

Eclipse でのデバッグのやり方

麻生情報ビジネス専門学校

目次

| | |
|--------------------------------------|----|
| 1. はじめに | 3 |
| 2. よく出るエラーメッセージ | 4 |
| ● エラーメッセージを見るコツ | 4 |
| ● エラーメッセージを見るコツまとめ | 8 |
| 3. デバッグ方法 | 9 |
| ● デバッグの手順 | 9 |
| 1. ブ레이크ポイントを設定する | 10 |
| 2. デバッグを実行する | 11 |
| 3. ステップ実行や変数ダンプ機能などでプログラムを調査する | 11 |
| 4. よくある問い合わせ | 15 |

1. はじめに

eclipse を使って開発をしているときにこんな場面に総合したことはありませんか？

- ・ コンソールに赤い文字でエラーが出たけど意味が分からない
- ・ エラーはないけど、うまく動かなくてどこが悪いのかが分からない

本書はこれらの場合にどういう風に対処すればよいかを記載しています。

開発の時にわからないことがあれば、まず本書を見てしらべてみては如何でしょうか

2. よく出るエラーメッセージ

● エラーメッセージを見るコツ

エラーは、eclipse のコンソールに出ることが多いです。

表示されるエラーは「例外発生時のメッセージ」がほとんどです。

例外発生時のメッセージにはどこを見ればよいかのコツがあります。

エラーがコンソールに表示された例↓

```
java.sql.SQLException: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'mail=' AND password=''' at line 1
at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:118)
at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:95)
at com.mysql.cj.jdbc.exceptions.SQLExceptionMapping.translateException(SQLExceptionMapping.java:122)
at com.mysql.cj.jdbc.ClientPreparedStatement.executeInternal(ClientPreparedStatement.java:980)
at com.mysql.cj.jdbc.ClientPreparedStatement.executeQuery(ClientPreparedStatement.java:1013)
at org.apache.tomcat.dbcp.dbcp2.DelegatingPreparedStatement.executeQuery(DelegatingPreparedStatement.java:82)
at org.apache.tomcat.dbcp.dbcp2.DelegatingPreparedStatement.executeQuery(DelegatingPreparedStatement.java:82)
at gatcha.model.LoginModel.login(LoginModel.java:51)
at gatcha.controller.LoginServlet.doPost(LoginServlet.java:31)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:648)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:723)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:232)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)
at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:240)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)
at gatcha.filter.LoginCheckFilter.doFilter(LoginCheckFilter.java:58)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:240)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:212)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:34)
at org.apache.catalina.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:504)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:141)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:79)
at org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:620)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:88)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:502)
at org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java:1132)
at org.apache.coyote.AbstractProtocol$AbstractConnectionHandler.process(AbstractProtocol.java:684)
at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1533)
at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.run(NioEndpoint.java:1483)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
at java.lang.Thread.run(Thread.java:748)
```

注目すべきは一番最初の↓の行です

```
java.sql.SQLException: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'mail=' AND password=''' at line 1
at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:118)
at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:95)
at com.mysql.cj.jdbc.exceptions.SQLExceptionMapping.translateException(SQLExceptionMapping.java:122)
```

`java.sql.SQLException` は発生した例外クラスの名前です。

大抵の場合、発生した例外から「どういうエラーが発生したか」を想像できます。

```
java.sql.SQLException: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'mail=' AND password=''' at line 1
at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:118)
at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:95)
at com.mysql.cj.jdbc.exceptions.SQLExceptionMapping.translateException(SQLExceptionMapping.java:122)
```

例外クラスの後続くのはエラーメッセージの具体的な内容で、このメッセージをコピーしてグーグルなどで検索すると、有益な情報が見つかることが多いです。

代表的な例外クラスを挙げておきます

| 例外クラス (パッケージは省略) | 発生しているエラー |
|-----------------------|--|
| SQLException | SQL の文法 (Syntax) エラー この例外が発生したときは、SQL が間違えていることが多い |
| NullPointerException | インスタンスが無い (null) なのにインスタンス変数やインスタンスメソッドにアクセスしようとするると発生する。 要するに new していないってこと。 エラーの例) LoginInfoBeans beans = null; beans.setName("nishino"); //beans は new していないので //Null ポインターエラーになる |
| NumberFormatException | Integer.parseInt など、文字列→数値に変換しようとしたときのエラー。 数値でない文字を変換しようとしたり、空文字を変換しようとするると発生する |
| SQLException | SQL (データベース) 関連のエラー このエラーが発生するのはいろんな原因があるので、後続のエラーメッセージを確認する必要がある |

