



УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

Взаимодействие с базами данных

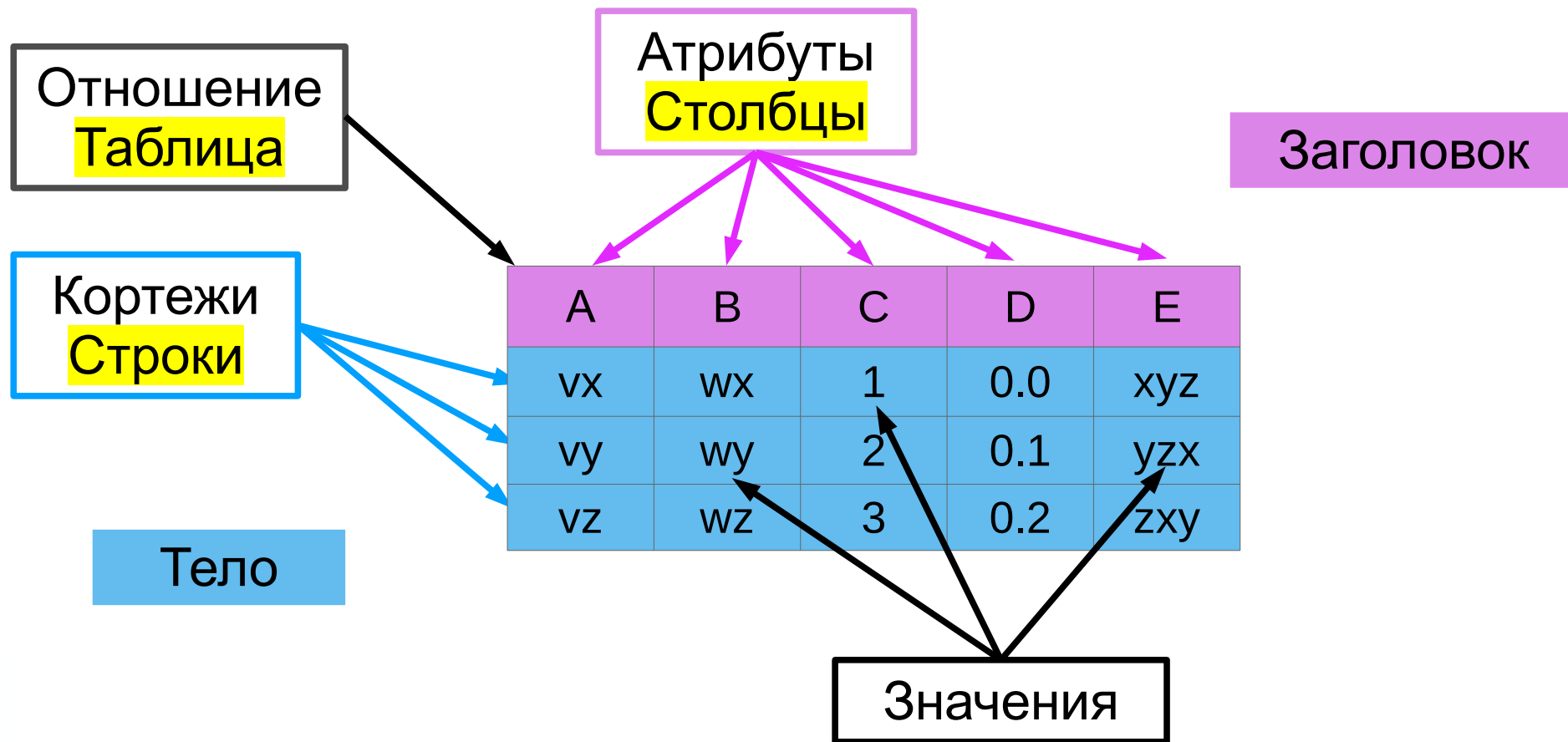
ITMO More than a
UNIVERSITY

- ☑ БД — структурно организованные данные о предметной области, хранящиеся вместе с информацией о данных и их взаимосвязях.

- ☑ БД — структурно организованные данные о предметной области, хранящиеся вместе с информацией о данных и их взаимосвязях.
- ☑ СУБД — вычислительная система для создания и использования баз данных.

- ✓ БД — структурно организованные данные о предметной области, хранящиеся вместе с информацией о данных и их взаимосвязях.
- ✓ СУБД — вычислительная система для создания и использования баз данных.
- ✓ Реляционная БД — база данных, основанная на реляционной модели данных.

- ✓ Отношение (relation) — структура данных, состоящая из заголовка и тела.
 - Заголовок отношения — множество атрибутов
 - Тело отношения — множество кортежей, содержащих значения атрибутов
- ✓ Свойства отношения
 - Каждый атрибут имеет тип, значения соответствуют типу
 - Атрибуты не повторяются, и их порядок не имеет значения
 - Кортежи не повторяются, и их порядок не имеет значения



Пример базы данных

students

student_id	name	group
289001	Иванов Петр	P3110
289002	Петров Сидор	P3130
289999	Сидоров Иван	R3140

groups

group	faculty
P3110	ПИиКТ
P3130	ПИиКТ
R3140	СУиР

grades

student_id	course_id	grade
289001	1	A
289001	2	B
289002	1	E
289002	2	A
289999	1	C
288999	3	D

courses

course_id	name	semester	type
1	Программирование	1	З
2	ОПД	2	Э
3	Физика	2	Э

- ☑ SQL (Structured Query Language) — декларативный язык для описания, изменения и получения данных из реляционных баз данных.

