♠ > La meua pàgina inicial > Els meus cursos > SEG9_CEE_46025799_2022_836104_0485 > UF13.- ACCÉS A BASES DE DADES > Java PreparedStatement

 Mostrar blocks

Vista estàndard

Java PreparedStatement

PreparedStatement

A La meua pàgina inicial

A Java, "PreparedStatement" és una interfície que s'utilitza per a precompilar una consulta SQL que s'executarà diverses vegades, la qual cosa pot millorar significativament el rendiment de l'aplicació. El PreparedStatement es crea a partir d'un objecte Connection i s'utilitza per a assignar valors als paràmetres de la consulta abans d'executar-la.

Avantatges de PreparedStatement

L'avantatge principal d'usar PreparedStatement a Java és que la consulta SQL es precompila abans de ser executada, la qual cosa pot millorar significativament el rendiment de l'aplicació, especialment quan s'executa la mateixa consulta diverses vegades amb diferents valors de paràmetres.

A més, en utilitzar PreparedStatement, es poden evitar els problemes de seguretat relacionats amb la injecció de SQL, ja que els valors dels paràmetres s'assignen de manera segura i es validen automàticament. Això fa que l'aplicació siga més segura i menys vulnerable a atacs maliciosos.

Un altre avantatge de PreparedStatement és que permet als desenvolupadors escriure codi més net i llegible, ja que no és necessari concatenar manualment les cadenes de consulta SQL i els valors dels paràmetres.

La inyecció SQL

La injecció SQL és un tipus d'atac informàtic que es produeix quan un atacant inserida codi SQL maliciós en una consulta, amb l'objectiu d'obtindre informació o prendre el control de la base de dades. A continuació es mostra un exemple simple d'injecció SQL:

SELECT * FROM usuarios WHERE usuario = '\$usuario' AND contrasena = '\$contrasena'

Que introdueix l'atacant al formulari de login com a camp usuari

El resultat de la consulta

import java.sql.*;

Aquesta consulta sempre retornarà totes les files de la taula "usuaris", ja que la condició "OR 1=1" sempre serà vertadera. El comentari "--" s'utilitza per a evitar que la resta de la consulta s'execute, de manera que l'atacant puga obtindre accés a la base de dades sense necessitat de conéixer la contrasenya.

Aquest és només un exemple molt simple d'injecció SQL, però hi ha moltes altres formes més sofisticades en les quals els atacants poden explotar vulnerabilitats en les aplicacions per a accedir a la base de dades.

Exemple de PreparedStatement

En aquest exemple, hem utilitzat PreparedStatement per a preparar una consulta parametritzada que selecciona les files de la taula "productos" que tenen un valor determinat en la columna 1 i la columna 2. Hem establit els valors dels paràmetres utilitzant els mètodes "setString" del PreparedStatement i després hem executat la consulta utilitzant el mètode "executeQuery". Com en l'exemple anterior, hem tancat els objectes "ResultSet", "PreparedStatement" i "Connection" manualment en el bloc "finally".

```
public class DatabaseConnector {
      public static void main(String[] args) {
         String url = "jdbc:mysql://localhost:3306/uf13";
         String user = "root";
         String password = "";
         Connection connection = null;
         PreparedStatement preparedStatement = null;
 9
         ResultSet resultSet = null;
10
         try {
            connection = DriverManager.getConnection(url, user, password);
13
            String query = "SELECT * FROM productos WHERE column1 = ? AND column2 = ?";
14
            preparedStatement = connection.prepareStatement(query);
15
16
            preparedStatement.setString(1, "value1");
            preparedStatement.setString(2, "value2");
17
            resultSet = preparedStatement.executeQuery();
18
19
            while (resultSet.next()) {
20
               System.out.println(resultSet.getString("column1") + " " + resultSet.getString("column2"));
21
22
23
         } catch (SQLException e) {
24
25
            e.printStackTrace();
26
         } finally {
27
            // tanquem manualment els objectes en ordre invers a com es van crear
28
            if (resultSet != null) {
               try {
29
                  resultSet.close();
30
               } catch (SQLException e) {
31
32
                  e.printStackTrace();
33
34
            }
35
            if (preparedStatement != null) {
36
37
               try {
38
                  preparedStatement.close();
               } catch (SQLException e) {
39
                  e.printStackTrace();
40
41
42
43
            if (connection != null) {
44
45
                  connection.close();
46
               } catch (SQLException e) {
                  e.printStackTrace();
48
50
51
52
53 }
```

Darrera modificació: dilluns, 24 abril 2023, 22:13

PREVIOUS ACTIVITY

Java AutoClose

Salta a...

NEXT ACTIVITY UF13.-Exercicis

© 2020 Generalitat

Conselleria d'Educació, Cultura i Esport

Avís legal | Contacte

UNIÓ EUROPEA