また、エラーメッセージの 2 行目以降は、「スタックトレース」と言って、メソッドの呼び出し履歴が表示されています。要するに「ソースコード上のどこでエラーが発生したか」が表示されています。↓

```

Exception in thread "main" java.sql.SQLException: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'mail=' AND password=''' at line 1
at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:118)
at com.mysql.cj.jdbc.exceptions.SQLError.createSQLException(SQLError.java:95)
at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
at com.mysql.cj.jdbc.ClientPreparedStatement.executeInternal(ClientPreparedStatement.java:960)
at com.mysql.cj.jdbc.ClientPreparedStatement.executeQuery(ClientPreparedStatement.java:1019)
at org.apache.tomcat.dbcp.dbcp2.DelegatingPreparedStatement.executeQuery(DelegatingPreparedStatement.java:82)
at org.apache.tomcat.dbcp.dbcp2.DelegatingPreparedStatement.executeQuery(DelegatingPreparedStatement.java:82)
at gatcha.model.LoginModel.login(LoginModel.java:51)
at gatcha.controller.LoginServlet.doPost(LoginServlet.java:31)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:648)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:232)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)
at org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:240)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)
at gatcha.filter.LoginCheckFilter.doFilter(LoginCheckFilter.java:58)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.java:240)
at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:207)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:212)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:84)
at org.apache.catalina.core.authenticator.AuthenticatorBase.invoke(AuthenticatorBase.java:504)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:141)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:79)
at org.apache.catalina.valves.AbstractAccessLogValve.invoke(AbstractAccessLogValve.java:620)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:88)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:502)
at org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java:1132)
at org.apache.coyote.http11.AbstractHttp11Processor.process(AbstractHttp11Processor.java:684)
at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.doRun(NioEndpoint.java:1533)
at org.apache.tomcat.util.net.NioEndpoint$SocketProcessor.run(NioEndpoint.java:1488)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617)
at org.apache.tomcat.util.threads.TaskThread$WrappingRunnable.run(TaskThread.java:61)
at java.lang.Thread.run(Thread.java:748)

```

スタックトレース

スタックトレースの見方にもコツがあります。

トレースは上の方が最新です。一番上に書かれている行でエラーになったということです。
at の後は、呼び出したメソッド、カッコ内はそのメソッドがあるファイルとそのメソッドを呼び出している行数です。

```
at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:118)
```

呼び出したメソッド

メソッドがある java ファイル
名と行数

上記の場合、

パッケージ名が `com.mysql.cj.jdbc.exceptions` というパッケージ内にある

クラス名が `SQLException` というクラスの

メソッド名が `createSQLException` というメソッドを呼び出したということです。

さらに、そのメソッドの呼び出しは `SQLException.java` の 118 行目で行っているよ
という意味になります。

そして、スタックトレースの下から順にメソッドが呼ばれてるということになります。

例えば・・・

```
at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:118)
at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:95)
at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
at com.mysql.cj.jdbc.ClientPreparedStatement.executeInternal(ClientPreparedStatement.java:960)
at com.mysql.cj.jdbc.ClientPreparedStatement.executeQuery(ClientPreparedStatement.java:1019)
at org.apache.tomcat.dbcp.dbcp2.DelegatingPreparedStatement.executeQuery(DelegatingPreparedStatement.java:82)
at org.apache.tomcat.dbcp.dbcp2.DelegatingPreparedStatement.executeQuery(DelegatingPreparedStatement.java:82)
at gatcha.model.LoginModel.login(LoginModel.java:51)
at gatcha.controller.LoginServlet.doPost(LoginServlet.java:31)
```