☑ helios

- `psql -h pg studs`
- пароль в файле `.pgpass`
 - ◆ `*:*:*:login:PaSsw0rD`

☑ <https://postgresql.org>

☑ `\?` - помощь по командам `psql`

☑ `\h` - помощь по командам SQL

✓ DDL — Data definition language

```
CREATE TABLE weather (  
    city        VARCHAR(80),  
    temp_lo     INT,          -- low temperature  
    temp_hi     INT,          -- high temperature  
    prcp        REAL,         -- precipitation  
    date        DATE  
);
```

```
CREATE TABLE IF NOT EXISTS persons
```

```
DROP TABLE weather;
```

☑ Ограничения

- Типы данных
 - ◆ INT, SMALLINT, REAL, DOUBLE, CHAR(n), VARCHAR(n), DATE, TIME, ...
- Возможные значения в столбце
 - ◆ NOT NULL, UNIQUE, CHECK (age >= 18)
- Ключи
 - ◆ Первичный ключ
 - PRIMARY KEY (UNIQUE, NOT NULL)
 - ◆ Внешние ключи
 - FOREIGN KEY (REFERENCES)

☑ DML — Data manipulation language

- INSERT
- UPDATE
- DELETE

☑ Вставка данных в таблицу

```
INSERT INTO weather VALUES ('Oslo', 46, 50, 0.25, '2021-11-27');
```

```
INSERT INTO weather (date, city, temp_hi, temp_lo)  
VALUES ('2021-11-29', 'Helsinki', 54, 37);
```

```
COPY persons TO file;
```

```
COPY persons FROM file;
```

☑ Обновление данных в таблице

```
UPDATE weather  
  SET temp_hi = temp_hi - 2,  temp_lo = temp_lo - 2  
 WHERE date > '2021-11-28';
```

☑ Удаление данных из таблицы

```
DELETE FROM weather WHERE city = 'Oslo';
```

```
DELETE FROM persons;
```

☑ Выборка данных - запрос

```
SELECT * FROM students;  
SELECT name, group FROM students;  
SELECT * FROM students WHERE group = 'P3110';  
SELECT * FROM students ORDER BY name;  
SELECT DISTINCT type FROM courses;  
SELECT COUNT(*) FROM students;
```


☑ Соединение таблиц

```
SELECT name, faculty FROM students JOIN groups
      ON students.group = groups.group;  -- USING(group);
```

students

student_id	name	group
289001	Иванов Петр	P3110
289002	Петров Сидор	P3130
289999	Сидоров Иван	R3140

groups

group	faculty
P3110	ПИиКТ
P3130	ПИиКТ
R3140	СУиР

JOIN

student_id	name	group	faculty
289001	Иванов Петр	P3110	ПИиКТ
289002	Петров Сидор	P3130	ПИиКТ
289999	Сидоров Иван	R3140	СУиР



УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

ОСНОВЫ JDBC

```
Compiled from "Hello.java" public class Hello {
  minor version: 0 major version: 52
  flags: ACC_PUBLIC ACC_SUPER
  Constant pool: #1 = Methodref #6.#15 // java/lang/Object.<init>:()V
               #16.#17 // java/lang/System.out:Ljava/io/PrintStream;
               #3 = String #18 // Hello world!
               #4 = Methodref #19.#20 // java/io/PrintStream.println:(Ljava/lang/String;)V
               #5 = Class #21 // Hello
               #6 = Class #22 // java/lang/Object
               #7 = Utf8 <init>
               #8 = Utf8 ()V
               #9 = Utf8 Code
               #10 = Utf8 LineNumberTable
               #11 = Utf8 main
               #12 = Utf8 ([Ljava/lang/String;)V
               #13 = Utf8 SourceFile
               #14 = Utf8 Hello.java
               #15 = NameAndType #7:#8 // <init>:()V
               #16 = Class #23 // java/lang/System.out:Ljava/io/PrintStream;
               #17 = Utf8 Hello world!
               #18 = Class #26 // java/io/PrintStream
               #19 = NameAndType #27:#28 // println:(Ljava/lang/String;)V
               #20 = Utf8 java/lang/Object
               #21 = Utf8 java/lang/System
               #22 = Utf8 out
               #23 = Utf8 Ljava/io/PrintStream;
               #24 = Utf8 java/io/PrintStream
               #25 = Utf8 println
               #26 = Utf8 (Ljava/lang/String;)V
               #27 = Utf8 public
               #28 = Utf8 Hello()
               descriptor: ()V
  flags: ACC_PUBLIC Code: 0: aload_0 1: invokespecial #1 // Method java/lang/Object.<init>:()V 4: return
  LineNumberTable: line 1: 0 public static void main(java.lang.String[]);
  descriptor: ([Ljava/lang/String;)V flags: ACC_PUBLIC, ACC_STATIC Code:
  stack=2, locals=1, args_size=1 0: getstatic #2 // Field java/lang/System.out:Ljava/io/PrintStream;
  3: ldc #3 // String Hello world!
  5: invokevirtual #4 // Method java/io/PrintStream.println:(Ljava/lang/String;)V
  8: return
  SourceFile: "Hello.java"
  LineNumberTable: line 3: 0 line 4: 8 }
```



