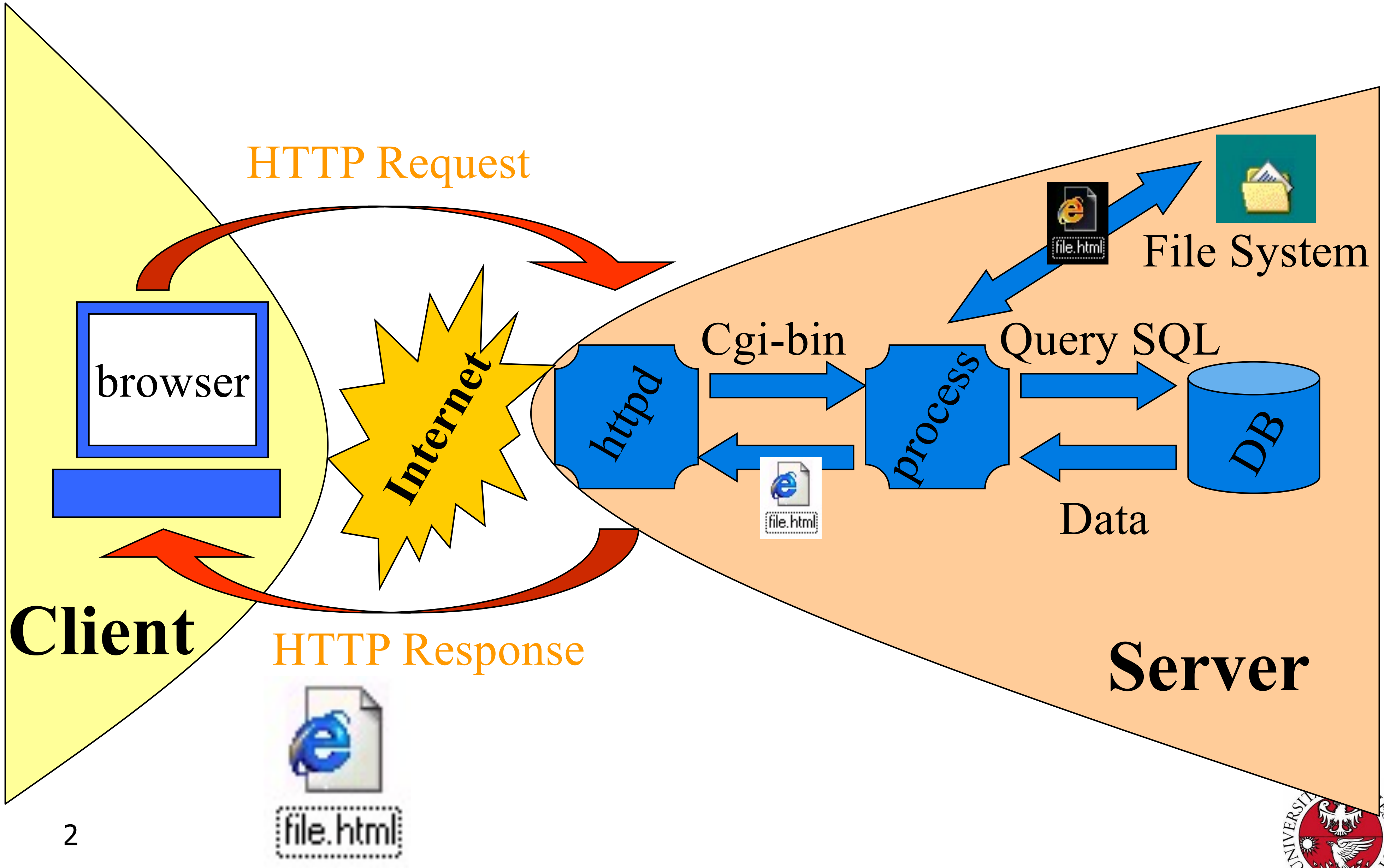
Access to DB

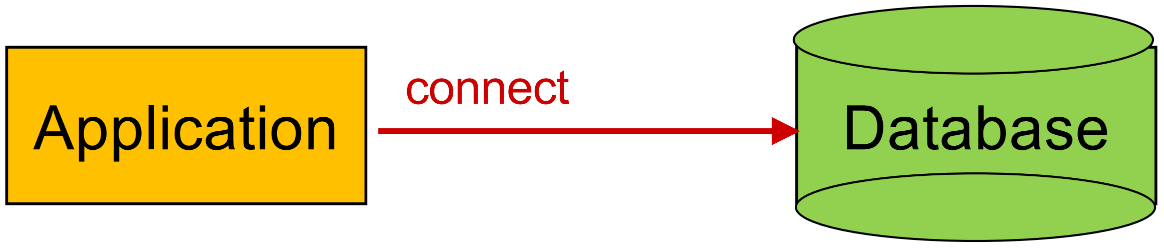
Using JDBC in servlets

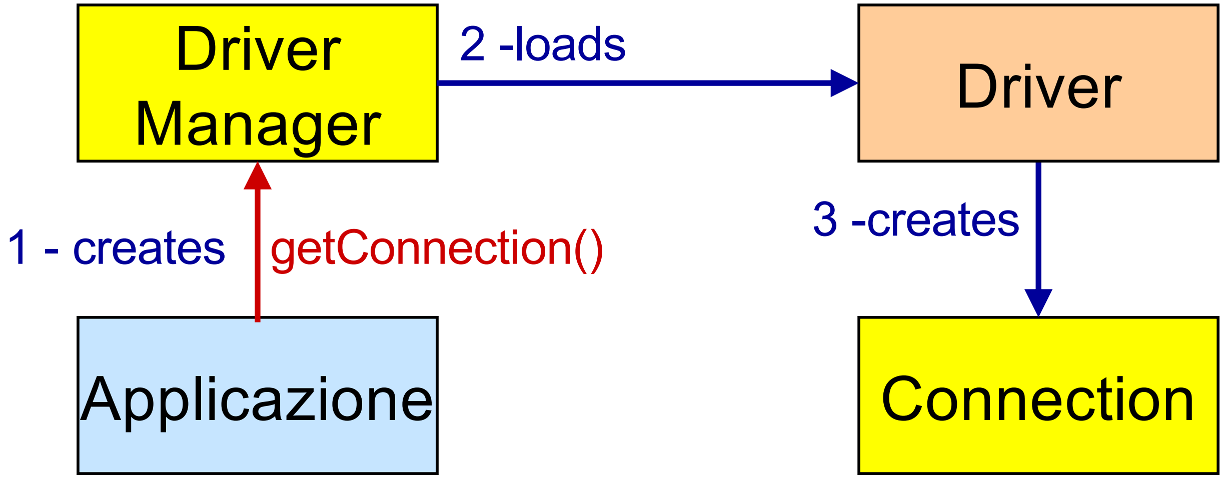

HTTP Request
browser
Internet
httpd
Cgi-bin
process
Query SQL
DB
Data
File System
file.html
file.html
Client
HTTP Response
file.html
Server

Q

How do I access a database from Java?

The java.sql Object Model





Reminder: Class.forName

static Class **forName**(String className)

Returns the Class object associated with the class or interface with the given string name.

Typical use:

```
Object o=Class.forName("java.lang.String").newInstance();
```

is equivalent to:

```
Object o=new String();
```

JDBC – Steps – 1 – Get the driver

```
Class.forName("org.apache.derby.jdbc.ClientDriver");
```

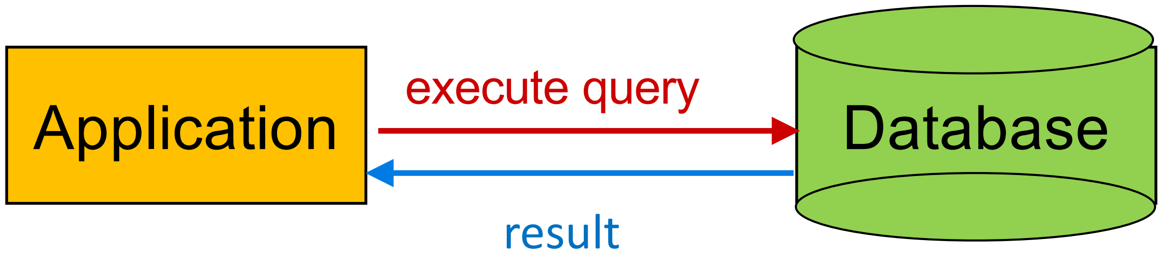
JDBC – Steps – 2 - LOAD THE DRIVER

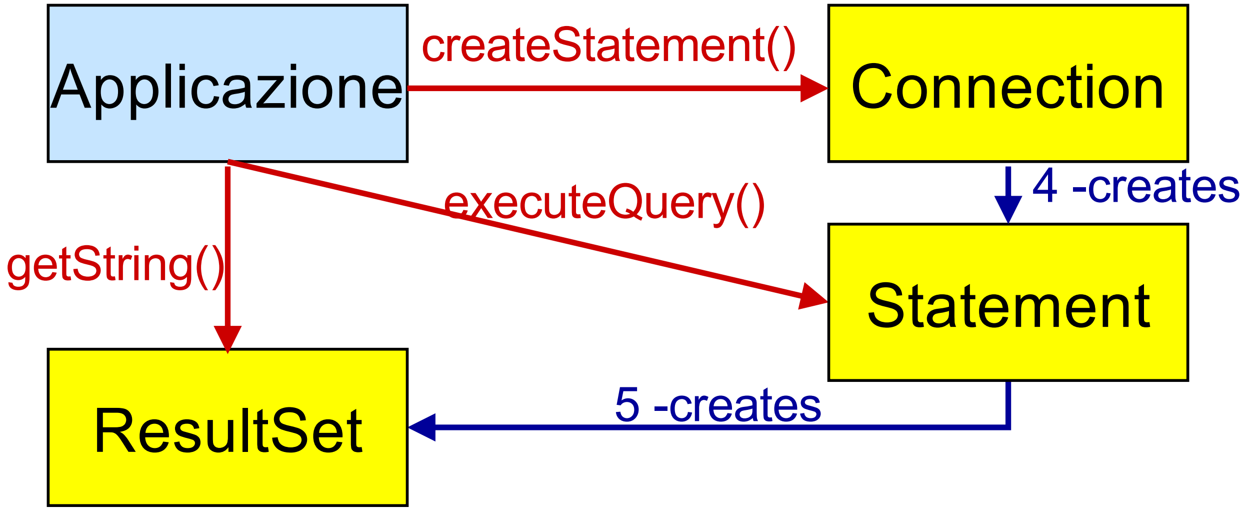
```
Connection con = DriverManager.getConnection(url, "myLogin",
"myPassword");
```