上記の場合、下から順にメソッドが呼ばれるので

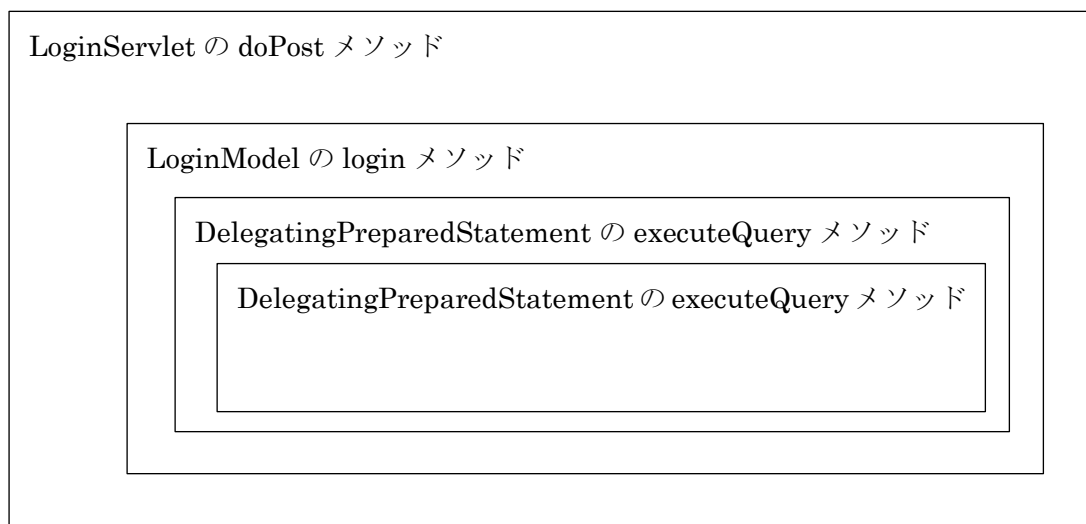
`LoginServlet` の `doPost` の中で `LoginModel` の `login` メソッドが呼ばれ

`LoginModel` の `login` の中で `DelegatingPreparedStatement` の `executeQuery` が呼ばれ

`DelegatingPreparedStatement` の `executeQuery` の中で、`DelegatingPreparedStatement`
の `executeQuery` が再度呼ばれ・・・と呼ばれているのが分かります。

※メソッドの中で呼ばれていることに注意！

イメージ図



自分が作ったメソッドのうち、一番上にあるものを見つけると、どこでエラーになったかが特定できます。

例えば・・・

ここは自分が作ったコードじゃない

```
java.sql.SQLException: You have an error in your SQL syntax; check the manual that corresponds to your
at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:118)
at com.mysql.cj.jdbc.exceptions.SQLException.createSQLException(SQLException.java:95)
at com.mysql.cj.jdbc.exceptions.SQLExceptionsMapping.translateException(SQLExceptionsMapping.java:122)
at com.mysql.cj.jdbc.ClientPreparedStatement.executeInternal(ClientPreparedStatement.java:960)
at com.mysql.cj.jdbc.ClientPreparedStatement.executeQuery(ClientPreparedStatement.java:1019)
at org.apache.tomcat.dbcp.dbcp2.DelegatingPreparedStatement.executeQuery(DelegatingPreparedStatement.java:82)
at org.apache.tomcat.dbcp.dbcp2.DelegatingPreparedStatement.executeQuery(DelegatingPreparedStatement.java:82)
at gatcha.model.LoginModel.login(LoginModel.java:51)
at gatcha.controller.LoginServlet.doPost(LoginServlet.java:31)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:648)
at javax.servlet.http.HttpServlet
```

