MySQL's JDBC Driver, And Making It Do What You Want

Mark Matthews
Enterprise Tools - MySQL @ Sun

Java

Sun microsystems

- Learn how to change core behavior of MySQL's JDBC™ driver without sending us a single patch

JDBC is Nice, But...

Sometimes You'd Like to Do More

- Profiling
- Security
- Query Rewrite
- Caching
- Semi-transparent scale-out

Sometimes Vendor Extensions Aren't Enough

- Usually can alter state, configuration of a session
- Can't inject new behavior
- Can't participate in "lifecycle" events

Sometimes You'd Like to Change Behavior

We don’t always know what the users want...

Sometimes…We guess wrong!

MySQL is Open Source!

Hack on the JDBC driver?

What happens when the next version ships?

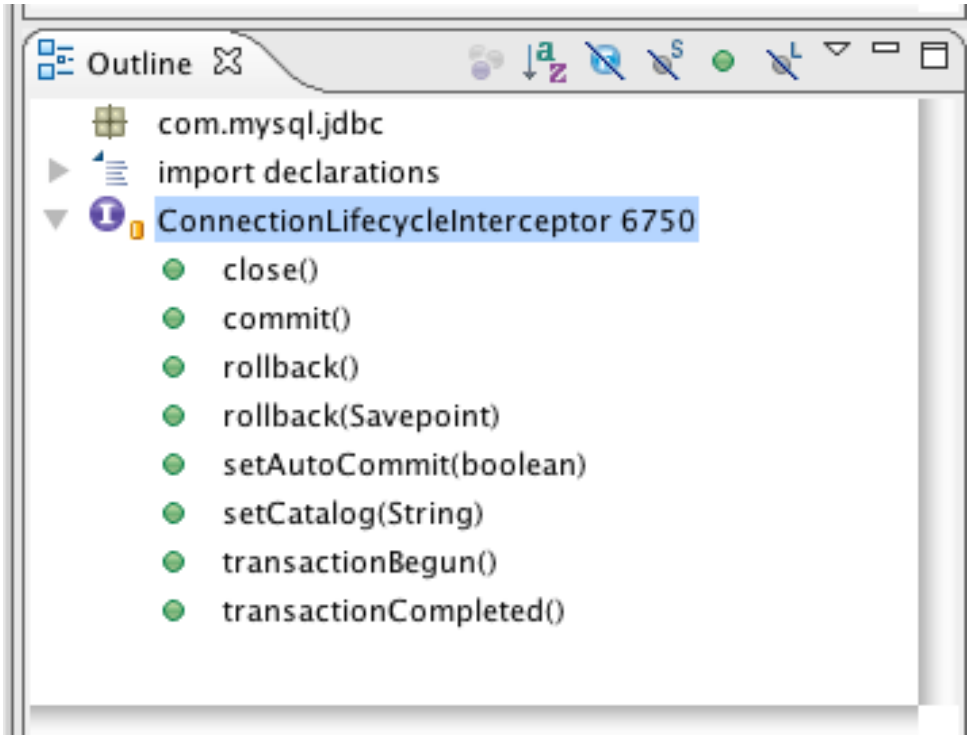
One Guess - Common Extension Points

Ho Hum Plugability

- Logging - pluggable
- Sockets - pluggable
- Profiler, usage advisor reporting - pluggable (but interesting to my team!)