- ☑ Много разных СУБД
 - API для каждой базы отдельно

☑ Много разных СУБД

- API для каждой базы отдельно
- единый интерфейс работы с базами + драйвер для конкретной базы

☑ Много разных СУБД

- API для каждой базы отдельно
- единый интерфейс работы с базами + драйвер для конкретной базы

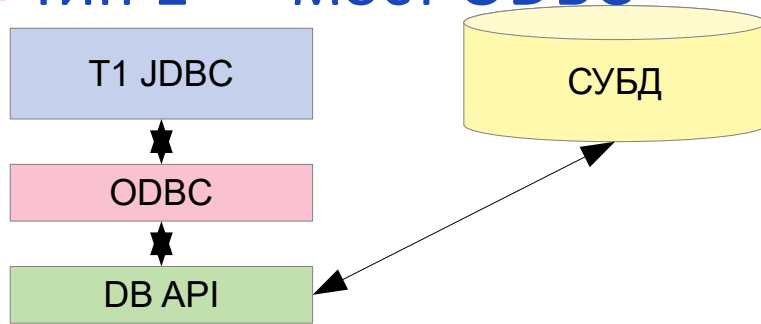
☑ Реализации

- ODBC — Open Database Connectivity
- JDBC — Java Database Connectivity

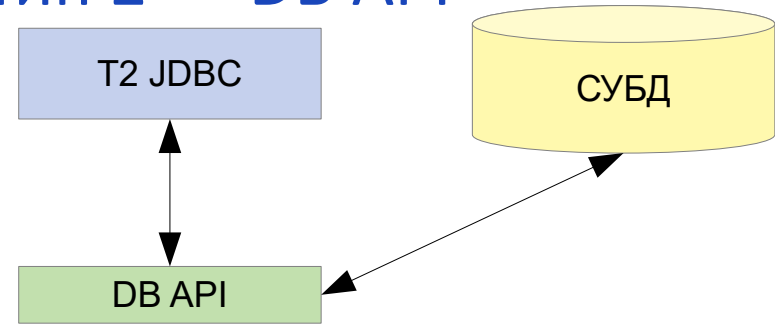
- ✓ JDBC — Java DataBase Connectivity
- ✓ JDBC API — высокоуровневый интерфейс для доступа к данным
- ✓ JDBC Driver API — низкоуровневый интерфейс для драйверов
- ✓ Пакеты `java.sql` (Core) и `javax.sql` (Extension)
- ✓ Стандарт взаимодействия с СУБД

Типы взаимодействия

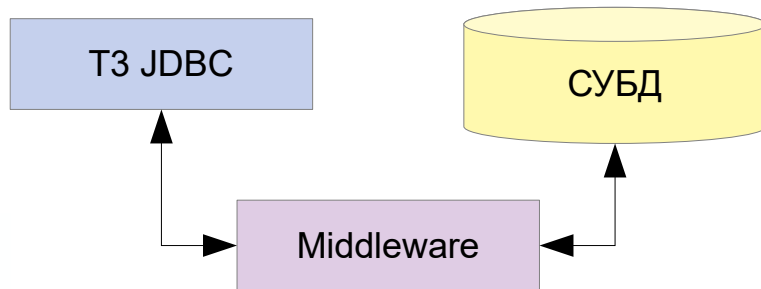
✓ Тип 1 — мост ODBC



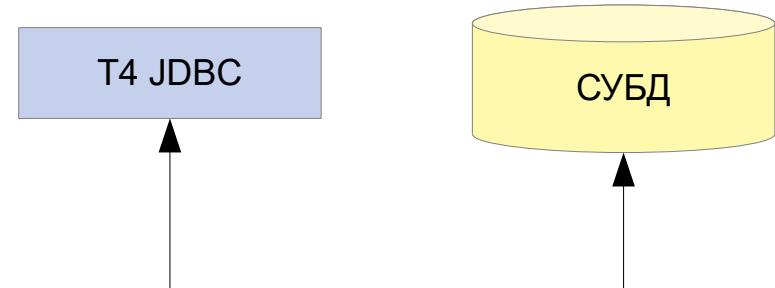
✓ Тип 2 — DB API



✓ Тип 3 — middleware



✓ Тип 4 — pure Java



```
Connection conn = DriverManager.getConnection( ... );  
  
Statement stat = conn.createStatement();  
  
ResultSet res = stat.executeQuery("SELECT ... ");  
  
while (res.next()) {  
    // получение и обработка данных  
}  
  
res.close();  
stat.close();  
conn.close();
```