これは、自分が作ったコードのうち一番上にある
つまり LoginModel.java の 51 行目でエラーが発生したとわかる

● エラーメッセージを見るコツまとめ

ここまでをまとめます。

コンソールに何かエラーが表示されたときにしらべる手順は以下の通りです。

1. 1 行目を見てどんなエラーかを理解する

→発生している例外から推測する。わからない場合はメッセージをググる！

2. スタックトレースからどこでエラーになったかを調べる

→発生している場所がわかれば、その前後のコードから理由を推測できる

3. エラーになった個所をじっくり見る

→エラー内容とソースコードからエラーの理由を考える

3. デバッグ方法

エラーが表示された場合は、エラーメッセージからどこが悪いかが推測できますがエラーが起きないのに、うまく動かない場合はどこが悪いのか特定するのが難しいです。

こういう場合に役に立つのが `eclipse` のデバッグ機能です。

デバッグ機能は以下のことができます。

- ・ステップ実行（1 行ずつプログラムを実行する）
- ・実行中の変数の中身を表示する

ステップ実行は 1 行ずつ実行するので、プログラムの流れが追いやすく、どこでエラーが発生しているかを発見しやすくなります。

変数の中身をみることで、例えば、値が取れているはずなのに取れていないなど、ダメな部分に気づくことができます。

● デバッグの手順

デバッグは以下の手順で行います。

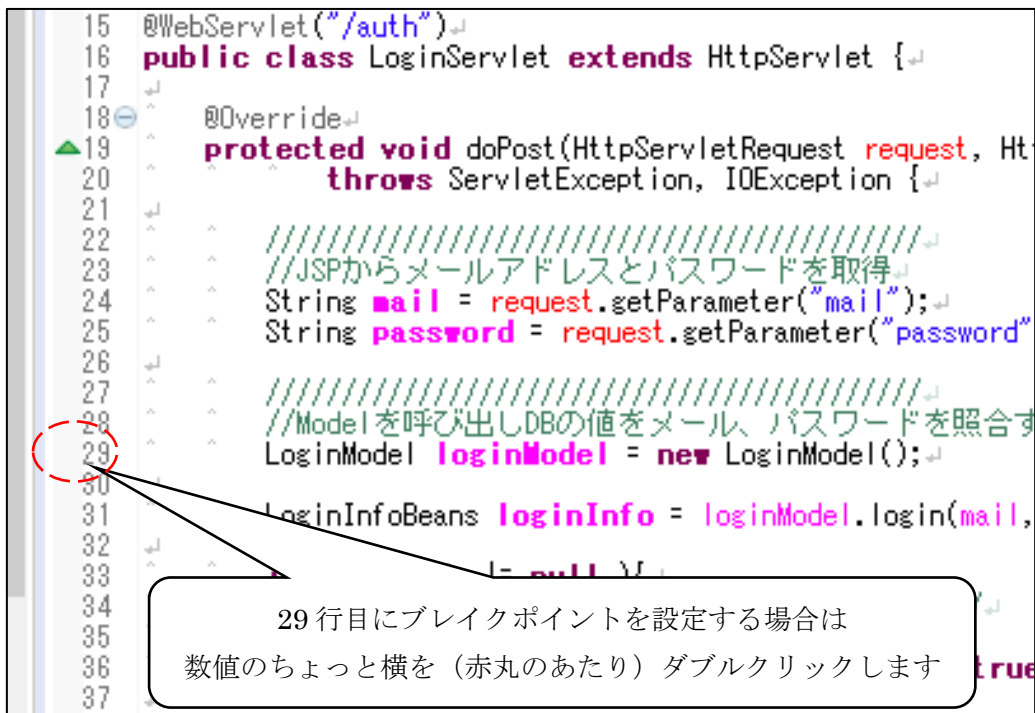
1. ブレイクポイントを設定する
2. デバッグを実行する
3. ステップ実行や変数ダンプ機能などでプログラムを調査する

次ページから 1 手順ごと解説します。

1. ブレイクポイントを設定する

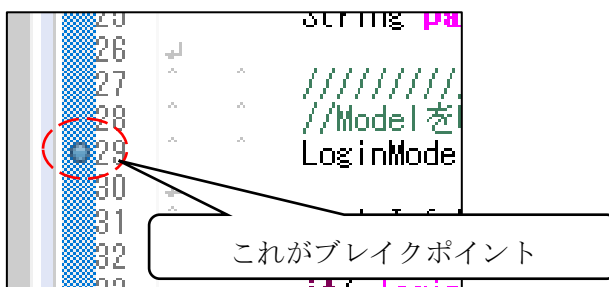
デバッグでは「**ブレイクポイント**」と言われるものをソースコード上に設定します。ブレイクポイントを設定することで、デバッグの時、プログラムを一時停止することができます。