Now, Something More Interesting

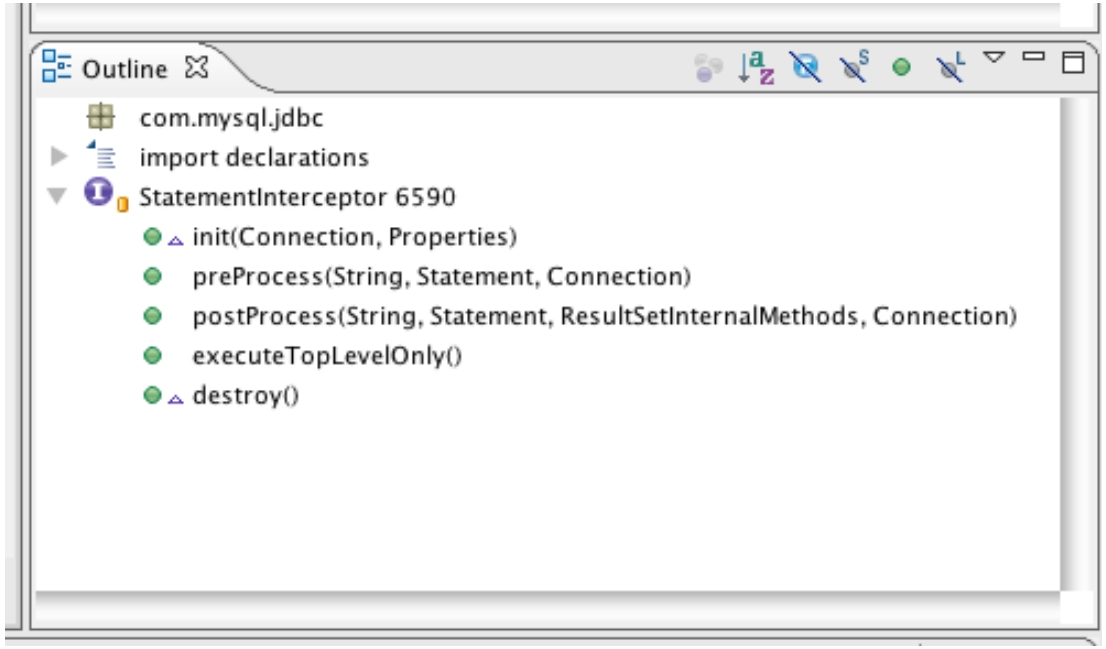
Connection Lifecycle



- Creation/Destruction
- Auto commit state change
- Application transaction demarcation
- Transaction state change on server

Statement Interception

- Inspired by Jan's work on mysql-proxy
- Pre/Post execution
- Access to original statement
- Access to original statement parameters (if prepared)
- Early return of ResultSets
- Can be chained



Some Examples

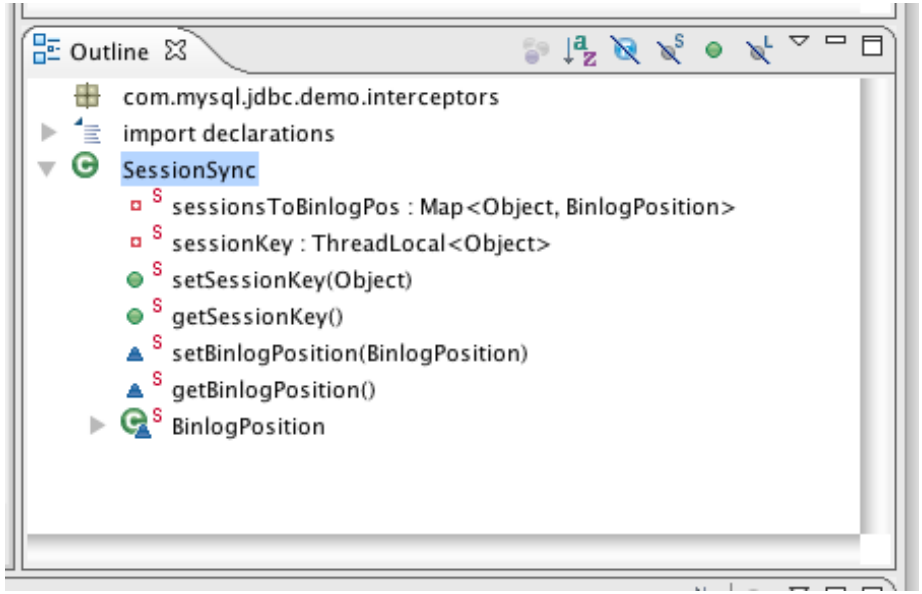
- Real-world problems
- Thought experiments for the purpose of this talk
- Do you have more intriguing use cases?
- Will you share?

