queryForObject() throws EmptyResultDataAccessException when record not found

By mkyong | April 20, 2014 | Updated : April 30, 2014 | Viewed : 97,755 | +457 pv/w

Reviewing a legacy project, and found this Spring JDBC code snippets :

```
public User getUser(String username) {

  String sql = "SELECT * FROM USER WHERE username = ?";

  return getJdbcTemplate().queryForObject(
              sql,
              new Object[] { username },
      new RowMapper<UserAttempts>() {
      public UserAttempts mapRow(ResultSet rs, int rowNum) throws SQLException {

          User user = new User();
          user.setId(rs.getInt("id"));
          user.setUsername(rs.getString("username"));
          user.setAge(rs.getInt("age"));
          user.setLastModified(rs.getDate("lastModified"));

          return user;
      }

  });

}
```

Problem

The developer assumes it will return a null when record not found.

```
User user = abc.getUser("mkyong");
if(user == null){
    //...do something
}
```

The problem is, Spring throws an `EmptyResultDataAccessException`, instead of returning a null when record not found.

JdbcTemplate .java

```
package org.springframework.jdbc.core;

public class JdbcTemplate extends JdbcAccessor implements JdbcOperations {

    //...
   public <T> T queryForObject(String sql, Object[] args,
       RowMapper<T> rowMapper) throws DataAccessException {
   List<T> results = query(sql, args, new RowMapperResultSetExtractor<T>(rowMapper, 1));
   return DataAccessUtils.requiredSingleResult(results);
   }
```

DataAccessUtils.java

```
package org.springframework.dao.support;

public abstract class DataAccessUtils {

    //...
    public static <T> T requiredSingleResult(Collection<T> results)
         throws IncorrectResultSizeDataAccessException {
    int size = (results != null ? results.size() : 0);
    if (size == 0) {
        throw new EmptyResultDataAccessException(1);
    }
    if (results.size() > 1) {
        throw new IncorrectResultSizeDataAccessException(1, size);
    }
    return results.iterator().next();
    }
```

P.S Spring version 3.2.8.RELEASE

Solution

Returning null is pretty standard, wonder why Spring wants to throw an `EmptyResultDataAccessException`? To fix it, just catch the exception and return null.

```
  public User getUser(String username) {

    String sql = "SELECT * FROM USER WHERE username = ?";

    try {
          User user = getJdbcTemplate().queryForObject(
                 sql,
                 new Object[] { username },
          new RowMapper<UserAttempts>() {
          public UserAttempts mapRow(ResultSet rs, int rowNum) throws SQLException {

             User user = new User();
             user.setId(rs.getInt("id"));
             user.setUsername(rs.getString("username"));
             user.setAge(rs.getInt("age"));
             user.setLastModified(rs.getDate("lastModified"));

             return user;
          }
                });
           return user;

    } catch (EmptyResultDataAccessException e) {
        return null;
    }
  }
```

spring jdbc

Related Articles

295k Spring JdbcTemplate batchUpdate() Exampl...

1.1m Spring JdbcTemplate Querying Examples

36k How to autowire DataSource in JdbcDaoSup...

91k JdbcTemplate queryForInt() is Deprecated

24k Spring - View content of HSQLDB embedded...

97k Spring Boot JDBC + Oracle database + Com...

About the Author

mkyong

Founder of Mkyong.com, love Java and open source stuff. Follow him on Twitter. If you like my tutorials, consider make a donation to these charities.

Comments

Join the discussion...

10 4 1 12

newest oldest most voted

Guest

Ariel

Do a regular query() and use DataAccessUtils.singleResult() instead, much easier, and does exactly what you need.

If you are returning a single column then SingleColumnRowMapper.newInstance() can be helpful.

Example:

```
return DataAccessUtils.singleResult(jdbcTemplate.query(query,
SingleColumnRowMapper.newInstance(Integer.class)));
```

This returns an Integer or NULL if there are no rows.

+ 5 − Reply 10 months ago

Guest

Fabio Almeida

MKyoung... you need to read the Clean Code by Robert Martin. Never return null. The Spring Framework is correct. Null can be returned by the JDBC Driver error ou a OutOfMemory. Y'Know... You only need to catch this exception and handle.

+ 3 − Reply 3 years ago

Guest

Cyril

There is the Optional class for that. You also shouldn't be throwing exceptions for expected behavior.

+ 9 − Reply 1 year ago

Guest

Dan

I tend to prefer just doing query() and getting a list, then returning the first item if it exists, else null.

+ 3 − Reply 4 years ago

Guest

Alessandro M.

// INSERT INTO a_table (column_a, nullable_column, column_b) VALUES ('a', NULL, 'b');

jdbc.queryForObject("SELECT nullable_column FROM a_table", params, String.class);

This is (also) why it does not return null.

+ 2 − Reply 2 years ago

Guest

Midos

In the solution above; what if i do not want to return null and I want to return an appropriate message in a json object response?

+ 1 − Reply 1 year ago

Guest

Matt

There is a trend away from the over use of null. Especially in languages/frameworks that include or are based on functional principles. When you return null, you still have to do a check outside the call for null, aka: if (myVar == null) {...} or if (myVar != null) {...}. So why not just catch the exception and handle it accordingly somewhere up the call stack? Otherwise you would have to catch the exception, return null (thus losing the context of the exception), check for null somewhere up the stack, handle accordingly if you can figure out what happened. Chances... Read more »

+ 1 − Reply 4 years ago

Guest

Matt

Programmer error being: "You idiot, you forgot to initialize me somewhere..."

+ 0 − Reply 4 years ago

Guest

Guest

Returning exception is not good when getting empty result is not an 'exceptional case'.

In terms of functional programming, it should return something like Scala's Option object.

+ 3 − Reply 4 years ago

Guest

Ruslan

Very usefull answer!!! thanks!

+ 0 − Reply 5 months ago

Guest

Ruslan

good answer!! thanks!!

+ 0 − Reply 5 months ago

Guest

chirag soni

Hi, bro, you are handling EmptyResultDataAccessException thrown by queryForObject() method but I want to know that when we use spring JDBC then handling the exception will be taken care by spring only then why you are handling that in your example.

+ 0 − Reply 10 months ago

Guest

Alon

If it's something that may occur often and is not really exception but a logical check, in my opinion it will be better to remove the catch and add check in mapRow

+ 0 − Reply 5 years ago

Guest

Charles

The exception is thrown by Spring when queryForObject is executed (before it maps the results), so you can't handle it in the row mapper.

+ 3 − Reply 5 years ago

Rising Articles

+3.1k Spring REST + Spring Security Example

+2.9k Spring REST Validation Example

+2.8k Java 8 - How to sort list with stream.so...

+2.7k Spring REST Error Handling Example

+2.2k Jackson - Convert JSON array string to L...

+2.1k Java - How to list all files in a direct...

+2k Spring Boot + JUnit 5 + Mockito

+2k Spring Boot - How to send email via SMTP

+1.9k Java - How to print an Array

+1.9k Maven - JaCoCo code coverage example