☑ java.sql.Driver

- Отвечает за связь с БД
- Метод Connection connect(String url, Properties info)
- Используется для написания драйверов для СУБД

- ✓ Управляет списком драйверов
- ✓ Загрузка драйвера
 - `Class.forName()`
 - `jdbc.drivers=`
- ✓ Неявная загрузка с помощью `ServiceLoader`
 - `META-INF/services/java.sql.Driver`

- ☑ Метод `Connection getConnection(String url, ...)`
 - `url = jdbc:protocol://host:port/database`
 - ◆ `jdbc:postgresql://db:5432/studs`
 - `getConnection(String url, Properties info)`
 - ◆ `Properties info = new Properties();`
 - ◆ `info.load(new FileInputStream("db.cfg"));`
файл `db.cfg`

```
user = s999999
password = sss999
```
 - `getConnection(String url, String name, String pass)`

☑ Абстракция соединения (сессия)

- методы:

- ◆ `Statement` `createStatement()`
- ◆ `PreparedStatement` `prepareStatement(String sql)`
- ◆ `CallableStatement` `prepareCall(String sql)`
- ◆ `DatabaseMetaData` `getMetaData()`

☑ Statement

- Статический SQL-запрос
- `Statement st = connection.createStatement();`
`st.executeQuery("SELECT * FROM table WHERE id = 15");`

☑ **PreparedStatement** (extends Statement)

- Динамический запрос с параметрами

```
ps = preparedStatement("SELECT * FROM table WHERE id = ?");
```

```
ps.setInt(1, 15); // 1 — номер параметра, 15 — значение
```

```
SELECT * FROM table WHERE id = 15
```

☑ Предотвращает SQL-инъекции

☑ CallableStatement (extends PreparedStatement)

- Вызов хранимой процедуры

SQL: CREATE PROCEDURE

```
cs = prepareCall("CALL getResult (?)");
```

```
cs.setInt(1, 15);
```

```
cs.registerOutParameter(1, Types.INTEGER);
```

...

```
int result = cs.getInt(1);
```

```
CALL getResult(15);
```



☑ **ResultSet** executeQuery(String sql)

- для исполнения команды SELECT
- Возвращает ResultSet

☑ String sql

- запрос для Statement
- отсутствует для PreparedStatement и CallableStatement

✓ **ResultSet** executeQuery(String sql)

- для исполнения команды SELECT
- Возвращает ResultSet

✓ **int** executeUpdate(String sql)

- для выполнения запросов INSERT, UPDATE, DELETE
- возвращает количество измененных строк
- Для команд DDL возвращает 0

☑ boolean execute(String sql)

- для выполнения любых запросов
- **true**, если результат — ResultSet : **ResultSet** getResultSet()
- **false**, если результат — updateCount : **int** getUpdateCount()

☑ Connection

- `setAutoCommit(true/false)`
- `commit()`
- `rollback()`
- `setSavepoint()`

☑ Connection

- `setAutoCommit(true/false)`
- `commit()`
- `rollback()`
- `setSavepoint()`

☑ Statement

- `addBatch(String sql)`
- `clearBatch()`
- `executeBatch()`

☑ Получение данных из ResultSet

```
ResultSet rs = preparedStatement.executeQuery();
```

```
while (rs.next()) {  
    String name = rs.getString(1); // по номеру столбца  
    int id = rs.getInt("id");      // по имени столбца  
}
```

☑️ ResultSet

- Connection.**createStatement**(sql, **type**, concurrency, holdability)
- ResultSetType
 - ◆ **TYPE_FORWARD_ONLY**
 - ◆ TYPE_SCROLL_INSENSITIVE
 - ◆ TYPE_SCROLL_SENSITIVE
- ResultSetConcurrency
 - ◆ **CONCUR_READ_ONLY**
 - ◆ CONCUR_UPDATABLE
- ResultSetHoldability
 - ◆ HOLD_CURSORS_OVER_COMMIT
 - ◆ CLOSE_CURSORS_AT_COMMIT

☑ Навигация

- next()
- previous()
- first()
- last()
- beforeFirst()
- afterLast()
- relative(int row)
- absolute(int row)
- moveToInsertRow()

☑ Получение данных

- getString(int)
- getString(String)
- getInt(int)
- getInt(String)
- getBoolean
- getLong
- getDouble
- getArray (SQL Array)
- getDate
- getTimestamp
- getReader
- ...