ブレイクポイントを設定するのは、設定したい行をクリックすれば OK です。例えば、以下のソースコードの 29 行目にブレイクポイントを設定するとします。



クリックすると、小さい丸が付きます。これがブレイクポイントです。

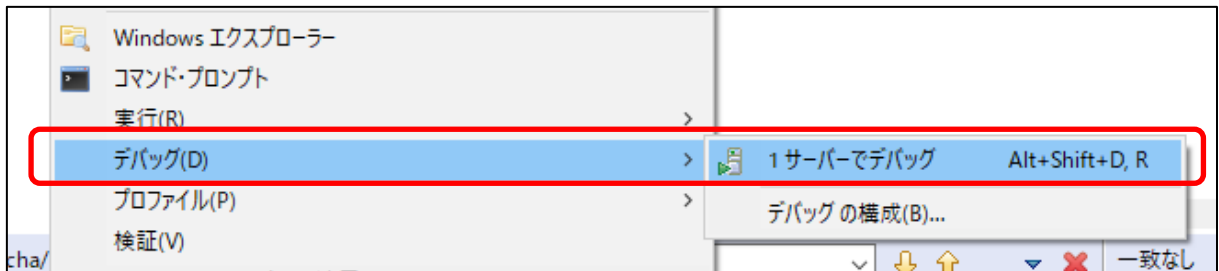
ブレイクポイントは複数設定できます！



※もう一度ダブルクリックすると、ブレイクポイントを解除できます。

2. デバッグを実行する

デバッグを実行するには、いつものように実行するファイルを選んで右クリックをして「実行」ではなく「デバッグ」を選びます。

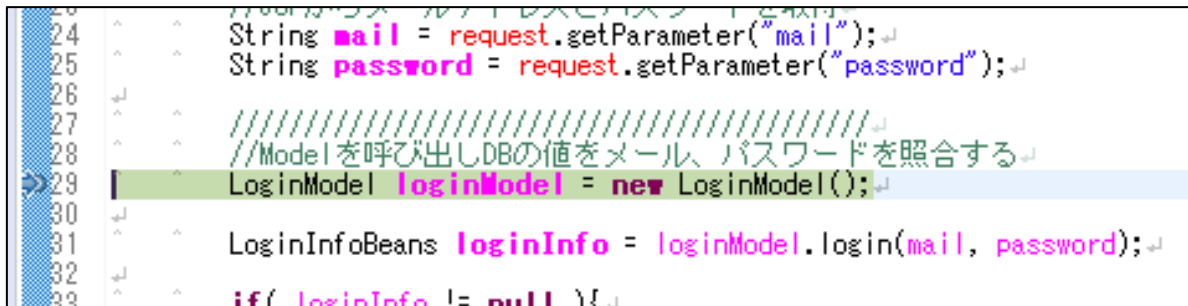


実行後、デバッグ用のパースペクティブを開くかどうかの確認画面が表示されますが、OKで閉じてください。

3. ステップ実行や変数ダンプ機能などでプログラムを調査する

デバッグ開始後は、いつもと同じように、画面を操作します。すると、ブレイクポイントを指定したところで、プログラムが一時停止します。

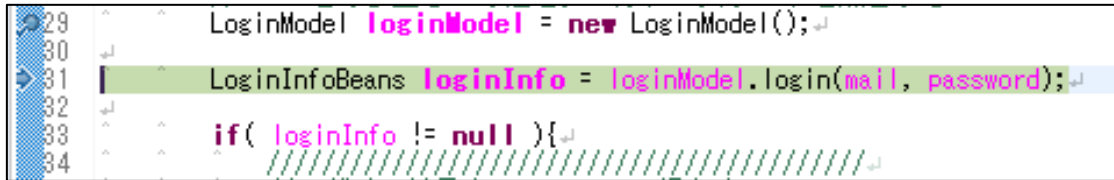
↓ブレイクポイントで止まった様子



この状態から、プログラムを1行ずつ進めたりして調査します。

| コマンド | 意味 |
|-------|--|
| F6 キー | ステップオーバー。 今のメソッドを1行進める。メソッドの中には入らない |
| F5 キー | ステップイン。 今のメソッドを1行進める。メソッドの中に入る |
| F8 キー | 再開。次のブレイクポイントまで実行する。 |
| F7 キー | ステップリターン。 今いるメソッドからリターン（抜ける） |