Use Lifecycle and Injection for Replication Consistency

Enabling Features

- MySQL's JDBC "ReplicationConnection"
  - jdbc:mysql:replication://master, slave1, slave2,slaveN.../
- For extra credit - mysql-proxy-based MySQL Enterprise™ Load Balancer with slave-awareness
  - jdbc:mysql:replication://master,load-balancer/
- Allows dynamic management of slaves

Multi Transaction Unit-of-Work



- Opaque token...HTTP session ID?Current thread?
- Set in business logicSet in ServletFilter (similar to “open session in view”)

Finding Where to Wait

```
public boolean transactionCompleted() throws SQLException {
        if (!this.ownerConn.isReadOnly()) {
            ResultSet rs =
this.ownerConn.createStatement().executeQuery(
                    "SHOW MASTER STATUS");

            try {
                rs.next();

                SessionSync.setBinlogPosition(new
                    SessionSync.BinlogPosition(rs.getString(1),
rs.getLong(2)));
            }  finally {....}
        }

        return true;
    }
```

Hurry Up and Wait...

```
if (connection.isReadOnly() && !connection.getAutoCommit()
            && !this.hasWaited) {

        PreparedStatement waitStmt = connection
              .prepareStatement("SELECT MASTER_POS_WAIT(?,?,?)");
        ...

        try {
            SessionSync.BinlogPosition binlogPosition = SessionSync
                  .getBinlogPosition();
            waitStmt.setString(1, binlogPosition.filename);
            waitStmt.setLong(2, binlogPosition.position);
            waitStmt.setInt(3, 30 /* timeout */);

            rs = waitStmt.executeQuery();

            rs.next();

            int waitResult = rs.getInt(1);

                          // error checking....
        } finally {
            ...
        }
    }

    return null; // process the original statement as-is
}
```

```
            SessionSync.BinlogPosition binlogPosition =
SessionSync.getBinlogPosition();

            waitStmt.setString(1, binlogPosition.filename);
            waitStmt.setLong(2, binlogPosition.position);
            waitStmt.setInt(3, 30 /* timeout */);

            rs = waitStmt.executeQuery();

            rs.next();

            int waitResult = rs.getInt(1);

           // error checking....

return null; // process the original statement as-is
   }
```

Use Statement Interception for External Cache Coherency

Piggyback a Magic Token on the Transaction

```
private static final String MEMCACHED_TOKEN = "/* memcached:";
private static final int TOKEN_OFFSET = MEMCACHED_TOKEN.length();

public ResultSetInternalMethods preProcess(String sql,
        Statement interceptedStatement, Connection connection)
        throws SQLException {

    if (sql.startsWith(MEMCACHED_TOKEN)) {
        int endIndex = sql.indexOf("*/", TOKEN_OFFSET);

        if (endIndex != -1) {
            this.evictKey = sql.substring(TOKEN_OFFSET, endIndex).trim();
        }
    }

    return null;
}
```

Queue for Eviction on DML Success

```
public ResultSetInternalMethods postProcess(String sql,
        Statement interceptedStatement,
        ResultSetInternalMethods originalResultSet, Connection connection)
        throws SQLException {
    if (originalResultSet.getUpdateCount() > 0) {
        if (this.evictKey != null) {
            connectionLocal.get(connection).evict(this.evictKey);

            this.evictKey = null;
        }
    }

    return null;
}
```

Evict on Transaction Completion

```
public boolean transactionCompleted() throws SQLException {
        if (!this.didRollback && this.didCommit) {
                connectionLocal.get(this.myConnection).transactionCompleted();
        }

        return true;
}

public boolean transactionBegun() throws SQLException {
        this.didCommit = false;
        this.didRollback = false;

        return true;
}

public boolean commit() throws SQLException {
        this.didCommit = true; return true;
}

public boolean rollback() throws SQLException {
        this.didRollback = true; return true;
}
```