☑ Обновление строк

- `updateInt(String, int)`
- `updateInt(int, int)`
- `updateString(String, String)`
- `updateString(int, String)`
- `updateRow()`

☑ Добавление строк

- `moveToInsertRow()`
- `updateInt(String, int)`
- `insertRow()`

☑ ResultSetMetaData ResultSet.getMetaData()

- `getTableName()`
- `getColumnCount()`
- `getColumnName(int n)`
- `getColumnType(int n)`

☑ DatabaseMetaData Connection.getMetaData()

- `getCatalogs()`
- `getTables()`
- `getSchemas()`



УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

Расширения JDBC



☑ `javax.sql.DataSource`

- Позволяет получить соединение с БД
- `org.postgresql.ds.PGSimpleDataSource`

☑ `ConnectionPoolDataSource`

- Поддержка виртуального пула соединений
- `org.postgresql.ds.PGPoolingDataSource`

☑ `XADataSource`

- Поддержка распределенных транзакций

```
import org.postgresql.ds.PGSimpleDataSource;  
PGSimpleDataSource ds = new PGSimpleDataSource();  
ds.setServerName(...);  
ds.setDatabaseName(...);  
ds.setUser(...);  
ds.setPassword(...);  
Connection conn = ds.getConnection();
```

```
import javax.naming.*;
```

```
Context ctx = new InitialContext();
```

```
DataSource ds = ...
```

```
ctx.bind("testDB", ds);
```

```
Context ctx = new InitialContext();
```

```
DataSource ds = (DataSource)ctx.lookup("testDB");
```



- ☑ `javax.rowset.*`
- ☑ Единый интерфейс для всех операций
- ☑ RowSet extends ResultSet
 - ◆ `setUrl()`,
 - ◆ `,`
 - ◆ `setPassword()`,
 - ◆ `setCommand("Select * from ...");`
 - `execute()`
 - `next()`
 - `getXXX()`
- ☑ `RowSetFactory factory = RowSetProvider.newFactory();`
`factory.createJdbcRowSet();`

☑ RowSetFactory factory = RowSetProvider.newFactory();

- factory.createJdbcRowSet();
- factory.createCachedRowSet();
- factory.createWebRowSet();
- factory.createFilteredRowSet();
- factory.createJoinRowSet();

☑ **JdbcRowSet** — простая разновидность RowSet

- Поддерживает соединение с базой данных
- По умолчанию:
 - ◆ Тип: SCROLL_INSENSITIVE
 - ◆ Конкурентность: CONCUR_UPDATABLE
 - ◆ Включено экранирование спецсимволов

☑ **JdbcRowSet** — простая разновидность RowSet

```
JdbcRowSet rs = factory.createJdbcRowSet();  
rs.setUrl(""); ...  
rs.setCommand("Select * from users");  
rs.execute();  
rs.last();  
rs.getInt("id");  
rs.updateString("name", "Pupkin");  
rs.UpdateRow();
```

☑ CachedRowSet

- Результат запроса может кэшироваться
- Синхронизация с базой
- Разрешение конфликтов
- `acceptChanges()`

☑ CachedRowSet

```
CachedRowSet rs = factory.createCachedRowSet();  
rs.setUrl(""); ...  
rs.setCommand("Select * from users");  
rs.execute();  
rs.last();  
rs.getInt("id");  
rs.updateString("name", "Pupkin");  
rs.acceptChanges();
```

☑ WebRowSet

- Может записывать и читать результат в виде XML
- writeXML()
- readXML()

☑ WebRowSet

```
WebRowSet rs = factory.createWebRowSet();  
rs.setUrl(""); ...  
rs.setCommand("Select * from users");  
rs.execute();  
rs.writeXML(new FileWriter("data.xml"));
```

☑ FilteredRowSet

- Фильтрация строк (аналог WHERE)
- `setFilter(Predicate p)`

☑ Predicate

- `boolean evaluate(Object value, int/String column)`
- `boolean evaluate(RowSet rowset)`

☑ FilteredRowSet

```
FilteredRowSet rs = factory.createFilteredRowSet();  
rs.setUrl(""); ...  
rs.setCommand("Select * from users");  
rs.execute();  
Predicate filter = new Predicate();  
rs.setFilter(filter);
```


☑ JoinRowSet

- Соединение нескольких результатов (JOIN)
- addRowSet()
- setJoinType()
- toCachedRowSet()

☑ JoinRowSet

```
JoinRowSet js = factory.createJoinRowSet();  
CachedRowSet users; // "Select * from users"  
CachedRowSet groups; // "Select * from groups"  
users.setMatchColumn("uid");  
groups.setMatchColumn("uid");  
js.addRowSet(users);  
js.addRowSet(groups);
```



УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

Провайдеры служб



- ✓ Служба или сервис
- ✓ Разные реализации
- ✓ Нужен механизм поиска и загрузки служб

```
class CheatSheet {  
    public String getAnswer(String question) {  
        Map source =  
        String answer = source.get(question);  
        if (answer != null) return answer;  
        else return "Epic fail";  
    }  
}
```

