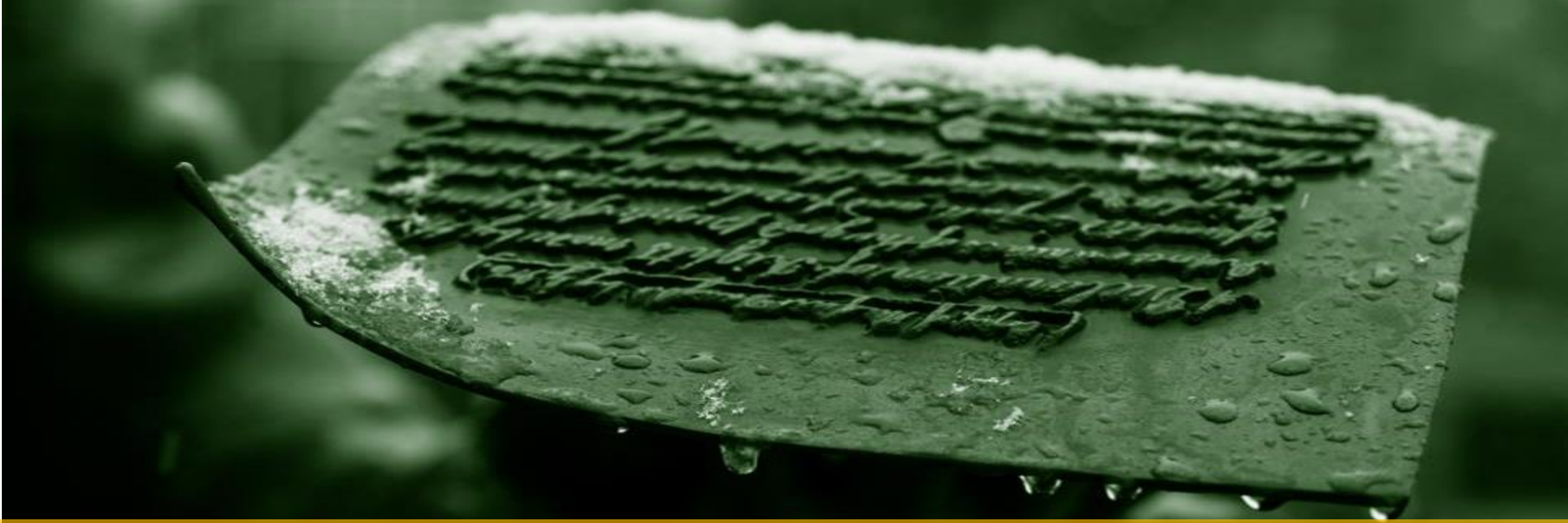

MODULE 11 – CALLING STORED PROCEDURES FROM NODEJS SCRIPTS

IT 207 – IT Programming

GEORGE MASON UNIVERSITY

Lecture Outline

- Pros and Cons for using Stored Procedures in Nodejs
- Helpful Stored Procedures commands on MYSQL server
- Calling Stored Procedures in Nodejs
  - Displaying stored procedure output
  - Passing IN parameters to stored procedures
  - Receiving values from OUT parameters

Stored Procedures Pros & Cons - Pros

- Using stored procedures in the server back-end has several advantages:

1. Improved Performance: stored procedures are cached on the MySQL server, so query execution will be faster.
2. Modularity: queries can be maintained and used as module function calls.
3. Isolation of Business logic: All queries are grouped in one place and can be updated and tested independently.
4. Localization of change: When the database schema changes, only the stored procedures need to be changed
5. Security: Stored procedures provides a level of indirection that shields the implementation and data from attacks

Stored Procedures Pros & Cons - Cons

- Drawbacks of using stored procedures include
  1. Represents another level of a deployment layer that needs to be maintained and updated
  2. Maintenance of stored procedures in big enterprise projects is challenging

HELPFUL SP COMMANDS ON MYSQL SERVER

LISTING STORED PROCEDURES

```
SHOW PROCEDURE STATUS [LIKE 'pattern' | WHERE search_condition]
```

- The SHOW PROCEDURE STATUS statement shows all characteristics of stored procedures including stored procedure names.
- Examples: Using the classicmodels database