Wiring it All Together

```
class CacheSynchronizer {
...

    void evict(String key) {
        evictions.add(key); // a List<String>
    }

    void transactionCompleted() {
        for (String key : this.evictions) {
            this.memcached.delete(key);
        }

        evictions.clear();
    }

    void destroy() {
        this.pool.shutDown();
    }
}
```

Use Statement Interception to Prevent Database Extrusion

```
        return
(ResultSetInternalMethods)Proxy.newProxyInstance(originalResultSet.getClass()
.getClassLoader(),
                new Class[] {ResultSetInternalMethods.class},
                    new InvocationHandler() {

                     public Object invoke(Object proxy, Method method,
                                Object[] args) throws Throwable {

                        Object invocationResult = method.invoke(finalResultSet, args);

                        String methodName = method.getName();

                         if (invocationResult != null &&
                           invocationResult instanceof String) {
                            if (regexP.matcher(invocationResult.toString()).matches()) {
                                    throw new SQLException("value disallowed by filter");
                            }
                         }

                         return invocationResult;
                       }});
```

Horizontal Partitioning with Proxy

```
 private String lastShardKey = null;

public static void setShardKey(String key) {
    threadLocalKey.set(key);
}

public ResultSetInternalMethods preProcess(String sql,
        Statement interceptedStatement, Connection connection)
        throws SQLException {

    String candidateKey = threadLocalKey.get();

    if (lastShardKey == null || !lastShardKey.equals(candidateKey)) {
        this.variableStmt.setString(1, candidateKey);
        this.variableStmt.execute();
        this.lastShardKey = candidateKey;
    }

    return null; // process the original statement as-is
}
```

Port a Black Box Application...

