S1. result = sqlSession.selectOne(command.getName(), param);执行DefaultSqlSession的selectOne方法；

在DefaultSqlSession中最终由selectList执行

```
1  public <E> List<E> selectList(String statement, Object parameter, RowBoun
ds rowBounds) {
2   try {
3   MappedStatement ms = configuration.getMappedStatement(statement);
4   return executor.query(ms, wrapCollection(parameter), rowBounds,
Executor.NO_RESULT_HANDLER);
5   } catch (Exception e) {
6   throw ExceptionFactory.wrapException("Error querying database. Cause: "
+ e, e);
7   } finally {
8   ErrorContext.instance().reset();
9   }
10  }
```

2. executor是之前创建的有三种（SimpleExecutor,ReuseExecutor,BatchExecutor以及他们的父类BaseExecutor 还有其它暂时先不管），这边是SimpleExecutor，query方法是由BaseExecutor实现的，所以首先进入BaseExecutor的query方法

```
1   public <E> List<E> query(MappedStatement ms, Object parameter, RowBounds
rowBounds, ResultHandler resultHandler) throws SQLException {
2   BoundSql boundSql = ms.getBoundSql(parameter);
3   CacheKey key = createCacheKey(ms, parameter, rowBounds, boundSql);
4   return query(ms, parameter, rowBounds, resultHandler, key, boundSql);
5  }
6
7  public <E> List<E> query(MappedStatement ms, Object parameter, RowBounds
rowBounds, ResultHandler resultHandler, CacheKey key, BoundSql boundSql) th
rows SQLException {
8   ErrorContext.instance().resource(ms.getResource()).activity("executing a
query").object(ms.getId());
9   if (closed) {
10   throw new ExecutorException("Executor was closed.");
11   }
12   if (queryStack == 0 && ms.isFlushCacheRequired()) {
13   clearLocalCache();
14   }
15   List<E> list;
16   try {
17   queryStack++;
```

```
18   list = resultHandler == null ? (List<E>) localCache.getObject(key) : nu
ll;
19   if (list != null) {
20   handleLocallyCachedOutputParameters(ms, key, parameter, boundSql);
21   } else {
22   list = queryFromDatabase(ms, parameter, rowBounds, resultHandler, key,
boundSql);
23   }
24   } finally {
25   queryStack--;
26   }
27   if (queryStack == 0) {
28   for (DeferredLoad deferredLoad : deferredLoads) {
29   deferredLoad.load();
30   }
31   // issue #601
32   deferredLoads.clear();
33   if (configuration.getLocalCacheScope() == LocalCacheScope.STATEMENT) {
34   // issue #482
35   clearLocalCache();
36   }
37   }
38   return list;
39  }
40
41  private <E> List<E> queryFromDatabase(MappedStatement ms, Object paramet
er, RowBounds rowBounds, ResultHandler resultHandler, CacheKey key, BoundSq
l boundSql) throws SQLException {
42   List<E> list;
43   localCache.putObject(key, EXECUTION_PLACEHOLDER);
44   try {
45   list = doQuery(ms, parameter, rowBounds, resultHandler, boundSql);
46   } finally {
47   localCache.removeObject(key);
48   }
49   localCache.putObject(key, list);
50   if (ms.getStatementType() == StatementType.CALLABLE) {
51   localOutputParameterCache.putObject(key, parameter);
52   }
53   return list;
54  }
```

3. 顺序执行上述三个方法以后，会进入SimpleExecutor的doQuery方法

```
1  public <E> List<E> doQuery(MappedStatement ms, Object parameter, RowBound
s rowBounds, ResultHandler resultHandler, BoundSql boundSql) throws SQLExce
ption {
2    Statement stmt = null;
3    try {
4    Configuration configuration = ms.getConfiguration();
5    StatementHandler handler = configuration.newStatementHandler(wrapper,
ms, parameter, rowBounds, resultHandler, boundSql);
6    stmt = prepareStatement(handler, ms.getStatementLog());
7    return handler.<E>query(stmt, resultHandler);
8    } finally {
9    closeStatement(stmt);
10    }
11  }
```

4. 最后在SimpleStatementHandler中执行jdbc的Statement.execute真正的执行sql

```
1  public <E> List<E> query(Statement statement, ResultHandler
resultHandler) throws SQLException {
2    String sql = boundSql.getSql();
3    statement.execute(sql);
4    return resultSetHandler.<E>handleResultSets(statement);
5  }
```

5. 执行过程如下