1. Listing all procedures in the classicmodels database;

```
SHOW PROCEDURE STATUS
WHERE db = 'classicmodels';
```

```
mysql> SHOW PROCEDURE STATUS WHERE db = 'classicmodels';
+--------------+----------------------+-----------+----------------+
| Db           | Name                 | Type      | Definer        |
+--------------+----------------------+-----------+----------------+
| classicmodels | ALLemployees        | PROCEDURE | root@localhost |
| classicmodels | countEmployee       | PROCEDURE | root@localhost |
| classicmodels | countEmployees      | PROCEDURE | root@localhost |
| classicmodels | GetDiscount         | PROCEDURE | root@localhost |
| classicmodels | GetOfficeByCountry  | PROCEDURE | root@localhost |
| classicmodels | GetOrderCountByStatus | PROCEDURE | root@localhost |
| classicmodels | getProductCodeName  | PROCEDURE | root@localhost |
| classicmodels | GetTotalOrder       | PROCEDURE | root@localhost |
| classicmodels | SetCounter          | PROCEDURE | root@localhost |
| classicmodels | USemployees         | PROCEDURE | root@localhost |
+--------------+----------------------+-----------+----------------+
```

LISTING STORED PROCEDURES – ORDER CLAUSE

```
SHOW PROCEDURE STATUS [LIKE 'pattern' | WHERE search_condition]
```

- The SHOW PROCEDURE STATUS statement shows all characteristics of stored procedures including stored procedure names.
- Examples: Using the classicmodels database

2. Listing all procedures in the classicmodels database whose names contain a specific word;

```
SHOW PROCEDURE STATUS LIKE '%Order%';
```

```
mysql> SHOW PROCEDURE STATUS LIKE '%Order%';
+---------------+----------------------+-----------+----------------+
| Db            | Name                 | Type      | Definer        |
+---------------+----------------------+-----------+----------------+
| classicmodels | GetOrderCountByStatus | PROCEDURE | root@localhost |
| classicmodels | GetTotalOrder        | PROCEDURE | root@localhost |
+---------------+----------------------+-----------+----------------+
```

LISTING STORED PROCEDURES USING THE DATA DICTIONARY - 1

- The `routines` table in the `information_schema` database contains all information on the stored procedures and stored functions of all databases in the current MySQL server.
- To show all stored procedures of a particular database, you use the following query:

```
SELECT
    routine_name
FROM
    information_schema.routines
WHERE
    routine_type = 'PROCEDURE'
        AND routine_schema = '<database_name>';
```

LISTING STORED PROCEDURES USING THE DATA DICTIONARY - 2

- The `routines` table in the `information_schema` database contains all information on the stored procedures and stored functions of all databases in the current MySQL server.
- Example: Using the classicmodels database

```
mysql> SELECT
    ->     routine_name
    -> FROM
    ->     information_schema.routines
    -> WHERE
    ->     routine_type = 'PROCEDURE'
    ->         AND routine_schema = 'classicmodels';
```

```
+----------------------+
| ROUTINE_NAME         |
+----------------------+
| ALLemployees         |
| countEmployee        |
| countEmployees       |
| GetDiscount          |
| GetOfficeByCountry   |
| GetOrderCountByStatus|
| getProductCodeName   |
| GetTotalOrder        |
| SetCounter           |
| USemployees          |
+----------------------+
```

LIST STORED PROCEDURE PARAMETERS

- To list information about a specific procedure, use the following query:

```
SELECT *
FROM information_schema.parameters
WHERE SPECIFIC_NAME = '<your_procedure>';
```

- Examples: Using the classicmodels database

1. Displaying all columns

```
mysql> SELECT *  FROM information_schema.parameters  WHERE SPECIFIC_NAME = 'GetOfficeByCountry';
+------------------+-----------------+--------------------+------------------+----------------+---------------+-----------+
| SPECIFIC_CATALOG | SPECIFIC_SCHEMA | SPECIFIC_NAME      | ORDINAL_POSITION | PARAMETER_MODE | PARAMETER_NAME | DATA_TYPE |
+------------------+-----------------+--------------------+------------------+----------------+---------------+-----------+
| def              | classicmodels   | GetOfficeByCountry |                1 | IN             | countryName   | varchar   |
+------------------+-----------------+--------------------+------------------+----------------+---------------+-----------+
```

2. Selecting specific columns

```
mysql> SELECT PARAMETER_MODE, PARAMETER_NAME, DATA_TYPE, CHARACTER_MAXIMUM_
ECIFIC_NAME = 'GetOfficeByCountry';
+----------------+----------------+-----------+--------------------------+
| PARAMETER_MODE | PARAMETER_NAME | DATA_TYPE | CHARACTER_MAXIMUM_LENGTH |
+----------------+----------------+-----------+--------------------------+
| IN             | countryName    | varchar   |                      255 |
+----------------+----------------+-----------+--------------------------+
```

SHOW STORED PROCEDURE STATEMENTS

- To show the header and body of a stored procedure , use the following query:

```
SHOW CREATE PROCEDURE <your_procedure>;
```