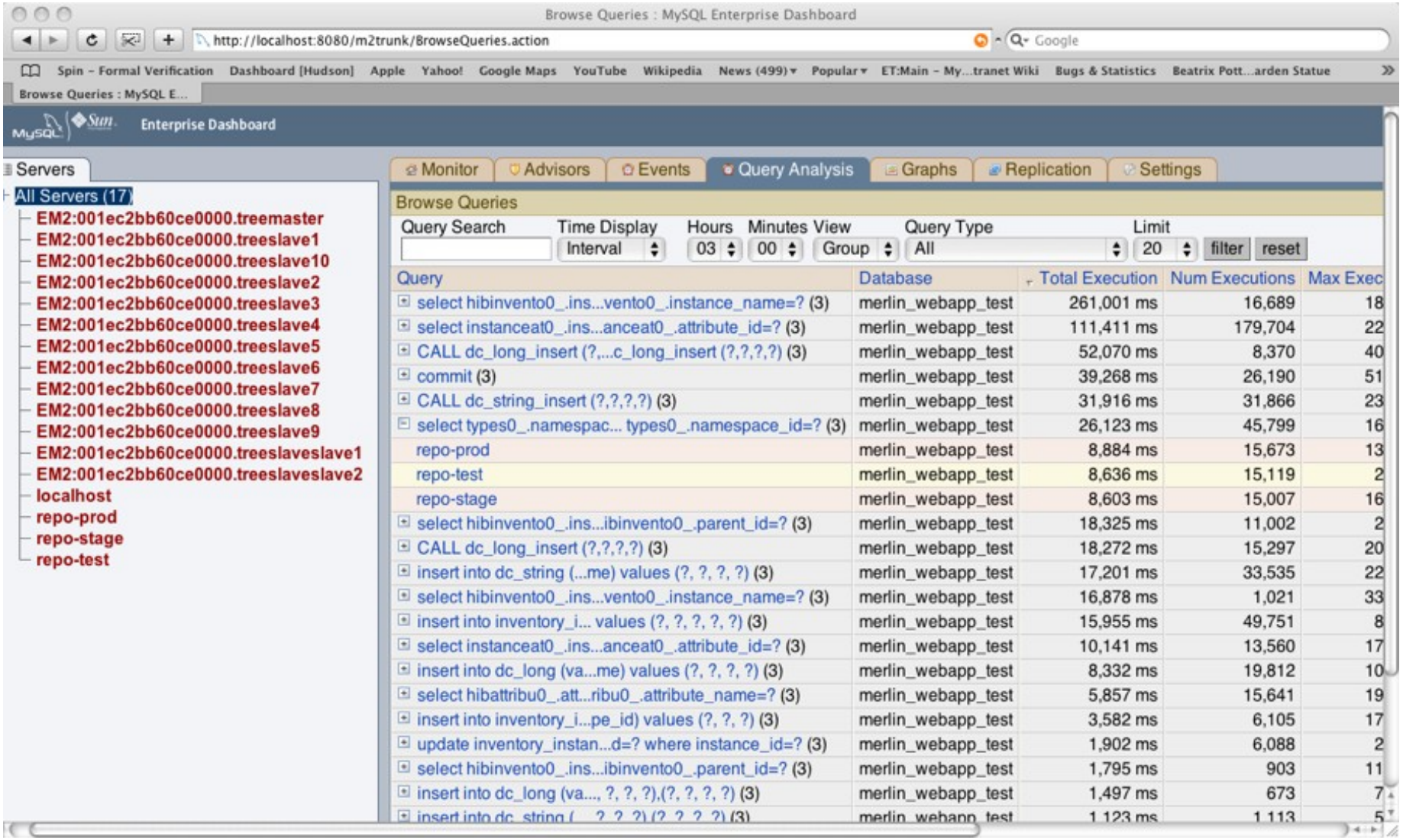
```
public ResultSetInternalMethods preProcess(String sql,
            Statement interceptedStatement, Connection connection)
            throws SQLException {
        if (interceptedStatement instanceof PreparedStatement) {
            String preparedSql = ((PreparedStatement) interceptedStatement)
                    .getPreparedSql();

            if (preparedSql.equals("SELECT TOP ? * FROM foo WHERE a=? AND b=?")) {
                java.sql.PreparedStatement pstmt = connection
                        .prepareStatement("SELECT * FROM foo WHERE a=? AND b=?
LIMIT ?");
                ParameterBindings bindings = ((PreparedStatement)
interceptedStatement)
                        .getParameterBindings();
                pstmt.setObject(1, bindings.getObject(2));
                pstmt.setObject(2, bindings.getObject(3));
                pstmt.setObject(3, bindings.getObject(1));

                return (ResultSetInternalMethods) pstmt.executeQuery();
            }
```

Evil Testing

How well does your application...

- Handle latency in query processing
  - Implement a random-latency SocketInputStream
- Handle transaction completion failure
  - Hook rollback/commit and throw 08007

Dogfood Time...


