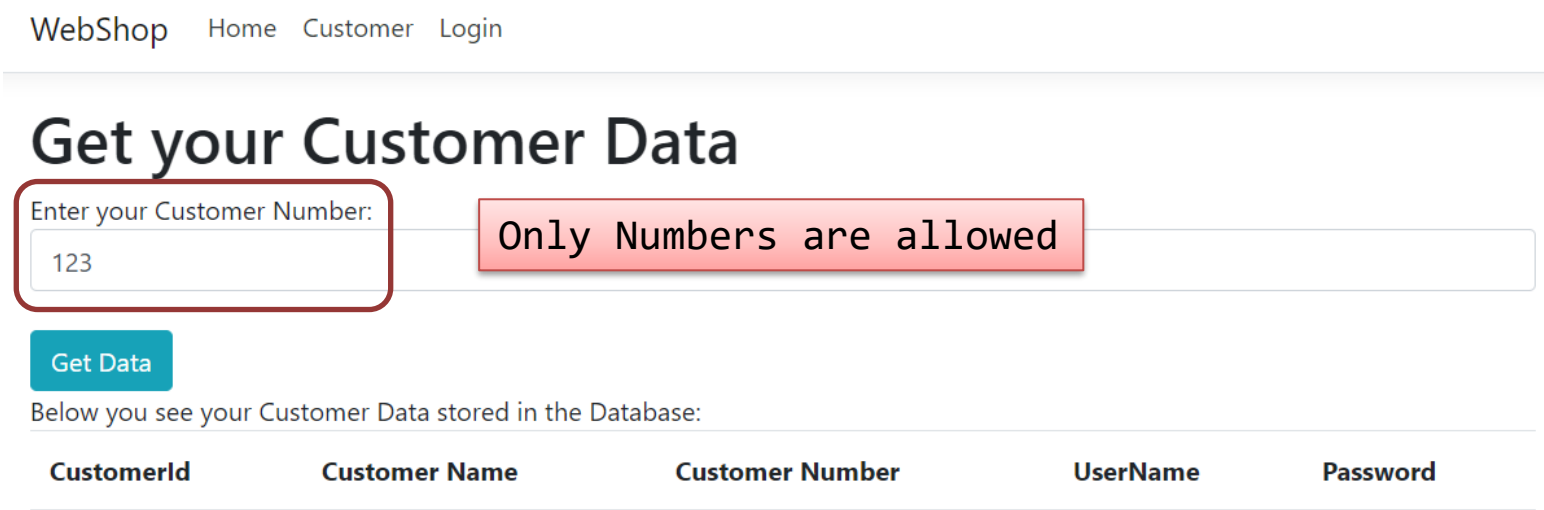
Avoid SQL Injection

- Use proper Data Types in your Code, i.e., int, etc. instead of string for everything
- Check Input in GUI if proper Data Type
- Use SQL Parameters
- ..

Check Input in GUI

```
<input name="CustomerNumber" type="text" value="@Model.customerNumber" />
```

```
<input name="CustomerNumber" type= "number" value="@Model.customerNumber" />
```



Input Types

```
<input type="button">
<input type="checkbox">
<input type="color">
<input type="date">
<input type="datetime-local">
<input type="email">
<input type="file">
<input type="hidden">
<input type="image">
<input type="month">
<input type="number">
<input type="password">
<input type="radio">
<input type="range">
<input type="reset">
<input type="search">
<input type="submit">
<input type="tel">
<input type="text">
<input type="time">
<input type="url">
<input type="week">
```

Based on your choice, the web browser will automatically do proper validation to check in the input from the user is correct