- Example: Using the classicmodels database

```
mysql> SHOW CREATE PROCEDURE countEmployee;
+---------------+-----------------------------------------------------------------------------------------------------------------
---------+-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------+----------------------+----------
--------+--------------------+
| Procedure     | sql_mode
         | Create Procedure
                                                                                                                  | character_set_client | collation_
nection | Database Collation |
+---------------+-----------------------------------------------------------------------------------------------------------------
---------+-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------+----------------------+----------
--------+--------------------+
| countEmployee | ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_S
TITUTION | CREATE DEFINER=`root`@`localhost` PROCEDURE `countEmployee`(IN c VARCHAR(255), OUT empCount INT)
BEGIN SELECT COUNT(e.employeeNumber) INTO empCount  FROM employees as e, offices as o WHERE o.city = c; END | utf8mb4
    | utf8mb4_0900_ai_ci   | latin1_swedish_ci  |
```

The procedure statements are marked

CALLING STORED PROCEDURES IN NODEJS

DISPLAYING STORED PROCEDURE OUTPUT

- The stored procedure Usemployees() displays all employees that are in the US.
- The procedure does not take any parameter or return any values.
- The procedure outputs the list to the MysQL server command line

```
mysql> call USemployees();
+-----------+-----------+-----------------+-----------+---------------+-------+
| lastName  | firstName | phone           | extension | city          | state |
+-----------+-----------+-----------------+-----------+---------------+-------+
| Murphy    | Diane     | +1 212 555 3000 | x5800     | NYC           | NY    |
| Murphy    | Diane     | +1 215 837 0825 | x5800     | Boston        | MA    |
| Murphy    | Diane     | +1 650 219 4782 | x5800     | San Francisco | CA    |
| Patterson | Mary      | +1 212 555 3000 | x4611     | NYC           | NY    |
| Patterson | Mary      | +1 215 837 0825 | x4611     | Boston        | MA    |
| Patterson | Mary      | +1 650 219 4782 | x4611     | San Francisco | CA    |
| Firrelli  | Jeff      | +1 212 555 3000 | x9273     | NYC           | NY    |
| Firrelli  | Jeff      | +1 215 837 0825 | x9273     | Boston        | MA    |
| Firrelli  | Jeff      | +1 650 219 4782 | x9273     | San Francisco | CA    |
| Patterson | William   | +1 212 555 3000 | x4871     | NYC           | NY    |
| Patterson | William   | +1 215 837 0825 | x4871     | Boston        | MA    |
| Patterson | William   | +1 650 219 4782 | x4871     | San Francisco | CA    |
| Bondur    | Gerard    | +1 212 555 3000 | x5408     | NYC           | NY    |
| Bondur    | Gerard    | +1 215 837 0825 | x5408     | Boston        | MA    |
| Bondur    | Gerard    | +1 650 219 4782 | x5408     | San Francisco | CA    |
| Bow       | Anthony   | +1 212 555 3000 | x5428     | NYC           | NY    |
| Bow       | Anthony   | +1 215 837 0825 | x5428     | Boston        | MA    |
| Bow       | Anthony   | +1 650 219 4782 | x5428     | San Francisco | CA    |
```

DISPLAYING STORED PROCEDURE OUTPUT IN NODEJS

- In Nodejs script the procedure call will be sent to the MySQL server using the query method defined on the connection object

```
let sql = 'call USemployees()';
db.query(sql, (err, results, fields)=>{
if (err)throw err;
console.log(results);
console.log(fields);
});
```

  - results is an array that will hold the following
    - At index 0: the output
    - At index 1: the MySQL server messages
  - fields is an array that will provide further information for whatever is returned in results

RESULTS & FIELDS PARAMETERS

results[0]: SP output

```
[
 [
  {
   lastName: 'Murphy',
   firstName: 'Diane',
   phone: '+1 212 555 3000',
   extension: 'x5800',
   city: 'NYC',
   state: 'NY'
  },
  {
   lastName: 'Murphy',
   firstName: 'Diane',
   phone: '+1 215 837 0825',
   extension: 'x5800',
   city: 'Boston',
   state: 'MA'
  },
  :
  :
```

results[1]:MySQL Message

```
ResultSetHeader {
   fieldCount: 0,
   affectedRows: 0,
   insertId: 0,
   info: '',
   serverStatus: 34,
   warningStatus: 0,
   changedRows: 0
 }
```

fields[0]:Extra information about result[0]

fields: Extra information about results

```
[
 [
  `lastName` VARCHAR(50) NOT NULL,
  `firstName` VARCHAR(50) NOT NULL,
  `phone` VARCHAR(50) NOT NULL,
  `extension` VARCHAR(10) NOT NULL,
  `city` VARCHAR(50) NOT NULL,
  `state` VARCHAR(50)
 ],
 undefined
]
```

fields[1]:There is no extra information for result[1]