Browse Queries : MySQL Enterprise Dashboard
http://localhost:8080/m2trunk/BrowseQueries.action
Google
Spin – Formal Verification
Dashboard [Hudson]
Apple
Yahoo!
Google Maps
YouTube
Wikipedia
News (499)
Popular
ET:Main – My...tranet Wiki
Bugs & Statistics
Beatrix Pott...arden Statue
Browse Queries : MySQL E...
MySQL Sun Enterprise Dashboard
Servers
All Servers (17)
EM2:001ec2bb60ce0000.treemaster
EM2:001ec2bb60ce0000.treeslave1
EM2:001ec2bb60ce0000.treeslave10
EM2:001ec2bb60ce0000.treeslave2
EM2:001ec2bb60ce0000.treeslave3
EM2:001ec2bb60ce0000.treeslave4
EM2:001ec2bb60ce0000.treeslave5
EM2:001ec2bb60ce0000.treeslave6
EM2:001ec2bb60ce0000.treeslave7
EM2:001ec2bb60ce0000.treeslave8
EM2:001ec2bb60ce0000.treeslave9
EM2:001ec2bb60ce0000.treeslaveslave1
EM2:001ec2bb60ce0000.treeslaveslave2
localhost
repo-prod
repo-stage
repo-test
Monitor
Advisors
Events
Query Analysis
Graphs
Replication
Settings
Browse Queries
Query Search
Time Display
Interval
Hours
03
Minutes
00
View
Group
Query Type
All
Limit
20
filter
reset
Query
Database
Total Execution
Num Executions
Max Exec
select hibinvento0_.ins...vento0_.instance_name=? (3) merlin_webapp_test 261,001 ms 16,689 18
select instanceat0_.ins...anceat0_.attribute_id=? (3) merlin_webapp_test 111,411 ms 179,704 22
CALL dc_long_insert (?,...c_long_insert (?,?,?,?) (3) merlin_webapp_test 52,070 ms 8,370 40
commit (3) merlin_webapp_test 39,268 ms 26,190 51
CALL dc_string_insert (?,?,?,?) (3) merlin_webapp_test 31,916 ms 31,866 23
select types0_.namespac... types0_.namespace_id=? (3) merlin_webapp_test 26,123 ms 45,799 16
repo-prod merlin_webapp_test 8,884 ms 15,673 13
repo-test merlin_webapp_test 8,636 ms 15,119 2
repo-stage merlin_webapp_test 8,603 ms 15,007 16
select hibinvento0_.ins...ibinvento0_.parent_id=? (3) merlin_webapp_test 18,325 ms 11,002 2
CALL dc_long_insert (?,?,?,?) (3) merlin_webapp_test 18,272 ms 15,297 20
insert into dc_string (...me) values (?, ?, ?, ?) (3) merlin_webapp_test 17,201 ms 33,535 22
select hibinvento0_.ins...vento0_.instance_name=? (3) merlin_webapp_test 16,878 ms 1,021 33
insert into inventory_i... values (?, ?, ?, ?, ?) (3) merlin_webapp_test 15,955 ms 49,751 8
select instanceat0_.ins...anceat0_.attribute_id=? (3) merlin_webapp_test 10,141 ms 13,560 17
insert into dc_long (va...me) values (?, ?, ?, ?) (3) merlin_webapp_test 8,332 ms 19,812 10
select hibattribu0_.att...ribu0_.attribute_name=? (3) merlin_webapp_test 5,857 ms 15,641 19
insert into inventory_i...pe_id) values (?, ?, ?) (3) merlin_webapp_test 3,582 ms 6,105 17
update inventory_instan...d=? where instance_id=? (3) merlin_webapp_test 1,902 ms 6,088 2
select hibinvento0_.ins...ibinvento0_.parent_id=? (3) merlin_webapp_test 1,795 ms 903 11
insert into dc_long (va..., ?, ?, ?),(?, ?, ?, ?) (3) merlin_webapp_test 1,497 ms 673 7

Why, oh why?

```
public void passivateObject(Object obj) throws Exception {
    if(obj instanceof Connection) {
        Connection conn = (Connection)obj;
        if(!conn.getAutoCommit() && !conn.isReadOnly()) {
            conn.rollback();
        }
        conn.clearWarnings();
        conn.setAutoCommit(true);
    }
    if(obj instanceof DelegatingConnection) {
        ((DelegatingConnection)obj).passivate();
    }
}
```

What do I do now????

- Fork DBCP?
- Submit a patch? When will it get in?
- Patch the JDBC Driver?

Aha!

```
public boolean setAutoCommit(boolean flag) throws
   SQLException {
        if (!flag) {
            return true;
        }

        return false;
}
```

- Autocommit toggling 10% or more of total execution time
- 2nd largest total execution time in our application
- Fixed with 4 lines of code and a configuration parameter - no patches to core code

Summary

- It's possible (easy?) to alter the behavior of MySQL's JDBC driver at runtime
- Use connection interceptors for participating/influencing transactions
- Use statement interceptors for changing the behavior of individual queries
- Combine to create more "interesting" behavior

Questions?

For More Information

- http://forge.mysql.com/
- java@lists.mysql.com
- mark@mysql.com

THANK YOU

MySQL's JDBC Driver, And Making It Do What You Want

Mark Matthews

Enterprise Tools - MySQL @ Sun