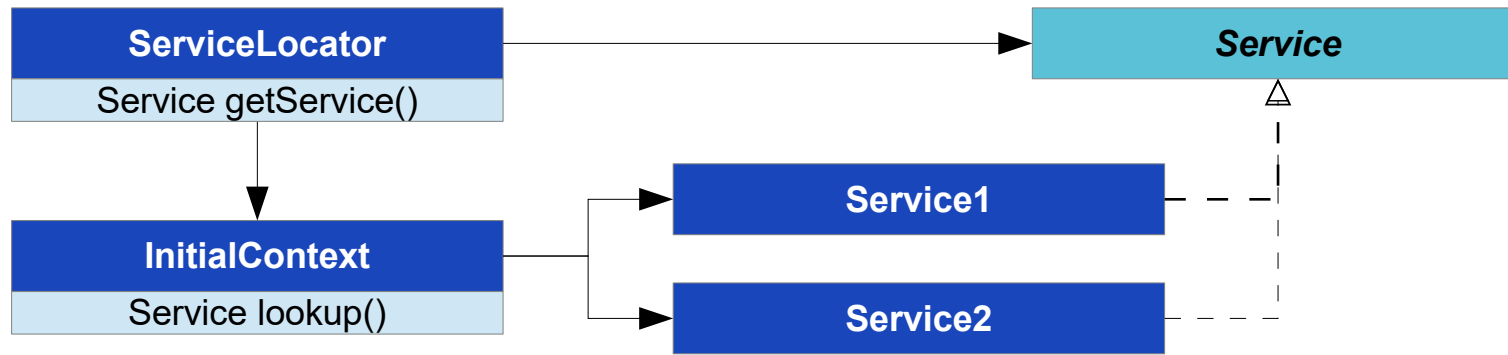
```
class CheatSheet {  
    public String getAnswer(String question) {  
        Map source = knowledge.getSource();  
        String answer = source.get(question);  
        if (answer != null) return answer;  
        else return "Epic fail";  
    }  
}  
  
interface Knowledge {  
    Map<String,String> getSource();  
}
```

```
class CheatSheet {  
    public String getAnswer(String question) {  
        Map source = knowledge.getSource();  
        String answer = source.get(question);  
        if (answer != null) return answer;  
        else return "Epic fail";  
    }  
}  
  
interface Knowledge {  
    Map<String,String> getSource();  
}
```

```
class Memory implements Knowledge  
class Lecture implements Knowledge  
class CallFriend implements Knowledge  
class Magic implements Knowledge
```

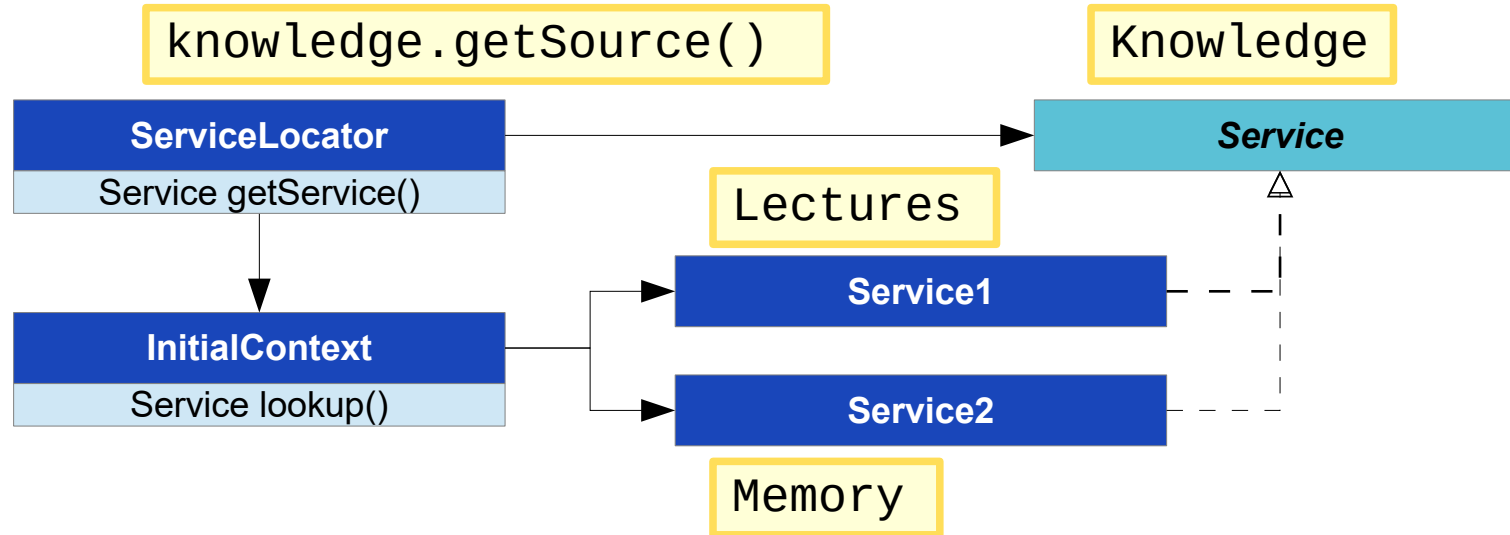
Как передать объект?

✓ Шаблон ServiceLocator



Как передать объект?