If you are using a JDBC driver developed by a third party, the documentation will tell you what subprotocol to use, that is, what to put after jdbc: in the JDBC URL. For example, if the driver developer has registered the name acme as the subprotocol, the first and second parts of the JDBC URL will be jdbc:acme: . The driver documentation will also give you guidelines for the rest of the JDBC URL. This last part of the JDBC URL supplies information for identifying the data source.

example: String dbURL = "jdbc:derby://localhost:1527/DemoDB";

The java.sql Object Model





JDBC – Steps – 3 CREATE STATEMENT

A Statement object is what sends your SQL statement to the DBMS.

For a SELECT statement, the method to use is executeQuery .
For statements that create or modify tables, the method to use is executeUpdate.

```
Statement stmt = con.createStatement();
stmt.executeUpdate("CREATE TABLE COFFEES " +
    "(COF_NAME VARCHAR(32), SUP_ID INTEGER, PRICE FLOAT, " +
    "SALES INTEGER, TOTAL INTEGER)");
```

Typically you would put the SQL statement in a String (called let's say createTableCoffees), and then use

```
stmt.executeUpdate(createTableCoffees);
```

JDBC – Steps – 4 RETRIEVING VALUES

JDBC returns results in a ResultSet object.

```
ResultSet rs = stmt.executeQuery( "SELECT COF_NAME, PRICE FROM COFFEES");
```

In order to access the names and prices, we will go to each row and retrieve the values according to their types. The method next moves what is called a cursor to the next row and makes that row (called the current row) the one upon which we can operate. Since the cursor is initially positioned just above the first row of a ResultSet object, the first call to the method next moves the cursor to the first row and makes it the current row. Successive invocations of the method next move the cursor down one row at a time from top to bottom. Note that with the JDBC 2.0 API, you can move the cursor backwards, to specific positions, and to positions relative to the current row in addition to moving the curs or forward.

```
String query = "SELECT COF_NAME, PRICE FROM COFFEES"; ResultSet rs = stmt.executeQuery(query);
while (rs.next()) {
		String s = rs.getString("COF_NAME");
		float n = rs.getFloat("PRICE");
		System.out.println(s + " " + n);
}
```

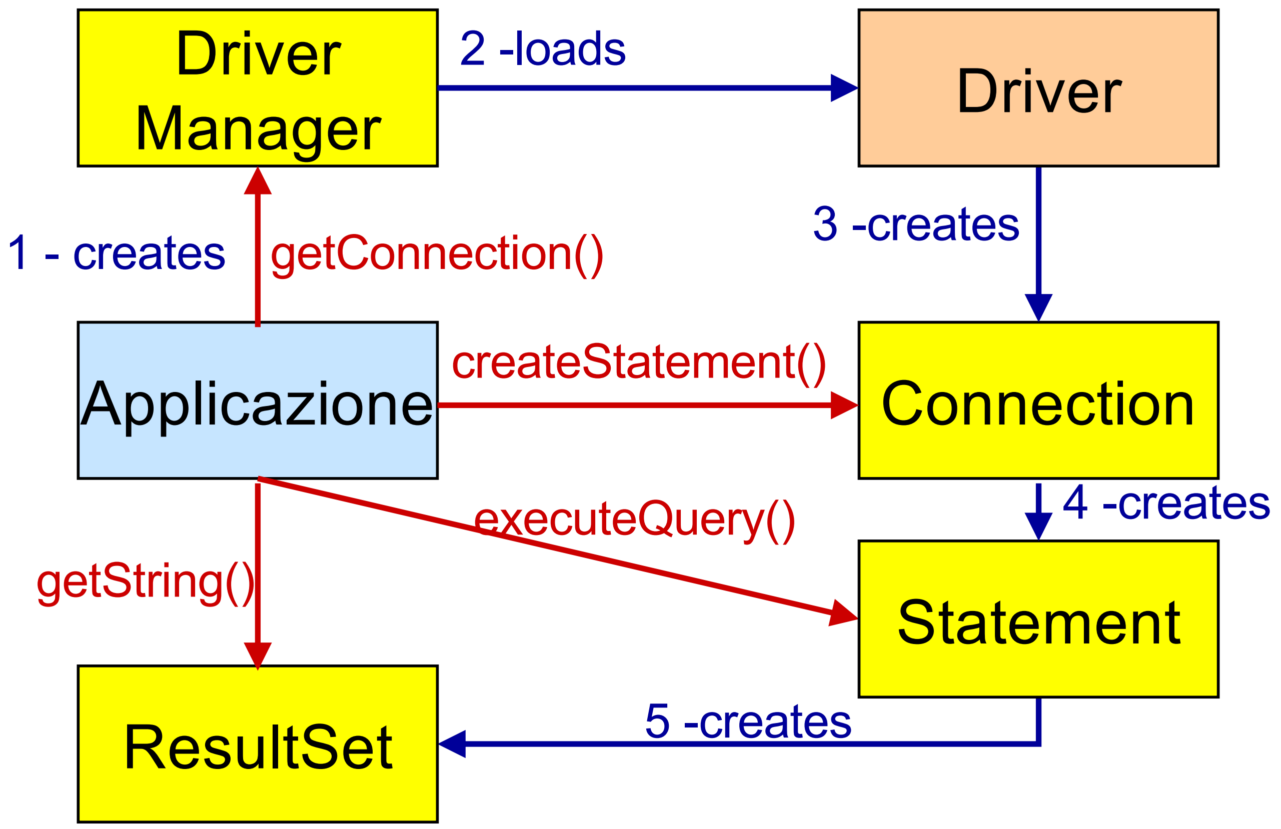
JDBC – Installation and usage (sum up)

1) Install a driver on your machine.
Your driver should include instructions for installing it. For JDBC drivers written for specific DBMSs, installation consists of just copying the driver onto your machine; there is no special configuration needed. .

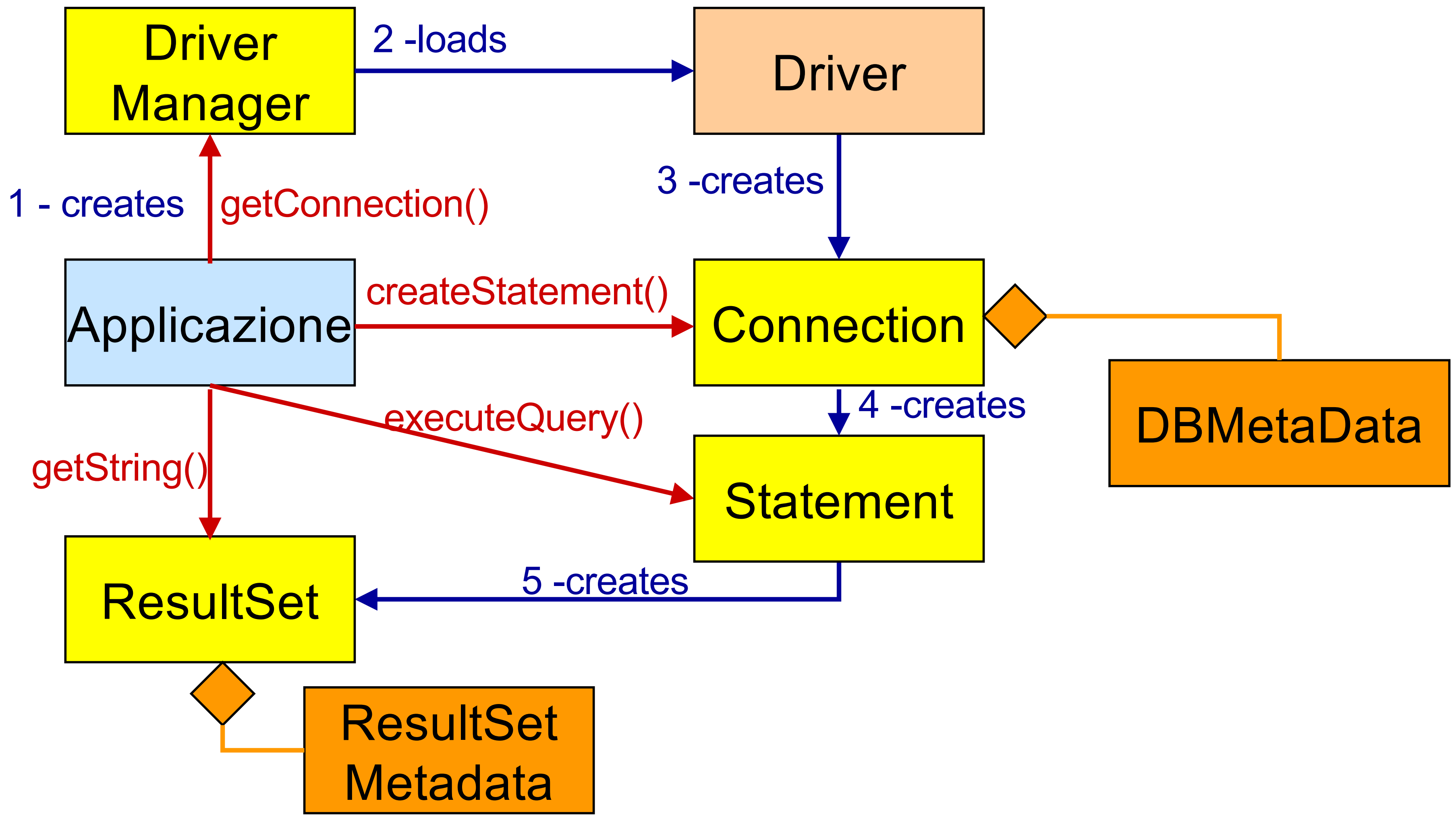
A) Load the driver.
B) Open a connection.
C) Create Statement.
D) Retrieve Values.