ステップオーバー、ステップインの違いについて



```
29 LoginModel loginModel = new LoginModel();  
30  
31 LoginInfoBeans loginInfo = loginModel.login(mail, password);  
32  
33 if( loginInfo != null ){  
34     //////////////////////////////////////
```

例えばこの状況の時に

F6 キー（ステップオーバー）をクリックすると、次の 32 行目（if 文）に行きます

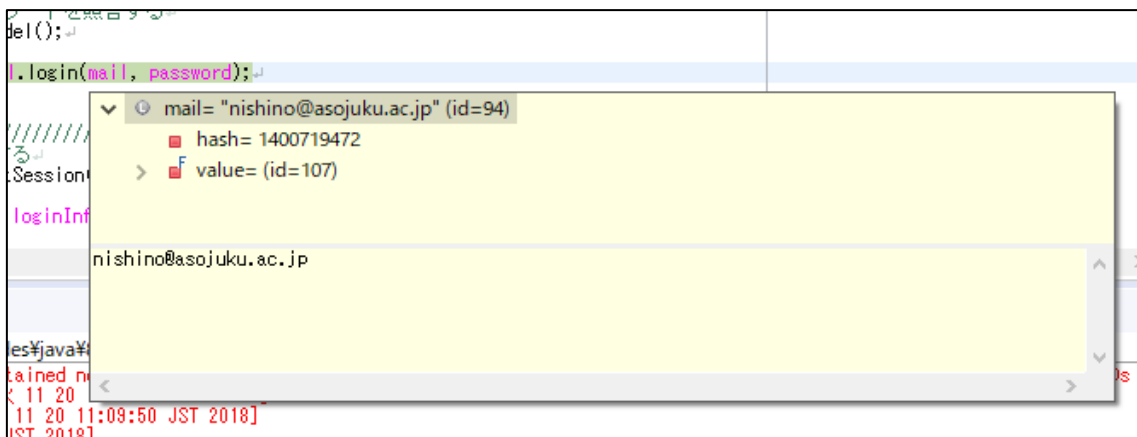
F5 キー（ステップイン）をクリックすると、loginModel の login メソッドに入ります

変数の中身を見るには？

方法が二つあります

1 つは、単純に中身を見たい変数にカーソルを合わせれば中身を見ることができます。

↓変数「mail」にカーソルを合わせると、ツールチップが表示され変数の中身が確認できる



```
del();  
loginModel.login(mail, password);  
////////////////////////////////////  
Session  
loginInfo  
nishino@asojuku.ac.jp  
les*java*  
tained m  
< 11:20  
11 20 11:09:50 JST 2018]  
JST 2018]
```

もうひとつは、右上の変数ウィンドウで確認する方法です

(x)= 変数 ☒ ブレークポイント

| 名前 | 値 |
|--------------|---------------------------------|
| > this | LoginServlet (id=85) |
| > request | RequestFacade (id=86) |
| > response | ResponseFacade (id=92) |
| > mail | "nishino@asojuku.ac.jp" (id=94) |
| > password | "pass" (id=96) |
| > loginModel | LoginModel (id=109) |