✓ Шаблон ServiceLocator



- ☑ Класс `java.util.ServiceLoader<Service>`
 - `static ServiceLoader<Service> load(Service.class)`
 - `Iterator<Service> iterator()`

✓ interface `spi.Service`

- `execute()`

✓ class `spi.DefaultServiceImpl`

- `public DefaultServiceImpl()`

✓ `service.jar`

- `META-INF/services/`

- ◆ `spi.Service`

- `spi.DefaultServiceImpl`

- ✓ `java.nio.file.spi.FileSystemProvider`
- ✓ `java.nio.channels.spi.AsynchronousChannelProvider`
- ✓ `java.nio.channels.spi.SelectorProvider`
- ✓ `java.nio.charset.spi.CharsetProvider`
- ✓ `java.text.spi.DateTimeFormatProvider`
- ✓ `java.text.spi.NumberFormatProvider`
- ✓ `java.util.spi.CalendarDataProvider`
- ✓ `java.sql.DriverManager`





УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

Безопасное хранение паролей

ITMO More than a
UNIVERSITY

✓ CREATE TABLE users (varchar name, varchar password);

Пароль нельзя хранить в открытом виде

- Доступ к базе
- Одинаковые пароли
- Резервное копирование
- SQL-инъекции и другие методы

```
CREATE TABLE users (varchar name, varchar password);  
CREATE TABLE cats (varchar name, varchar owner);  
stmt = "SELECT name, owner FROM cats where owner = ";  
ownerName = Scanner.readLine();  
createStatement(stmt + "'" + ownerName + "'");  
execute();
```



```
CREATE TABLE users (varchar name, varchar password);  
CREATE TABLE cats (varchar name, varchar owner);  
stmt = "SELECT name, owner FROM cats where owner = ";  
ownerName = Scanner.readLine();  
createStatement(stmt + "'" + ownerName + "'");  
execute();
```

Simon

```
SELECT name, owner FROM cats where owner = 'Simon'
```




```
CREATE TABLE users (varchar name, varchar password);  
CREATE TABLE cats (varchar name, varchar owner);  
stmt = "SELECT name, owner FROM cats where owner = ";  
ownerName = Scanner.readLine();  
createStatement(stmt + "' " + ownerName + "'");  
execute();
```

' union select name, password from users where name <> '

```
SELECT name, owner FROM cats where owner = '  
union select name, password from users where name <> ''
```

Лучше хранить хеш

- ✓ Вместо пароля — хеш пароля
- ✓ Функция **необратимая**
- ✓ Минимальное число **коллизий**
- ✓ MD4, MD5, SHA-1, SHA-2
- ✓ `md5("hello") = 5d41402abc4b2a76b9719d911017c592`

- ✓ Вместо пароля — хеш пароля
- ✓ Функция **необратимая** — **но можно подобрать**
- ✓ Минимальное число **коллизий** — **тоже можно подобрать**
- ✓ MD4, MD5, SHA-1, SHA-2
- ✓ $\text{md5}(\text{"hello"}) = 5\text{d}41402\text{abc}4\text{b}2\text{a}76\text{b}9719\text{d}911017\text{c}592$
- ✓ Словарные атаки

- ✓ Вместо пароля — хеш пароля
- ✓ Функция **необратимая** — **но можно подобрать**
- ✓ Минимальное число **коллизий** — **тоже можно подобрать**
- ✓ MD4, MD5, SHA-1, SHA-2
- ✓ $\text{md5}(\text{"hello"}) = 5\text{d}41402\text{abc}4\text{b}2\text{a}76\text{b}9719\text{d}911017\text{c}592$
- ✓ Словарные атаки

<https://md5.gromweb.com/>

<https://www.md5hashgenerator.com/>

Лучше хранить хеш с солью

- ✓ Вместо пароля — хеш пароля + соль
- ✓ Соль — некая случайная последовательность

`md5("hello" + "$!ns50D") = fcac10c41f9f67090e45010`

`md5("hello" + "H0xc3@") = a893f9063e4314635e05f8c46b547f7a`



Лучше хранить хеш с солью

- ✓ Вместо пароля — хеш пароля + соль
- ✓ Соль — некая случайная последовательность



`md5("hello" + "$!ns50D") = fcac10c41f9f67090e4501e`

`md5("hello" + "HOxc3@") = a893f9063e4314635e05f8c46b547f7a`

- ✓ Храним пароль + соль + алгоритм + сложность
- ✓ Все равно — "hello", "12345" — словарные атаки

Добавим перец

- ✓ В базе храним хеш пароля + соль
- ✓ Перец храним в приложении (отдельно)
- ✓ `md5("kFz<Q%ps" + "hello" + "$Ins50D")`



```
java.security.MessageDigest
```

```
MessageDigest md = MessageDigest.getInstance("MD5");  
String user      = Console.readLine()  
String passwd    = Console.readPassword();  
String salt      = getRandomString();  
String pepper    = "*63&^mVLC(#"  
byte[] hash = md.digest(  
    (pepper + passwd + salt).getBytes("UTF-8"));  
  
insert into users (user, salt, hash);
```



- ☑ Входящие пароли - хеш, соль
- ☑ Исходящие пароли - отдельно от кода!
 - Запрашивать при старте
 - Конфигурационный файл с ограничением прав
 - Файлы свойств (Properties)
 - Переменная окружения



