



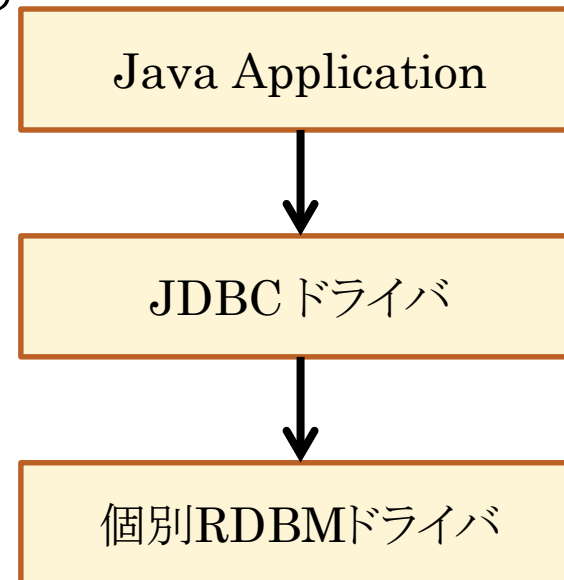
# データベースへの接続

オブジェクト指向プログラミング特論

只木進一:工学系研究科

# JAVAとデータベース

- 関係データベースには様々なものがある
  - Oracle、Postgresql、MySQL、Derby
- 一般にはRDBM毎に専用ドライバが必要
  - プログラムから利用するにはどうする?
- Javaは共通のAPIを提供する
  - 操作が容易
  - 保守性向上



# DBへの接続

- DBをURLとして指定する
  - jdbc:RDMB名://ホスト名:ポート/DB名
- ドライバのロード
  - Class.forName(ドライバ名)
- 接続
  - java.sql.DriverManager.getConnection(URL, ユーザ、パスワード)



# 注意

- 一旦接続すると、**RDBM**に依存しない**API**を利用できる
- 終了時には切断すること
  - `java.sql.Connection.close()`
- **DB**のエラーへの対応
  - 例外処理による
  - `java.sql.SQLException`



# JAVADB(APACHE DERBY)

- Apacheプロジェクトが作成したDB
- Javaと一緒に配布
- ユーザファイルにDBを作成できる
- ローカルのDBを構築できる



# サンプルDBへの接続

```
C:¥Users¥tadaki¥SkyDrive¥ドキュメント¥lecture¥ObjectOrientedProgramming¥DB>"C:¥Program
Files¥Java¥jdk1.8.0_05¥db¥bin¥ij.bat"
```

```
ijバージョン10.10
```

```
ij> connect 'jdbc:derby:C:¥Users¥tadaki¥SkyDrive¥ドキュメント
¥lecture¥ObjectOrientedProgramming¥DB¥dbsample';
```

```
ij> show tables in APP;
```

TABLE_SCHEM	TABLE_NAME	REMARKS
-------------	------------	---------

APP	ROLES	
APP	STAFFS	

2行が選択されました

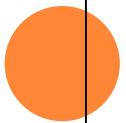
```
ij> select * from roles;
```

ROLE_ID	NAME	DESCRIPTION
---------	------	-------------

1	administrator	全体管理者
2	developer	システム開発者
3	operator	システム運用者
4	user	システム利用者

4行が選択されました

```
ij>
```



# JDBCの利用の注意


- RDBMに応じたjarが必要
- Derbyの場合にはderbyrun.jarが必要



# DBの検索

- select 文に相当する
- select文を文字列sqlに保存しておく
- java.sql.Statementの生成
- 結果はjava.sql.ResultSetへ

```
private ResultSet query(String sql) throws SQLException(  
    Statement select = con.createStatement();  
    ResultSet resultSet=select.executeQuery(sql);  
    return resultSet;  
}
```





## ResultSetからの読み出し

- 注意: 結果が複数であることがある
  - next()メソッドで順に調べる
- ResultSet.getClass名(フィールド名)
- ResultSet.getClass名(フィールド番号)

```
ResultSet r;  
while(r.next()){  
    Integer id = r.getInteger("id");  
    String name = r.getString("name");  
}
```



## 挿入・更新

- insert文、update文
- Statement.executeUpdate(String)を利用
  - 更新件数が戻ってくる

```
private int insert(String sql) throws SQLException{  
    int count;  
    Statement stm;  
    con.setAutoCommit(false);  
    stm = con.createStatement();  
    count = stm.executeUpdate(sql);  
    if(count!=0){ con.commit();  
    } else { con.rollback(); }  
    return count;  
}
```



# DB更新時の注意

- 更新に失敗する可能性を考える
  - 複数テーブルの場合に特に注意
- DBの仮変更の機能を利用
- 自動更新の抑制
  - `setAutoCommit(false)`
- 更新の実行とキャンセル
  - `commit()`
  - `rollback()`



# サンプルプログラムの構成

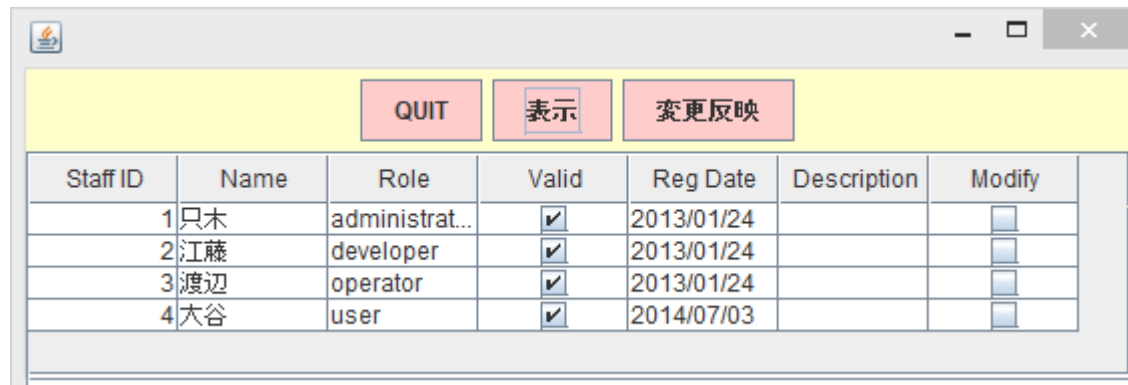
## ○ dataパッケージ

- AbstractDataクラス
  - 特定のRDBに依存しない、DB接続のクラス
- DBクラス
  - AbstractDataクラスの拡張
  - Derbyに接続
- Roleクラス
  - テーブルroleに対応したクラス
- Staffクラス
  - テーブルstaffに対応したクラス



## ○ guiパッケージ

- StaffRecordModelクラス
  - javax.swing.table.AbstractTableModelクラス拡張
  - DBの内容を表形式で表示
  - 編集可能
- DBSampleMainクラス
  - JFrameの拡張



Staff ID	Name	Role	Valid	Reg Date	Description	Modify
1	只木	administrat...	<input checked="" type="checkbox"/>	2013/01/24		<input type="checkbox"/>
2	江藤	developer	<input checked="" type="checkbox"/>	2013/01/24		<input type="checkbox"/>
3	渡辺	operator	<input checked="" type="checkbox"/>	2013/01/24		<input type="checkbox"/>
4	大谷	user	<input checked="" type="checkbox"/>	2014/07/03		<input type="checkbox"/>