Always catch exceptions!
JDBC lets you see the warnings and exceptions generated by your DBMS and by the Java compiler. To see exceptions, you can have a catch block print them out. .

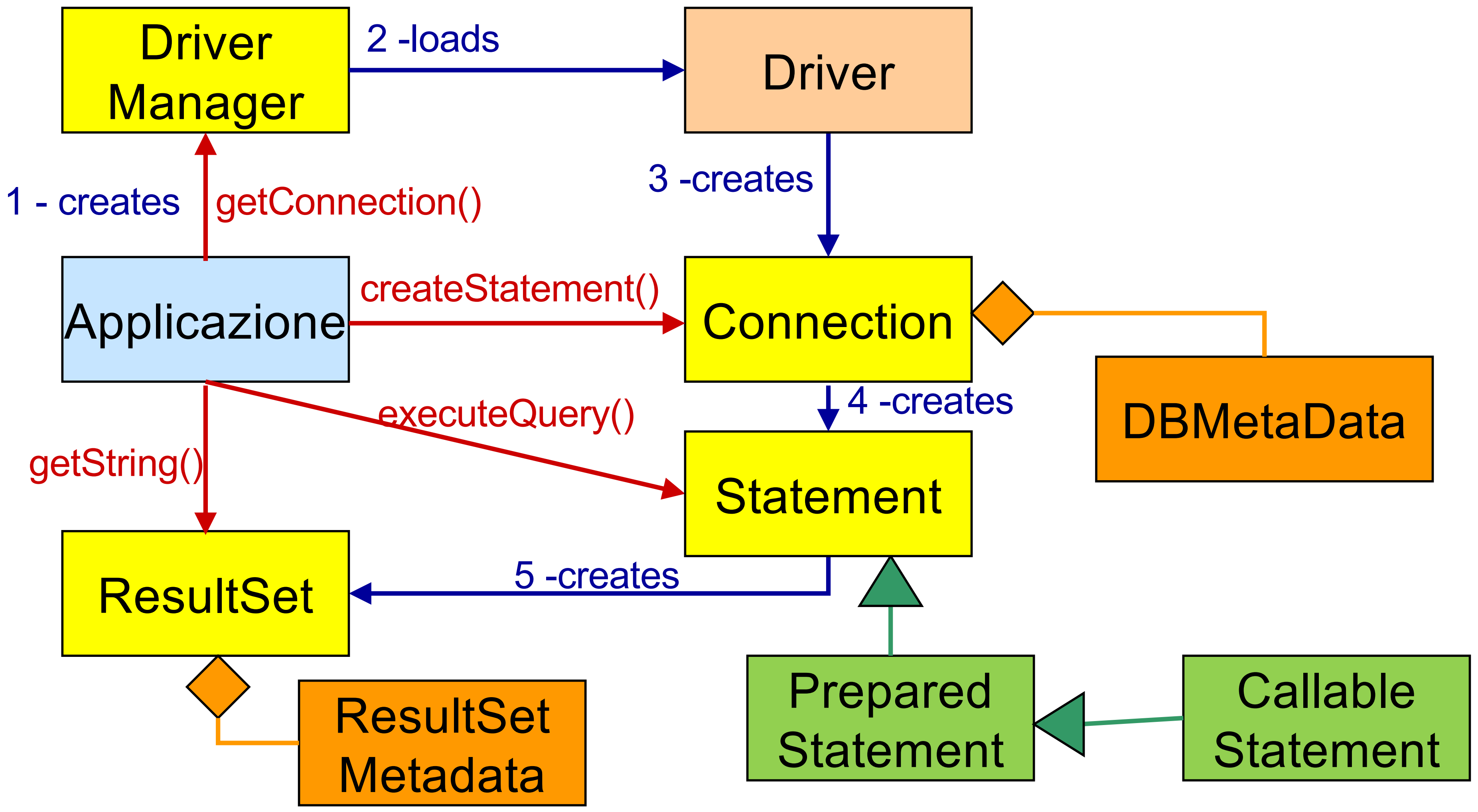
The java.sql Object Model



The java.sql Object Model: Metadata



The java.sql Object Model: predefined statement



JDBC – Prepared statements

If you want to execute a Statement object many times, it will normally reduce execution time to use a PreparedStatement object instead.

The main feature of a PreparedStatement object is that, unlike a Statement object, it is given an SQL statement when it is created.

The advantage to this is that in most cases, this SQL statement will be sent to the DBMS right away, where it will be compiled. As a result, the PreparedStatement object contains not just an SQL statement, but an SQL statement that has been precompiled.
This means that when the PreparedStatement is executed, the DBMS can just run the PreparedStatement 's SQL statement without having to compile it first.

```
PreparedStatement updateSales = con.prepareStatement( "UPDATE COFFEES
SET SALES = ? WHERE COF_NAME LIKE ?");
updateSales.setInt(1, 75);
```

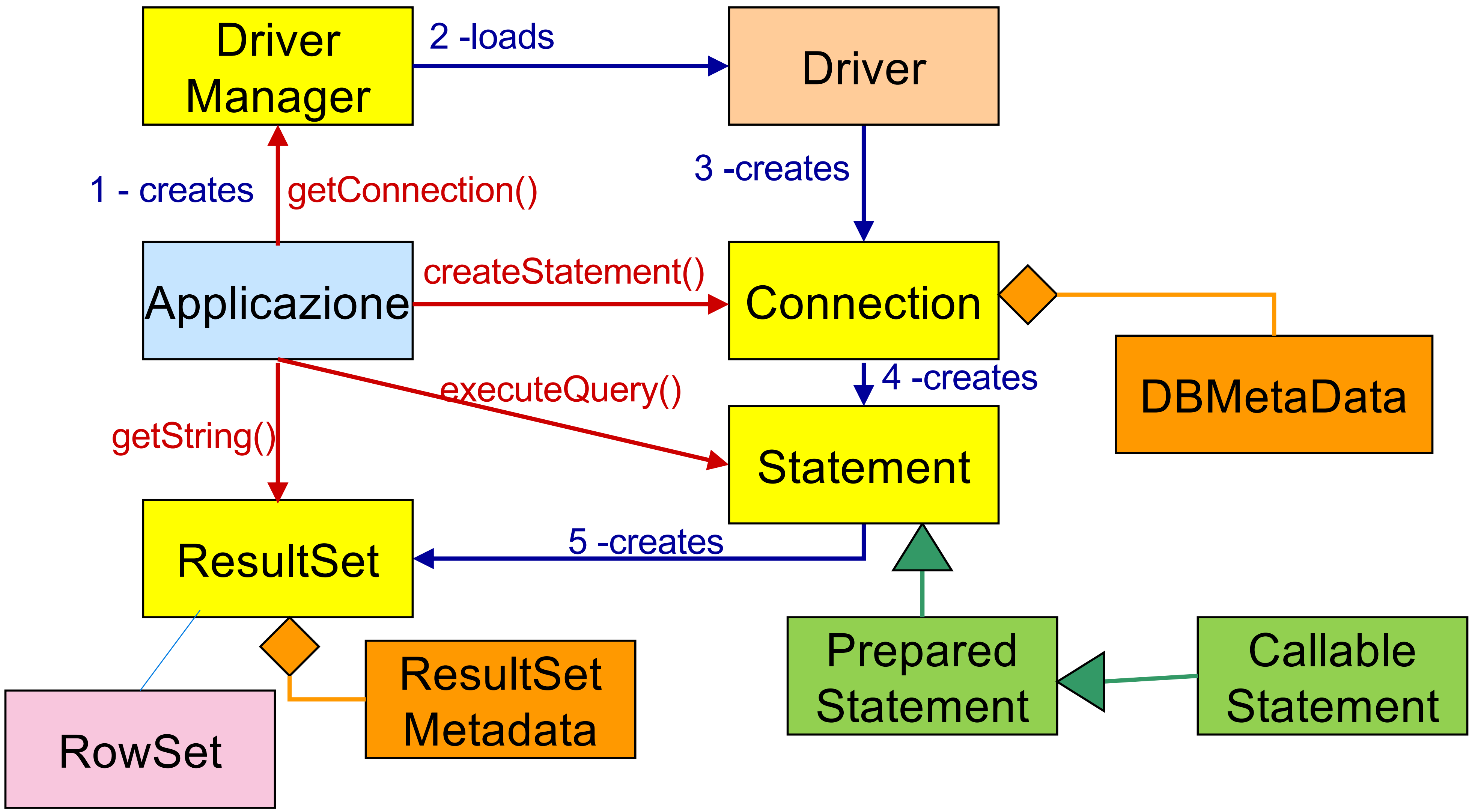
JDBC – Callable statements

A **stored procedure** is a group of SQL statements that form a logical unit and perform a particular task. Stored procedures are used to encapsulate a set of operations or queries to execute on a database server. For example, operations on an employee database (hire, fire, promote, lookup) could be coded as stored procedures executed by application code. Stored procedures can be compiled and executed with different parameters and results, and they may have any combination of input, output, and input/output parameters.
Stored procedures are supported by most DBMSs, but there is a fair amount of variation in their syntax and capabilities.