УНИВЕРСИТЕТ ИТМО

Программирование. 2 семестр

Дата и время

ITMO More than a
UNIVERSITY

- ☑ Дата и время — 2 варианта представления:
 - Человеческое время — часы, минуты, дни, недели, месяцы
 - Машинное время — миллисекунды от нулевой точки отсчета
 - ◆ 1 января 1970 года, 00:00:00

☑ Date

- в версии 1.0 — единственный класс даты
- человеческое и машинное представление
- форматирование даты
- версия 1.1 — Date — момент времени
- почти все методы объявлены deprecated


☑ Конструкторы

- Date
- Date(long)

☑ Методы

- long getTime()
- boolean after(Date)
- boolean before(Date)

- ☑ Временная зона — смещение от стандартного:
- ☑ до 1972 года - Гринвич (GMT)
- ☑ после 1972 — UTC — всемирное координированное
- ☑ Методы
 - getDefault()
 - getAvailableIDs()
 - getRawOffset() - смещение без учета летнего времени
 - getOffset(long date) — с учетом летнего времени

- ☑  Класс SimpleTimeZone — реализованный потомок

☑ Абстрактный класс — преобразование из машинных в человеческие единицы

- Calendar getInstance()
- add(int field, int amount);
- roll(int field, int amount);
- set(int field, int value);
- Date getTime()
- setTime(Date)

☑ реализованный класс GregorianCalendar

- сочетает 2 календаря (григорианский и юлианский)

- ☑ `java.time` - дата, время, периоды
 - `Instant`, `Duration`, `Period`, `LocalDate`, `LocalTime`, `LocalDateTime`, `OffsetTime`, `OffsetDateTime`, `ZonedDateTime`
- ☑ `java.time.chrono` - календарные системы
- ☑ `java.time.format` - форматирование даты и времени
- ☑ `java.time.temporal` - единицы измерения и отдельные поля
- ☑ `java.time.zone` - временные зоны и правила

Дни недели и месяцы (enums)

- ✓ enum DayOfWeek (1 (MONDAY) — 7 (SUNDAY))
- ✓ enum Month (1 (JANUARY) — 12 (DECEMBER))
- ✓ метод `getDisplayName(style, locale)`
- ✓ стиль — FULL, NARROW, SHORT / STANDALONE

- ✓ Year
- ✓ YearMonth
- ✓ MonthDay
- ✓ LocalDate
- ✓ LocalTime
- ✓ LocalDateTime

☑ Статические

- of — создает экземпляр на основе входных параметров
 - ◆ `LocalDate.of(year, month, day)`, `ofYearDay(year, dayOfYear)`
- from — конвертирует экземпляр из другого типа
 - ◆ `LocalDate.from(LocalDateTime)`
- parse — создает экземпляр из строкового представления
 - ◆ `LocalDate.parse("2022-02-22")`

☑ Методы экземпляра

- `format` — форматирует объект в строку
- `get` — возвращает поля объекта // `getHours()`
- `with` — возвращает копию с изменением // `withYear(2021)`
- `plus` — возвращает копию с добавлением // `plusDays(2)`
- `minus` — возвращает копию с убавлением // `minusWeeks(3)`
- `to` — преобразует объект в другой тип // `toLocalTime()`
- `at` — комбинирует объект с другим // `date.atTime(time)`

- ☑ Zoneld — идентификатор зоны
 - Europe/Moscow
- ☑ ZoneOffset — разница со стандартным временем
 - UTC+01:00, GMT-2
- ☑ $\text{OffsetTime} = \text{LocalTime} + \text{ZoneOffset}$
- ☑ $\text{OffsetDateTime} = \text{LocalDateTime} + \text{ZoneOffset}$
- ☑ $\text{ZonedDateTime} = \text{LocalDateTime} + \text{Zoneld}$
 - использует `java.time.zone.ZoneRules`

☑ Класс Instant

- now()
- plusNanos()
- plusMillis()
- plusSeconds()
- minusNanos()
- minusMillis()
- minusSeconds()

☑ java.time.format.DateTimeFormatter

- формат можно выбрать из констант:
 - ◆ BASIC_ISO_DATE
 - ◆ ISO_DATE/TIME/DATETIME
 - ◆ ISO_LOCAL_DATE/TIME/DATETIME
 - ◆ ISO_OFFSET_DATE/TIME/DATETIME
 - ◆ ISO_ZONED_DATETIME
 - ◆ ISO_INSTANT
- задать шаблон
 - ◆ ofPattern()
- методы format() и parse()

- ✓ **Duration** — продолжительность в часах и менее
 - `toNanos()`, `toMillis()`, `toSeconds()`, `toMinutes()`, `toHours()`, `toDays()`
- ✓ **Period** — период в днях и более
 - `getDays()`, `getMonths()`, `getYears()`
- ✓ `.between()`
- ✓ `.plus`
- ✓ `.minus`

Соответствия:

- Date — Instant
- GregorianCalendar — ZonedDateTime
- TimeZone — ZoneId, ZoneOffset

☑ Методы:

- Calendar.toInstant()
- GregorianCalendar.toZonedDateTime()
- GregorianCalendar.fromZonedDateTime()
- Date.fromInstant()
- Date.toInstant()
- TimeZone.toZoneId()