PASSING IN PARAMETERS TO STORED PROCEDURES

- IN parameters are read from the client side and will be passed to store procedures

```
let sql = `call GetOfficeByCountry(?)`;
let values = ["USA"];
db.query(sql,values,(err, results, fields)=>{
if (err) throw err;
console.log("Results[0]: ");
console.log(results[0]) ;
console.log("Results[1]: ");
console.log(results[1]);
console.log("Fields[0]: ");
console.log(fields[0]);
});
```

RESULTS & FIELDS PARAMETERS – IN PARAMETERS

results[0]: SP output

```
Results[0]:
[
 {
  officeCode: '1',
  city: 'San Francisco',
  phone: '+1 650 219 4782',
  addressLine1: '100 Market Street',
  addressLine2: 'Suite 300',
  state: 'CA',
  country: 'USA',
  postalCode: '94080',
  territory: 'NA'
 },
 {
  officeCode: '2',
  city: 'Boston',
  phone: '+1 215 837 0825',
  addressLine1: '1550 Court Place',
  addressLine2: 'Suite 102',
  state: 'MA',
  country: 'USA',
  postalCode: '02107',
  territory: 'NA'
 },
  :
  :
```

results[1]:MySQL Message

```
Results[1]:
ResultSetHeader {
  fieldCount: 0,
  affectedRows: 0,
  insertId: 0,
  info: '',
  serverStatus: 34,
  warningStatus: 0,
  changedRows: 0
}
```

fields: Extra information about results

```
Fields[0]:
[
  `officeCode` VARCHAR(10) NOT NULL PRIMARY KEY,
  `city` VARCHAR(50) NOT NULL,
  `phone` VARCHAR(50) NOT NULL,
  `addressLine1` VARCHAR(50) NOT NULL,
  `addressLine2` VARCHAR(50),
  `state` VARCHAR(50),
  `country` VARCHAR(50) NOT NULL,
  `postalCode` VARCHAR(15) NOT NULL,
  `territory` VARCHAR(10) NOT NULL
]
```

RECEIVING VALUES FROM OUT PARAMETERS

- Receiving values from OUT parameters in stored procedures is done on two steps
  1. Receiving the value of the OUT parameter as a session variable.
  2. Displaying the session variable

```
mysql> call countEmployees('NYC',@empCount);
Query OK, 1 row affected (0.01 sec)

mysql> select @empCount;
+-----------+
| @empCount |
+-----------+
|         2 |
+-----------+
1 row in set (0.00 sec)
```

Session variable

Receiving values from OUT Parameters in Nodjs

- To receive values from OUT parameters in Nodejs requires submitting multiple SQL statements in the query() method
  - Set multipleStatements option to true in the connection object

```
multipleStatements: true,
```

```
let sql = 'call countEmployees(?,@empCount); Select @empCount';
values = ['NYC' ];
db.query(sql, values,(err, results,fields)=>{
if(err) throw err;
console.log("Results[0]");
console.log(results[0]);
console.log("Results[1]");
console.log(results[1]);
});
```

Will hold the results for both queries

  - Note that there is not output for the stored procedure
  - Two queries are submitted, yet one results parameter is used

RESULTS & FIELDS PARAMETERS - OUT PARAMETERS

- results[0] will hold the values returned from the MySQL server
- results[1] will hold the output of the second query
- fields [0] will be undefined, while fields[1] will hold extra information about the value returned in results[1]

```
Results[0]
ResultSetHeader {
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  info: '',
  serverStatus: 42,
  warningStatus: 0,
  changedRows: 0
}
```

```
Results[1]
[ { '@empCount': 2 } ]
```

```
Fields[0]:
undefined
Fields[1]:
[ `@empCount` BIGINT(21) ]
```

Accessing the Value for the @session Key

- Use the bracket notation to access the value for the session key

```
console.log("Results[1][0]['@empCount']");
console.log(results[1][0]['@empCount']);
```

```
Results[1][0]['@empCount']
2
```

SUMMARY

- Using stored procedures in the back-end has several advantages
- Stored procedures can be called from Nodejs scripts using the query() method
- For receiving the value of an OUT parameter in Nodejs scripts, multipleStatements option need to be set to true
- The bracket notation can be used to access the value for the session variable in the returned object in the results parameter of the query() method