If you want to call stored procedures, you must use a CallableStatement (subclass of PreparedStatement).

WARNING: stored procedures move the business logic WITHIN THE DB!

The java.sql Object Model: RowSet



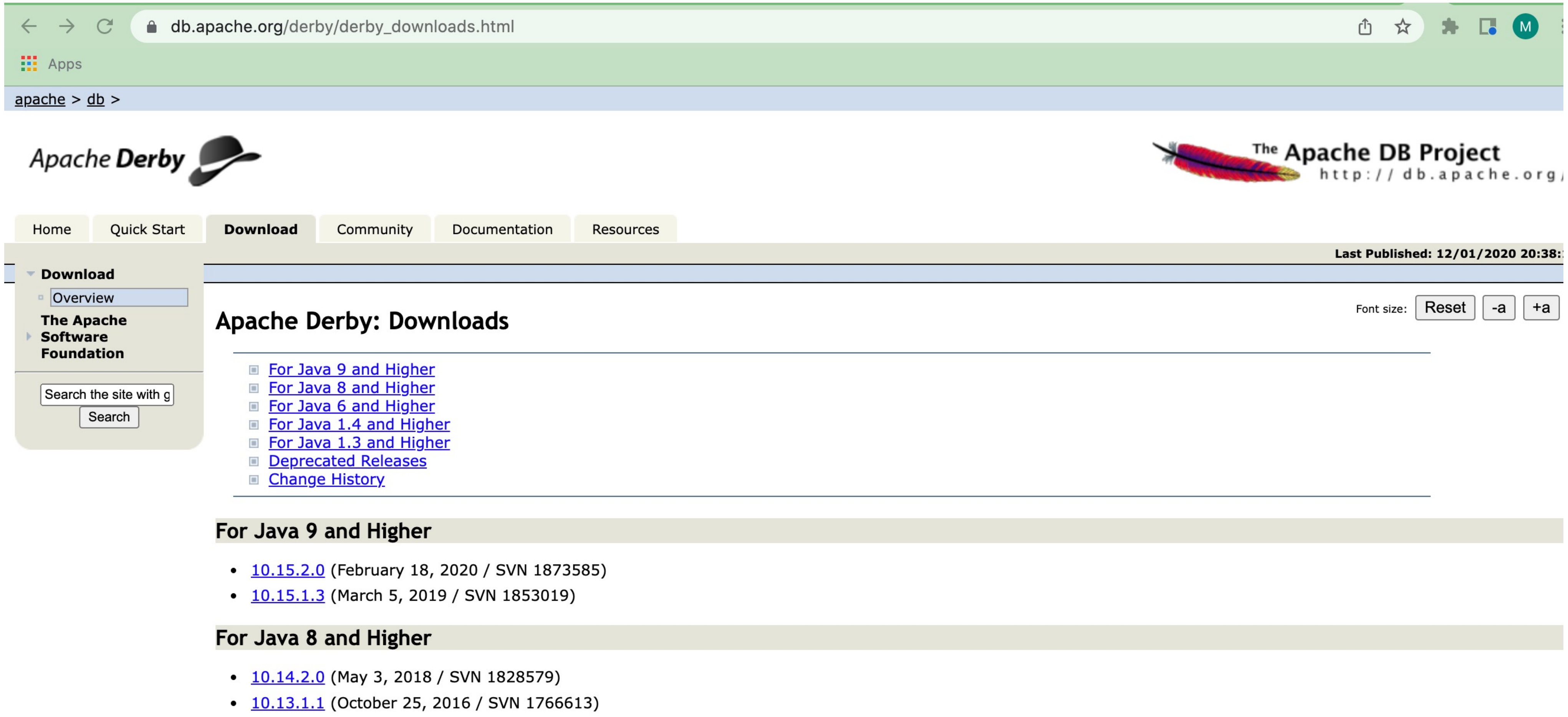
for RowSet see https://docs.oracle.com/javase/tutorial/jdbc/basics/jdbcrowset.html

Q

What do I create a IntelliJ project that uses a DB?

Example 1

Using Derby



db.apache.org/derby/derby_downloads.html

Apps

apache > db >

Apache Derby

The Apache DB Project http://db.apache.org

Last Published: 12/01/2020 20:38:

Apache Derby: Downloads

- For Java 9 and Higher
- For Java 8 and Higher
- For Java 6 and Higher
- For Java 1.4 and Higher
- For Java 1.3 and Higher
- Deprecated Releases
- Change History

For Java 9 and Higher

- 10.15.2.0 (February 18, 2020 / SVN 1873585)
- 10.15.1.3 (March 5, 2019 / SVN 1853019)

For Java 8 and Higher

- 10.14.2.0 (May 3, 2018 / SVN 1828579)
- 10.13.1.1 (October 25, 2016 / SVN 1766613)

Define environment

Make sure you have defined DERBY_HOME

e.g., put in your bash_profile:

```
export DERBY_HOME="$HOME/Download/db-derby-10.15.2.0-bin"
```

Basic (network) server operations

```
# start the network server
java -jar $DERBY_HOME/lib/derbyrun.jar server start &

#get info about the server
./NetworkServerControl sysinfo

#shudown server
java -jar $DERBY_HOME/lib/derbyrun.jar server shutdown
```

Interacting with the (EMBEDDED) server using ij

```
java -jar $DERBY_HOME/lib/derbyrun.jar ij
CONNECT 'jdbc:derby:firstdb;create=true';
CREATE TABLE FIRSTTABLE (ID INT PRIMARY KEY, NAME VARCHAR(12));
INSERT INTO FIRSTTABLE VALUES (10,'TEN'),(20,'TWENTY'),(30,'THIRTY');
SELECT * FROM FIRSTTABLE;
DROP TABLE FIRSTTABLE;
exit;
```

https://db.apache.org/derby/docs/10.15/getstart/getstartderby.pdf page 24

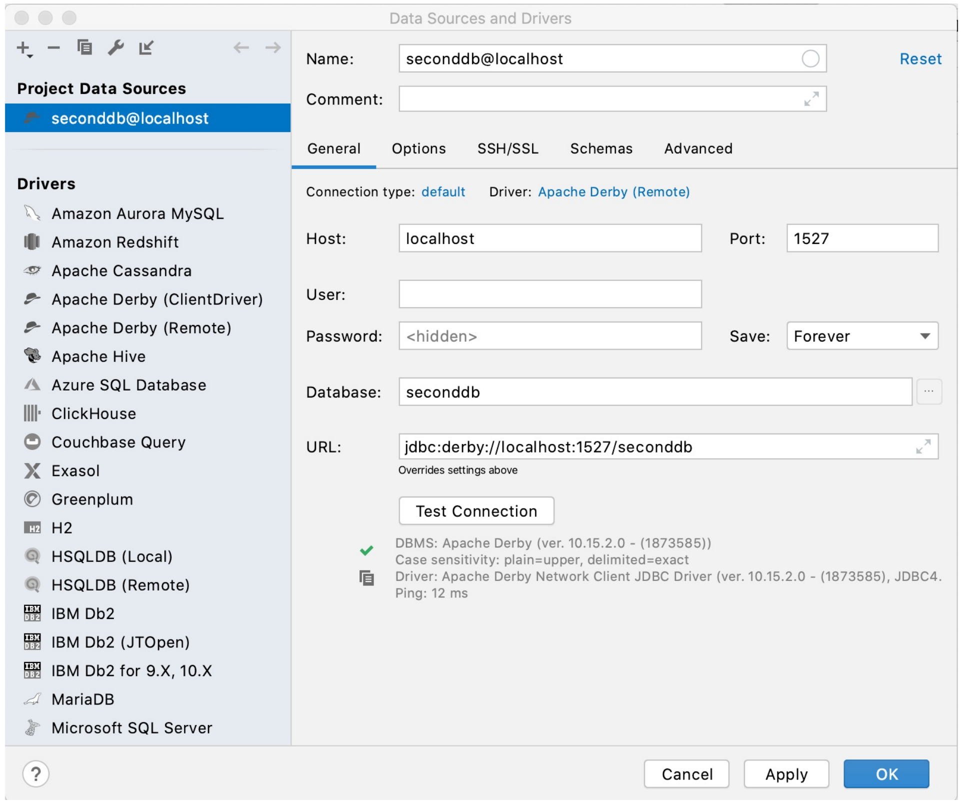
Interacting with the (NETWORK) server using ij

```
java -jar $DERBY_HOME/lib/derbyrun.jar server start &
java -jar $DERBY_HOME/lib/derbyrun.jar ij
CONNECT 'jdbc:derby://localhost:1527/seconddb;create=true';
SHOW CONNECTIONS;
CREATE TABLE EMPLOYEE( ID INTEGER not null GENERATED ALWAYS AS IDENTITY (START WITH 1,
INCREMENT BY 1) constraint  EMPLOYEE_PK primary key,  FIRSTNAME VARCHAR(30), LASTNAME
VARCHAR(30) );
INSERT INTO EMPLOYEE (FIRSTNAME, LASTNAME) VALUES ('Valentino','Rossi'),('Sofia','Goggia');
SELECT * FROM EMPLOYEE;
exit;
```

https://db.apache.org/derby/docs/10.15/getstart/getstartderby.pdf page 27

Viewing it from IntelliJ

View->Tool Windows -> Database



Data Sources and Drivers

Project Data Sources
- seconddb@localhost

Drivers
- Amazon Aurora MySQL
- Amazon Redshift
- Apache Cassandra
- Apache Derby (ClientDriver)
- Apache Derby (Remote)
- Apache Hive
- Azure SQL Database
- ClickHouse
- Couchbase Query
- Exasol
- Greenplum
- H2
- HSQLDB (Local)
- HSQLDB (Remote)
- IBM Db2
- IBM Db2 (JTOpen)
- IBM Db2 for 9.X, 10.X
- MariaDB
- Microsoft SQL Server

Name: seconddb@localhost Reset
Comment:

General | Options | SSH/SSL | Schemas | Advanced

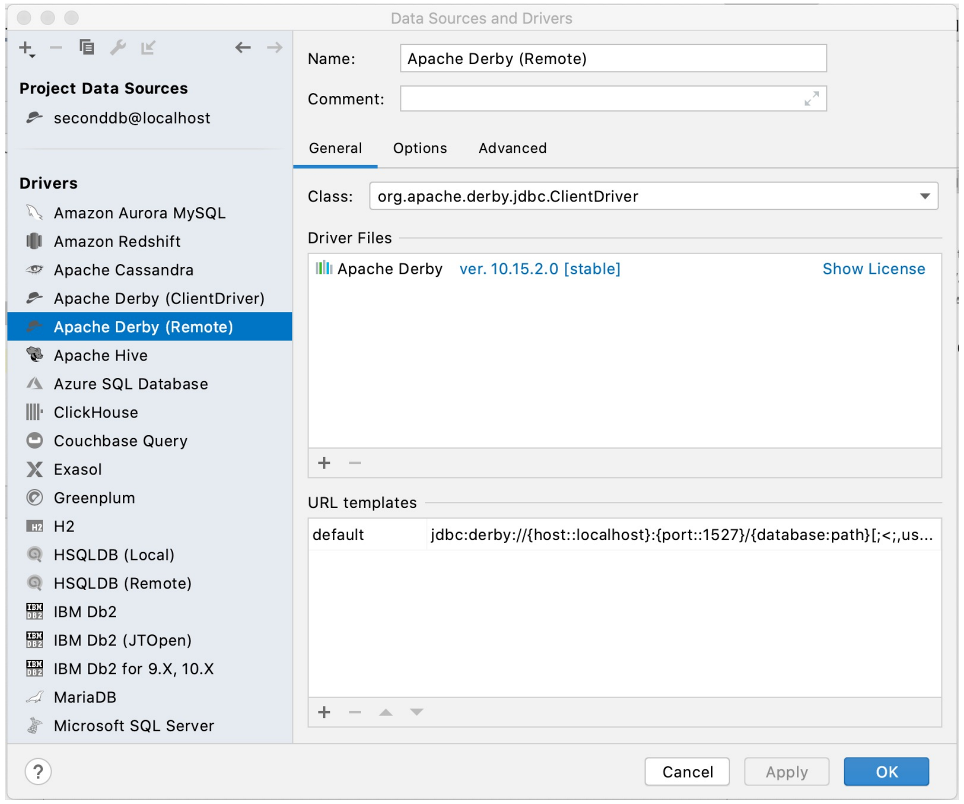
Connection type: default Driver: Apache Derby (Remote)

Host: localhost Port: 1527
User:
Password: <hidden> Save: Forever
Database: seconddb
URL: jdbc:derby://localhost:1527/seconddb
Overrides settings above

Test Connection

```
DBMS: Apache Derby (ver. 10.15.2.0 - (1873585))
Case sensitivity: plain=upper, delimited=exact
Driver: Apache Derby Network Client JDBC Driver (ver. 10.15.2.0 - (1873585), JDBC4.
Ping: 12 ms
```

Cancel | Apply | OK



Data Sources and Drivers

Project Data Sources
- seconddb@localhost

Drivers
- Amazon Aurora MySQL
- Amazon Redshift
- Apache Cassandra
- Apache Derby (ClientDriver)
- Apache Derby (Remote)
- Apache Hive
- Azure SQL Database
- ClickHouse
- Couchbase Query
- Exasol
- Greenplum
- H2
- HSQLDB (Local)
- HSQLDB (Remote)
- IBM Db2
- IBM Db2 (JTOpen)
- IBM Db2 for 9.X, 10.X
- MariaDB
- Microsoft SQL Server

Name: Apache Derby (Remote)
Comment:

General | Options | Advanced

Class: org.apache.derby.jdbc.ClientDriver

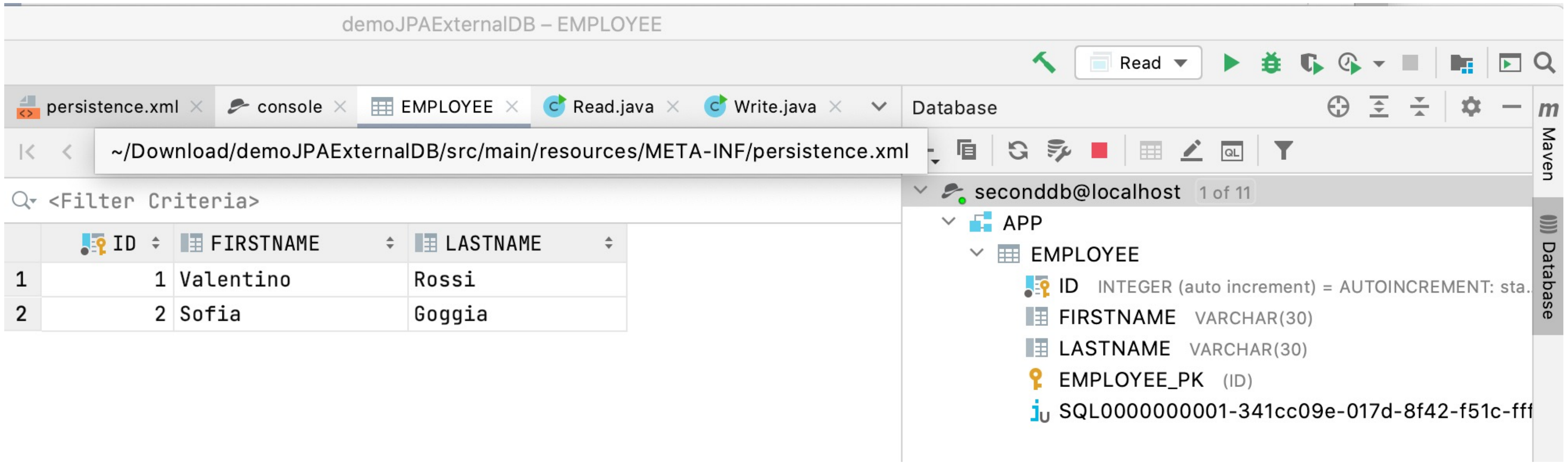
Driver Files
Apache Derby ver. 10.15.2.0 [stable] Show License

URL templates

default	jdbc:derby://{host::localhost}:{port::1527}/{database:path}[;<;,us...

Cancel | Apply | OK

Viewing it from IntelliJ



~/Download/demoJPAExternalDB/src/main/resources/META-INF/persistence.xml

<Filter Criteria>

	ID	FIRSTNAME	LASTNAME
1	1	Valentino	Rossi
2	2	Sofia	Goggia

```
seconddb@localhost  1 of 11
  APP
    EMPLOYEE
      ID  INTEGER (auto increment) = AUTOINCREMENT: sta..
      FIRSTNAME  VARCHAR(30)
      LASTNAME  VARCHAR(30)
      EMPLOYEE_PK  (ID)
      SQL0000000001-341cc09e-017d-8f42-f51c-fff
```

Viewing it from code in IntelliJ

ADD POM DEPENDENCY!

```xml
<dependency>
   <groupId>org.apache.derby</groupId>
   <artifactId>derbyclient</artifactId>
   <version>10.15.2.0</version>
</dependency>
```

VERY IMPORTANT!

MODIFY PERSISTENCE.XML

```xml
<properties>
   <property name="hibernate.connection.url" value="jdbc:derby://localhost:1527/seconddb"/>
   <property name="hibernate.connection.driver_class" value="org.apache.derby.jdbc.ClientDriver"/>
```

Note: since we now have an autoincrementing key in the DB,
we must change the generation strategy in EmployeeEntity.java:
it was: `@GeneratedValue(strategy=SEQUENCE)`
it is now: `@GeneratedValue(strategy = GenerationType.IDENTITY)`

see also https://thorben-janssen.com/jpa-generate-primary-keys/

Create servlet

Go to the project, and create a servlet called "TheServlet"

Edit it as shown in the next slides

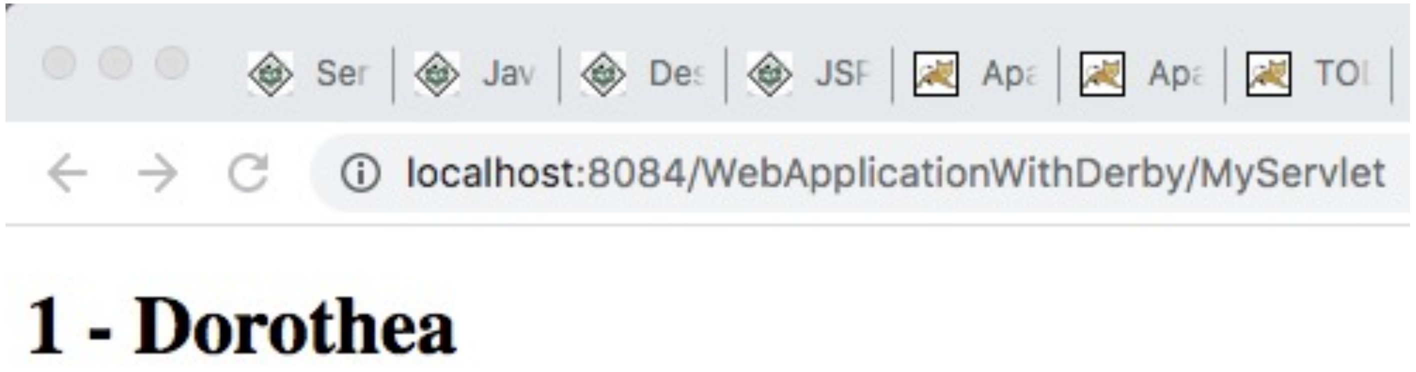
Example

```
@WebServlet(urlPatterns = {"/TheServlet"})
public class TheServlet extends HttpServlet {

    String dbURL = "jdbc:derby://localhost:1527/MyDerbyDB";
    String user = "name";
    String password = "pw";
    Connection conn = null;

    @Override
    public void init() {
        try {
            Class.forName("org.apache.derby.jdbc.ClientDriver");
            conn = DriverManager.getConnection(dbURL, user, password);
        } catch (ClassNotFoundException | SQLException ex) {
            ex.printStackTrace();
        }
    }
    @Override
    public void destroy() {
        try {
            conn.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
```

Example

```
protected void processRequest(HttpServletRequest request,
     HttpServletResponse response) throws ServletException, IOException {
   response.setContentType("text/html;charset=UTF-8");
   StringBuffer dbOutput = new StringBuffer("<h1>");
   try {
     Statement stmt = conn.createStatement();
     String sql = "SELECT ID, NAME FROM DEMO";
     ResultSet results = stmt.executeQuery(sql);
     while (results.next()) {
        dbOutput.append(results.getString(1)).append(" - ");
        dbOutput.append(results.getString(2)).append("</h1>");
     }
   } catch (SQLException ex) {
        dbOutput.append(ex.toString()).append("</h1>");
   }
   try (PrintWriter out = response.getWriter()) {
      out.println("<!DOCTYPE html><html><head>");
      out.println("<title>Servlet TheServlet</title>");
      out.println("</head><body>");
      out.println(dbOutput.toString());
      out.println("</body><html>");
        }
    }
… doGet, doPost, getServletInfo: leave them as they are
}
```

Run file…



localhost:8084/WebApplicationWithDerby/MyServlet

1 - Dorothea

Q

What is the best way to manage DB connections?

Connection management

We created the connection in the init method, and closed it in the destroy ("per Servlet connection"). Is this a good idea?

Alternatives:

create the connection in the doXXX (or processRequest) method ("per Request connection")

perServlet:

- many connections simultaneously open
- concurrency bottleneck (Connection's methods are synchronozed)

perRequest

- lots of open/close (slow!)

Connection management

We could create "per Session" connection.

perSession:

- every user has one connection, and reuses it
- potentially many connections , with low usage each
- sessions reman alive as long as the connection lives
- you should use HttpSessionBinding interface to monitor the closing of sessions due to timeout

Connection pooling

- servlets share a set of existing connection
- more complex
- infrastructures exist to allow it

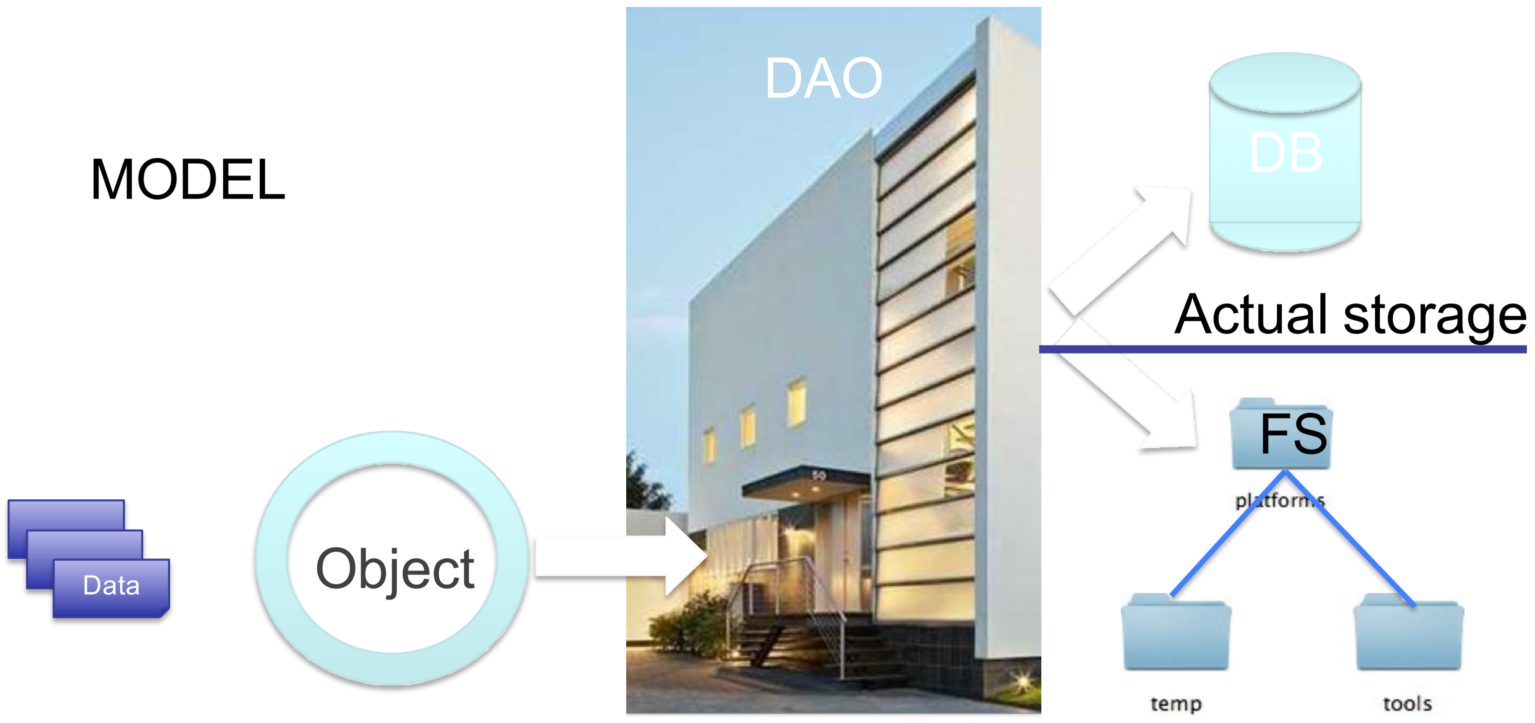
(Yet another possibility would have been "one connection per Web App". How could you have implemented it? What are its advantages and disadvantages?)

Connection management

In depth discussion, with examples

https://www.oreilly.com/library/view/java-programming-with/059600088X/ch04s02.html

Data Access Object



References about jdbc

https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

https://www.journaldev.com/2471/jdbc-example-mysql-oracle

https://www.tutorialspoint.com/servlets/servlets-database-access.htm

Extra references

if you need to refresh your SQL:

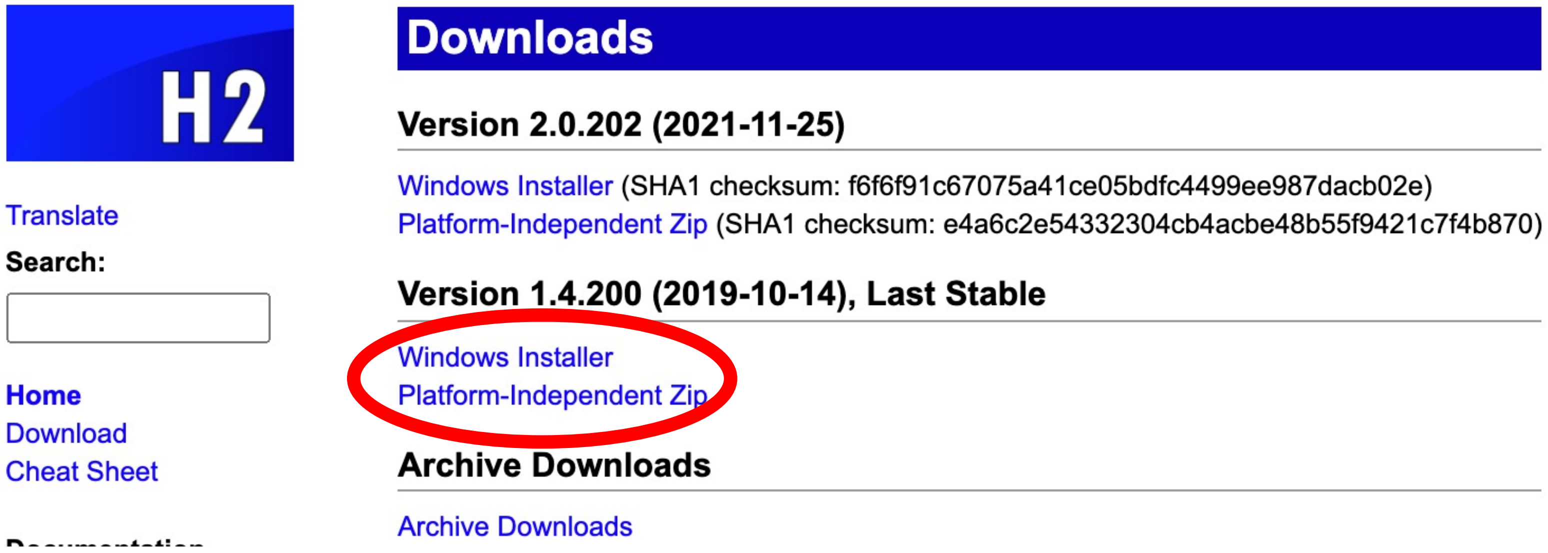
https://www.w3schools.com/sql/default.asp