Use proper Data Types in your Code

```
..

namespace WebShop.Pages
{
    public class CustomerModel : PageModel
    {
        public string customerNumber;
        public List<Customer> customerDataList = new List<Customer>();

        public void OnGet()
        {

        }

        public void OnPost()
        {
            Customer customer = new Customer();

            customerNumber = Request.Form["CustomerNumber"];

            customerDataList = customer.GetCustomerData(customerNumber);
        }
    }
}
```

```
public class Customer
    {
        public int CustomerId { get; set; }
        public string CustomerName { get; set; }
        public string CustomerNumber { get; set; }
        public string UserName { get; set; }
        public string Password { get; set; }

        public List<Customer> GetCustomerData(string customerNumber)
        {
            string connectionString = "DATA SOURCE=LOCALHOST\\SQLEXPRESS;DATABASE=WEBSHOP;Integrated Security = True";
            List<Customer> customerDataList = new List<Customer>();
            SqlConnection con = new SqlConnection(connectionString);

            string sqlQuery = "select * from CUSTOMER where CustomerNumber=" + customerNumber;

            con.Open();

            SqlCommand cmd = new SqlCommand(sqlQuery, con);

            SqlDataReader dr = cmd.ExecuteReader();

            if (dr != null)
            {
                while (dr.Read())
                {
                    Customer customerData = new Customer();

                    customerData.CustomerId = Convert.ToInt32(dr["CustomerId"]);
                    customerData.CustomerName = dr["CustomerName"].ToString();
                    customerData.CustomerNumber = dr["CustomerNumber"].ToString();
                    customerData.UserName = dr["UserName"].ToString();
                    customerData.Password = dr["Password"].ToString();

                    customerDataList.Add(customerData);
                }
            }
            return customerDataList;
        }
..
```

Use SQL Parameters

SQL parameters are values that are added to an SQL query at execution time, in a controlled manner

```
sqlQuery = "select * from CUSTOMER where CustomerNumber = @customernumber";

customerNumberParameter = new SqlParameter("customernumber", SqlDbType.Int);
customerNumberParameter.Value = customerNumber;

cmd.Parameters.Add(customerNumberParameter);
```

Without SQL Parameters

```
..

SqlConnection con = new SqlConnection(connectionString);

string sqlQuery = "select * from CUSTOMER where CustomerNumber=" + customerNumber;

con.Open();

SqlCommand cmd = new SqlCommand(sqlQuery, con);

SqlDataReader dr = cmd.ExecuteReader();

..
```

Use SQL Parameters

```
..
SqlConnection con = new SqlConnection(connectionString);

string sqlQuery = "select * from CUSTOMER where CustomerNumber = @customernumber";

con.Open();

SqlCommand cmd = new SqlCommand(sqlQuery, con);

var customerNumberParameter = new SqlParameter("customernumber", SqlDbType.Int);
customerNumberParameter.Value = customerNumber;

cmd.Parameters.Add(customerNumberParameter);

SqlDataReader dr = cmd.ExecuteReader();

..
```

Use SQL Parameters

- Using SQL Parameters are safer than putting the values into the string because the parameters are passed to the database separately, protecting against SQL injection attacks.
- It is also be more efficient if you execute the same SQL repeatedly with different parameters.
- The Example was showing an ASP.NET Core using C#
- Other Languages like PHP, Python, etc. offer the same functionality

SQL Parameters - Python Example

Example:

```
import pyodbc
import database

connectionString = database.GetConnectionString()

conn = pyodbc.connect(connectionString)

cursor = conn.cursor()

query = "select BookId, Title, Author, Category from BOOK where Category=?"

parameters = ['Data']

for row in cursor.execute(query, parameters):
    print(row.BookId, row.Title, row.Author, row.Category)
```

SQL Parameters - PHP Example

Example:

```
..

$sql = $conn->prepare("CALL CreateWorkGroup (?, ?, ?, ?)");

$sql->bind_param("iiss", $workgroupid, $personid, $workgroupname, $workgroupicon);

$sql->execute();

..
```

References

- W3school: https://www.w3schools.com/sql/sql_injection.asp
- Stack Overflow: https://stackoverflow.com/questions/7505808/why-do-we-always-prefer-using-parameters-in-sql-statements

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog