Create custom method in hibernate

To create a custom method in Hibernate for JDBC interactions, you typically follow these steps:

1. **Define the Custom Method**: Create a method in your DAO (Data Access Object) or Repository class that performs the desired JDBC operation.

2. **Obtain JDBC Connection**: Obtain a JDBC Connection object from the Hibernate Session or SessionFactory.

3. **Execute JDBC Operation**: Use the JDBC Connection to execute the desired JDBC operation, such as executing a SQL query, updating data, etc.

4. **Handle Resources**: Properly handle resources like Statement, ResultSet, and Connection by closing them in a finally block to ensure they are released correctly, even in case of exceptions.

Here's an example to illustrate these steps:

```java
import org.hibernate.Session;
import org.hibernate.jdbc.Work;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class CustomRepository {

    private final Session sessionFactory;

    public CustomRepository(Session sessionFactory) {
        this.sessionFactory = sessionFactory;
    }

    // Custom method to execute JDBC query
    public List<String> executeCustomQuery() {
        List<String> results = new ArrayList<>();
        sessionFactory.doWork(new Work() {
            @Override
            public void execute(Connection connection) throws SQLException {
                // Create and execute SQL query
                String sql = "SELECT name FROM example_table";
                try (PreparedStatement statement =
connection.prepareStatement(sql);
                     ResultSet resultSet = statement.executeQuery()) {
                    // Process ResultSet
                    while (resultSet.next()) {
                        String name = resultSet.getString("name");
                        results.add(name);
                    }
                }
            }
        });
        return results;
    }
}
```

In this example:

- We have a `CustomRepository` class with a method `executeCustomQuery()` that executes a custom JDBC query.
- We obtain a JDBC Connection using `sessionFactory.doWork(new Work(){...})` method, passing a `Work` implementation.
- Inside the `Work` implementation, we create a PreparedStatement to execute a SQL query and process the ResultSet to retrieve the results.
- We return the results fetched from the database.

Remember to properly manage resources like Connection, PreparedStatement, and ResultSet to ensure efficient resource utilization and avoid resource leaks.