If you need to install JavaDB (Derby)

https://www.codejava.net/java-se/jdbc/how-to-get-started-with-apache-derby-javadb

Changing DBMS: H2

Install H2



Downloads

Version 2.0.202 (2021-11-25)

Windows Installer (SHA1 checksum: f6f6f91c67075a41ce05bdfc4499ee987dacb02e)
Platform-Independent Zip (SHA1 checksum: e4a6c2e54332304cb4acbe48b55f9421c7f4b870)

Version 1.4.200 (2019-10-14), Last Stable

Windows Installer
Platform-Independent Zip

Archive Downloads

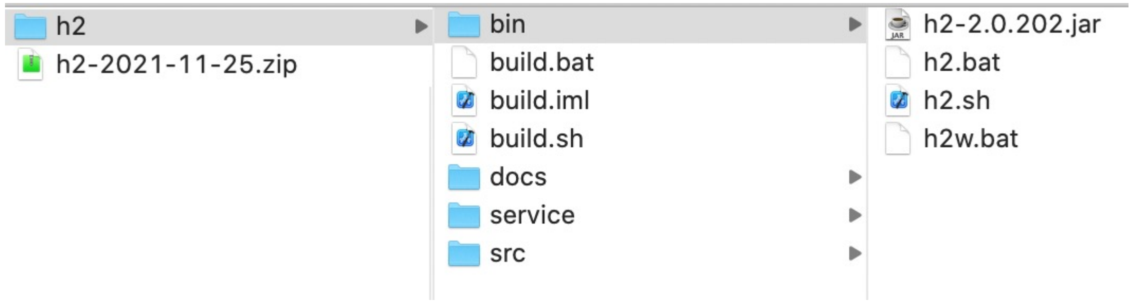
Archive Downloads

DOWNLOAD:

http://www.h2database.com/html/download.html

TUTORIAL:

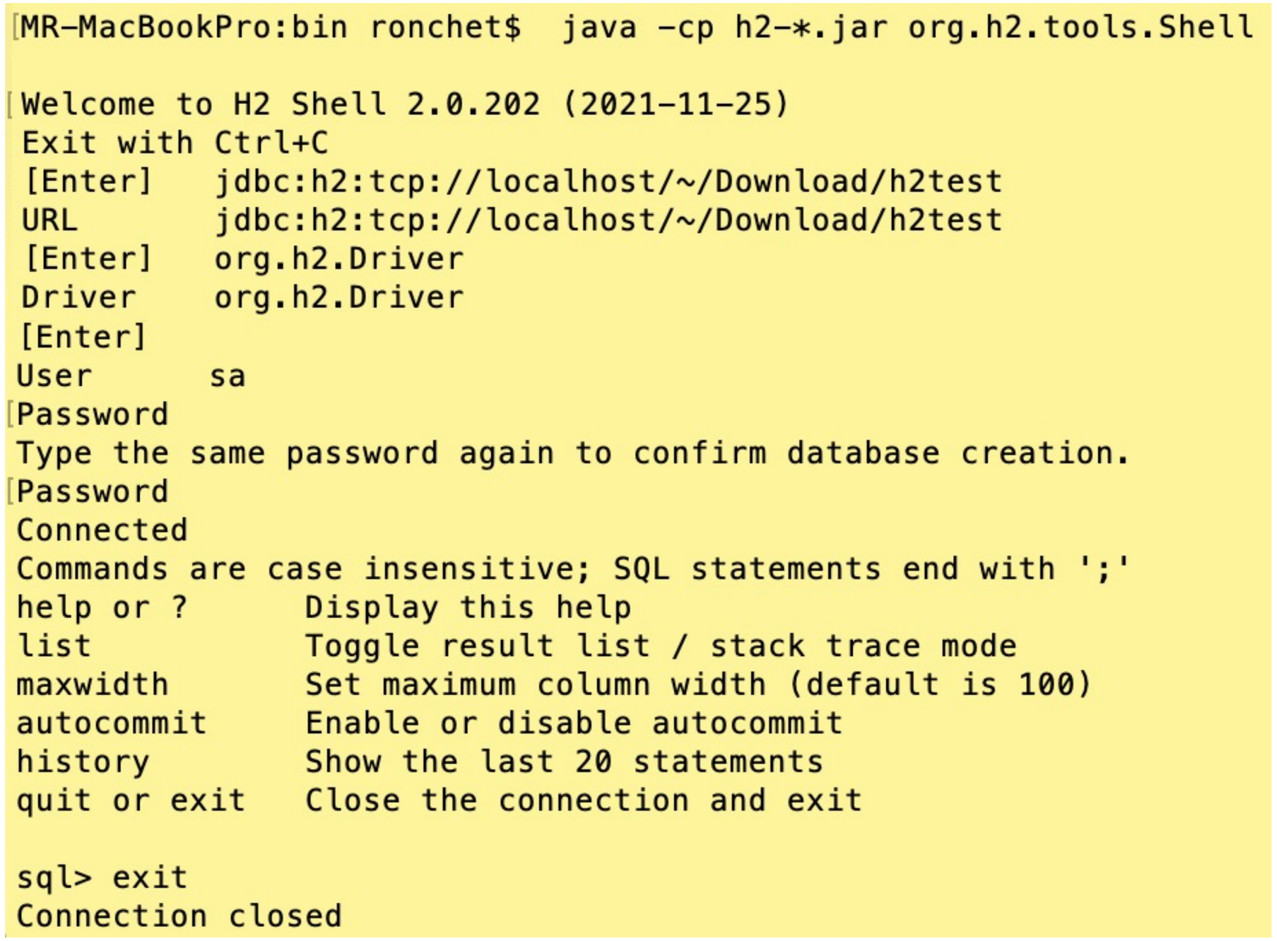
https://www.h2database.com/html/tutorial.html



Create new DB with H2

```
-cp <class search path of directories and zip/jar
files>
```

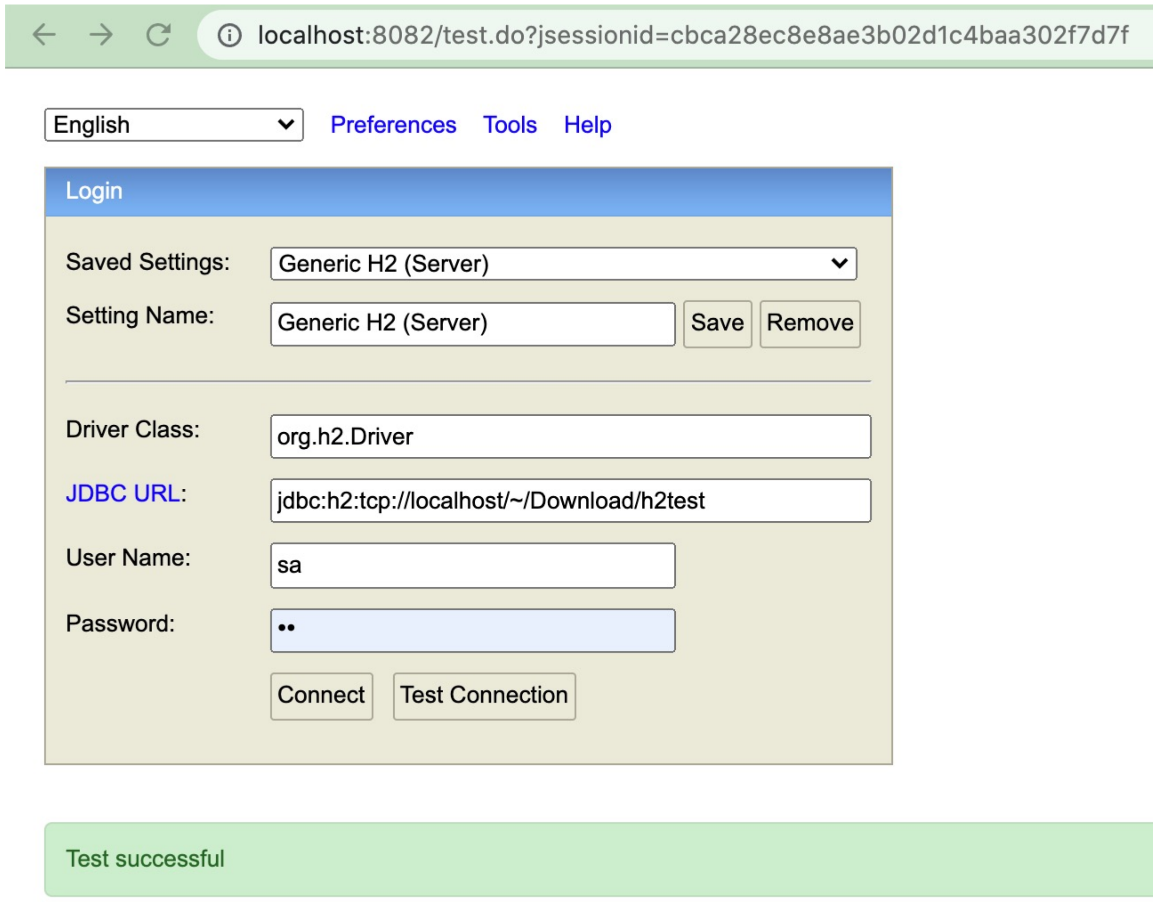
cd into h2/bin



```
MR-MacBookPro:bin ronchet$  java -cp h2-*.jar org.h2.tools.Shell

Welcome to H2 Shell 2.0.202 (2021-11-25)
Exit with Ctrl+C
[Enter]   jdbc:h2:tcp://localhost/~/Download/h2test
URL       jdbc:h2:tcp://localhost/~/Download/h2test
[Enter]   org.h2.Driver
Driver    org.h2.Driver
[Enter]
User      sa
Password
Type the same password again to confirm database creation.
Password
Connected
Commands are case insensitive; SQL statements end with ';'
help or ?      Display this help
list           Toggle result list / stack trace mode
maxwidth       Set maximum column width (default is 100)
autocommit     Enable or disable autocommit
history        Show the last 20 statements
quit or exit   Close the connection and exit

sql> exit
Connection closed
```