(x)= 変数 ☒ ブレークポイント

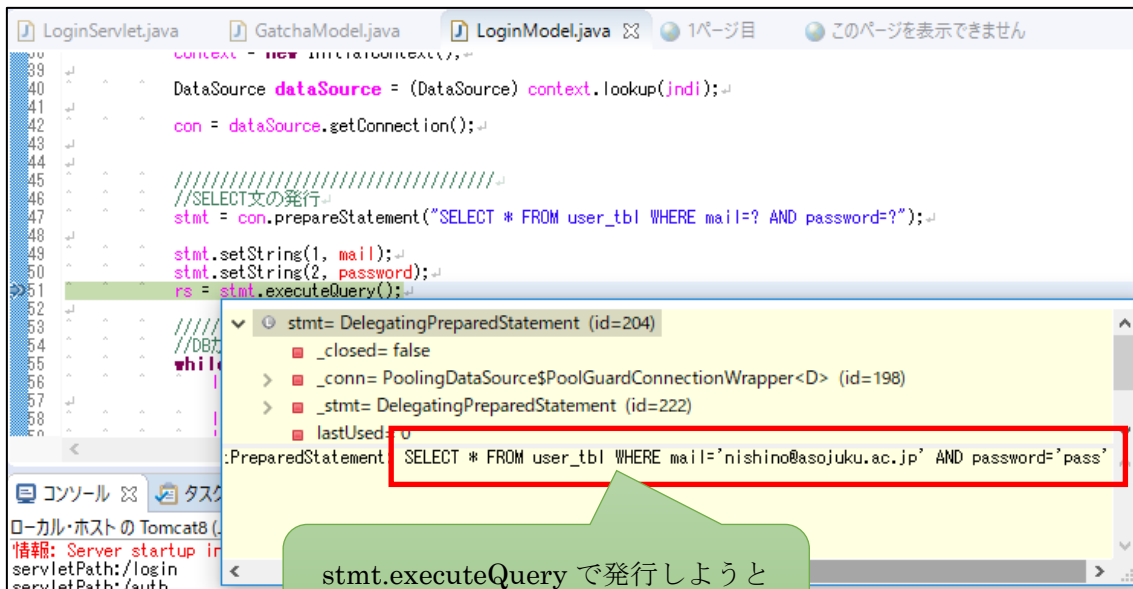
| 名前 | 値 |
|------------------|---------------------------------|
| > this | LoginServlet (id=85) |
| ▼ request | RequestFacade (id=86) |
| ▼ request | Request (id=112) |
| asyncContext | null |
| asyncSupported | Boolean (id=114) |
| attributes | ConcurrentHashMap<K,V> (id=116) |
| authType | null |
| comet | false |
| connector | Connector (id=121) |
| cookies | null |
| cookiesConverted | false |
| cookiesParsed | true |
| coyoteRequest | Request (id=124) |

このウィンドウでは例えば **request** の左にある **>** をクリックすると詳しい中身を見ることができます。

| (x)= 変数 ☒ ブレークポイント | |
|--------------------|---------------------------------|
| 名前 | 値 |
| > this | LoginServlet (id=85) |
| ▼ request | RequestFacade (id=86) |
| ▼ request | Request (id=112) |
| asyncContext | null |
| asyncSupported | Boolean (id=114) |
| attributes | ConcurrentHashMap<K,V> (id=116) |
| authType | null |
| comet | false |
| connector | Connector (id=121) |
| cookies | null |
| cookiesConverted | false |
| cookiesParsed | true |
| coyoteRequest | Request (id=124) |

なお、発行している SQL 文を見たいときは `PreparedStatement` のオブジェクトの中身を見ると確認できます。

↓のような感じです



4. よくある問い合わせ

| 内容 | 確認すること |
|--|---|
| tomcat が起動しません | サーブレットの@WebServlet アノテーションを確認してください。 <ul style="list-style-type: none">・違うサーブレットに同じ URL を割り当ててないか・/ (スラッシュ) が抜けていないか |
| DB に接続ができません！ | context.xml を確認してください。 DB 名が違ったり、パスワードが違っていたりしませんか？ |
| JSP を表示しようとする と 500 エラーが発生し、 NullPointerException と表 示される | JSP 側での NullPointerException は、サーブレットから JSP へ値がうまく渡っていない場合が多いです。 getAttribute など値を取得している部分を見て、サーブレット側で setAttribute している値と同じかどうかを確認してください。 |
| 日本語が文字化けする | EncodingFilter を追加してください ※過去の演習で作っているはずです |