Access H2 with browser

java -jar h2*.jar



localhost:8082/test.do?jsessionid=cbca28ec8e8ae3b02d1c4baa302f7d7f

Saved Settings: Generic H2 (Server)

Setting Name: Generic H2 (Server)

Driver Class: org.h2.Driver

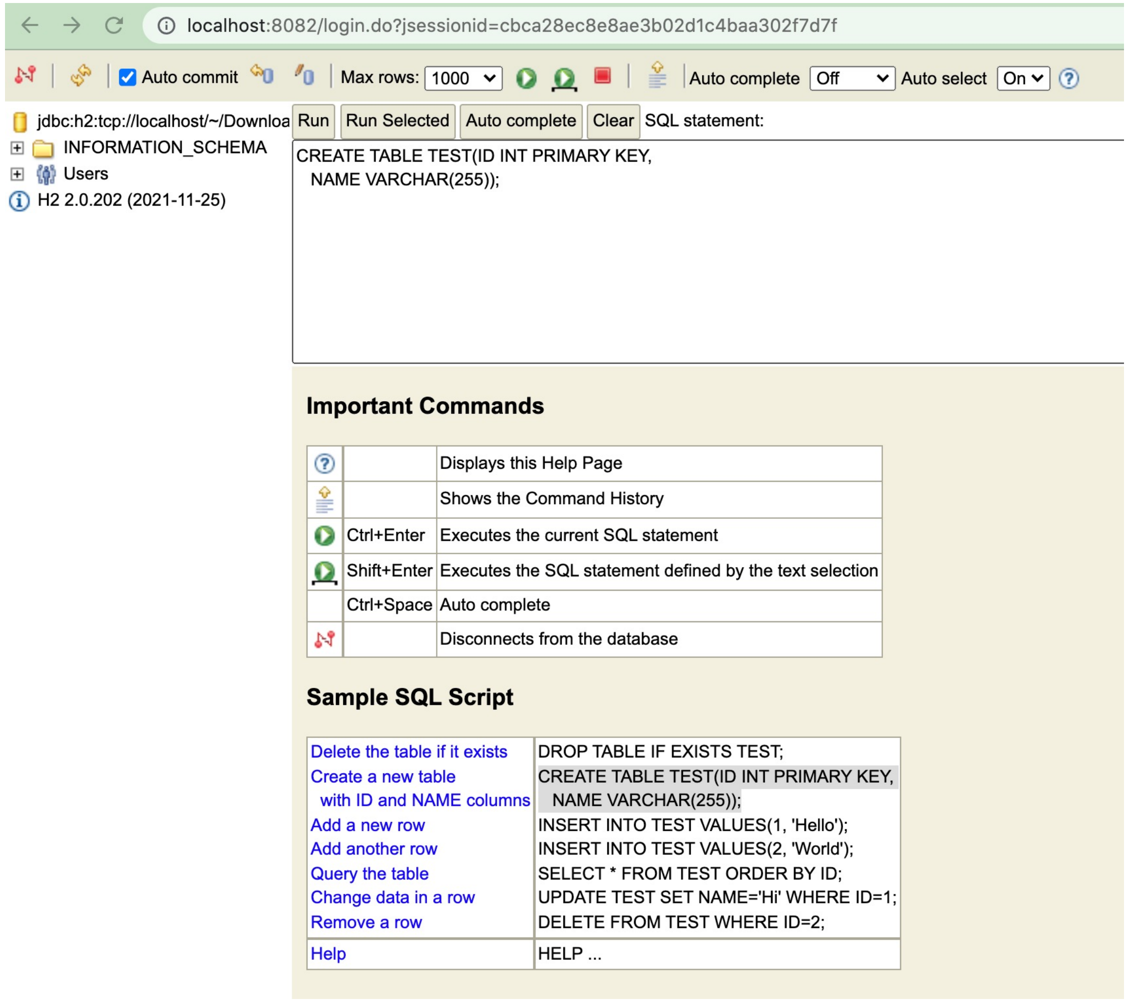
JDBC URL: jdbc:h2:tcp://localhost/~/Download/h2test

User Name: sa

Password: ••

Test successful

Access H2 with browser



localhost:8082/login.do?jsessionid=cbca28ec8e8ae3b02d1c4baa302f7d7f

SQL statement:

```
CREATE TABLE TEST(ID INT PRIMARY KEY,
  NAME VARCHAR(255));
```

Important Commands

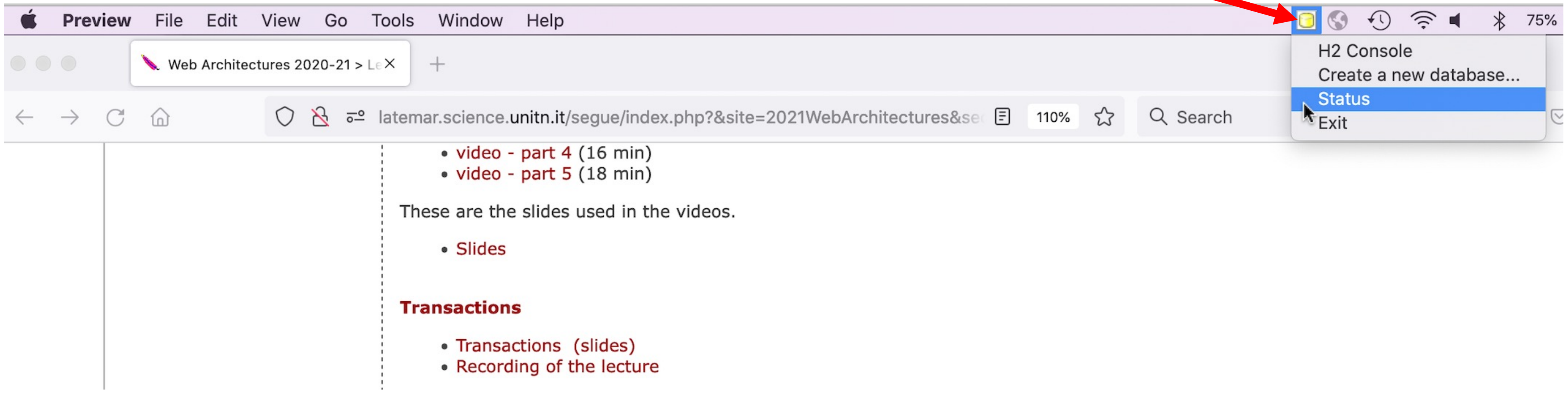
?		Displays this Help Page
		Shows the Command History
▶	Ctrl+Enter	Executes the current SQL statement
▶	Shift+Enter	Executes the SQL statement defined by the text selection
	Ctrl+Space	Auto complete
		Disconnects from the database

Sample SQL Script

Delete the table if it exists	DROP TABLE IF EXISTS TEST;
Create a new table with ID and NAME columns	CREATE TABLE TEST(ID INT PRIMARY KEY, NAME VARCHAR(255));
Add a new row	INSERT INTO TEST VALUES(1, 'Hello');
Add another row	INSERT INTO TEST VALUES(2, 'World');
Query the table	SELECT * FROM TEST ORDER BY ID;
Change data in a row	UPDATE TEST SET NAME='Hi' WHERE ID=1;
Remove a row	DELETE FROM TEST WHERE ID=2;
Help	HELP ...

Access H2 with browser

ALSO:



Preview File Edit View Go Tools Window Help
75%
Web Architectures 2020-21 > Le
latemar.science.unitn.it/segue/index.php?&site=2021WebArchitectures&se
110%
Search
H2 Console
Create a new database...
Status
Exit

- video - part 4 (16 min)
- video - part 5 (18 min)

These are the slides used in the videos.

- Slides

Transactions

- Transactions (slides)
- Recording of the lecture

Table definition

```
CREATE TABLE EMPLOYEE( ID INTEGER not null GENERATED ALWAYS
AS IDENTITY constraint  EMPLOYEE_PK primary key,  FIRSTNAME
VARCHAR(30), LASTNAME  VARCHAR(30) );

INSERT INTO EMPLOYEE (FIRSTNAME, LASTNAME) VALUES
('Valentino','Rossi'),('Sofia','Goggia');

SELECT * FROM EMPLOYEE;

exit;
```

Viewing it from code in IntelliJ

CHANGE POM DEPENDENCY!

```
<dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <version>2.0.202</version>
 </dependency>
```

VERY IMPORTANT!

MODIFY PERSISTENCE.XML

```
 <class>it.unitn.disi.ronchet.demojpa.entities.EmployeeEntity</class>
      <properties>
        <property name="hibernate.connection.url" value="jdbc:h2:tcp://localhost/~/Download/h2test"/>
         <property name="hibernate.connection.driver_class" value="org.h2.Driver" />
         <property name="hibernate.connection.username" value="sa"/>
         <property name="hibernate.connection.password" value="sa"/>
         <property name="hibernate.show_sql" value="true"/>
         <property name="hibernate.format_sql" value="true"/>
         <property name="hibernate.use_sql_comments" value="true"/>
         <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect" />
      </properties>
```