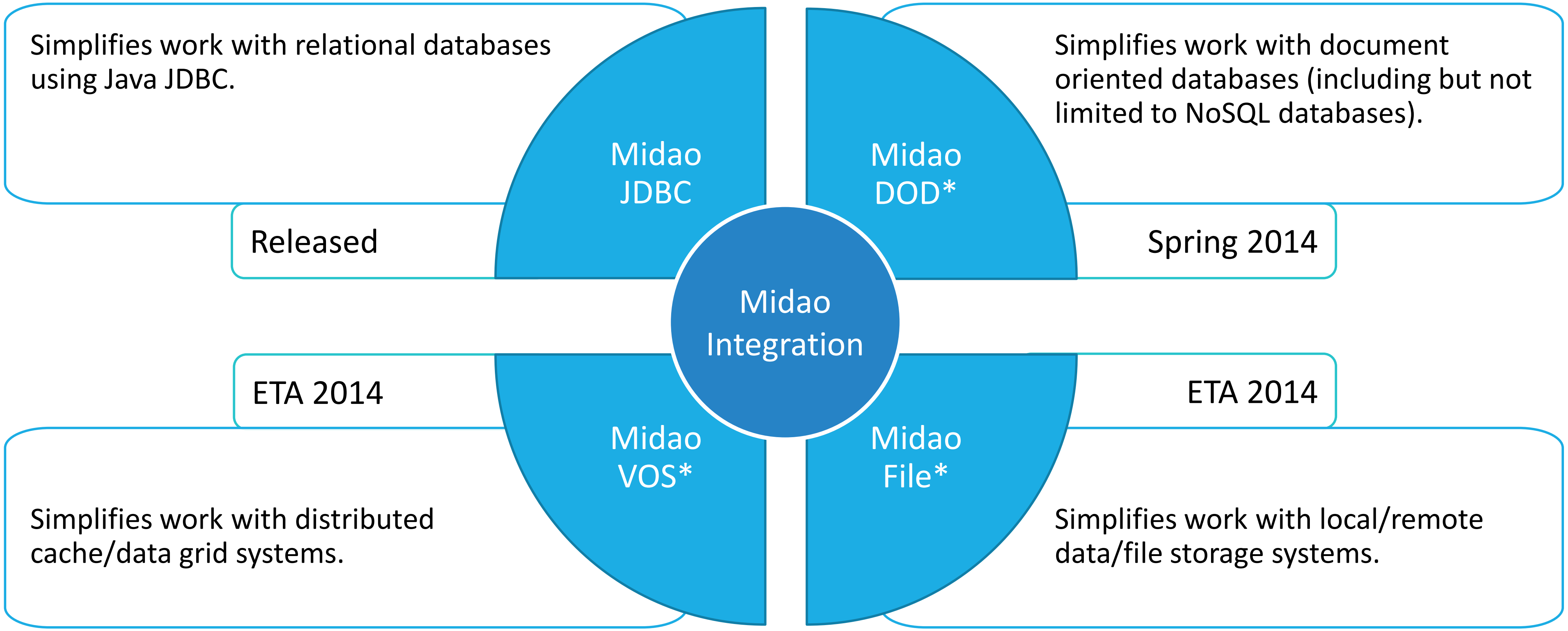
Midao

DATA ORIENTED UMBRELLA PROJECT

Midao Project

- What is it ?
  - Midao is a Project which unites few data oriented projects under it's umbrella.
- Write once – execute everywhere ?
  - Not really. Current world is all about information. Amount of tools to work with it grows very quickly, as infrastructure and complexity along with it. Amount of development and support for such systems increases because of it's diversity and complexity. Midao was created to simplify all that.
- When it would be released ?
  - Midao JDBC was already released. Midao DOD, Midao VOS, Midao File and Midao Integration would be released in Autumn 2013 – Summer 2014.
- What would be supported ?
  - Goal is to provide support for as much data sources as possible.
- Which languages are supported ?
  - Midao is developed using Java. In future, native support of other languages is possible.
- Which license is used ?
  - Apache License Version 2.0.

Midao components



Simplifies work with relational databases using Java JDBC.

Simplifies work with document oriented databases (including but not limited to NoSQL databases).

Simplifies work with distributed cache/data grid systems.

Simplifies work with local/remote data/file storage systems.

** - those are code names and are subject to change*

Midao JDBC

- What is it ?
  - Midao JDBC is part of Midao Project. It is created to provide support for RDBMS Databases using JDBC driver.
- Write once – execute everywhere ?
  - Not completely. Goal is to simplify: it hides complexity/nuances of vendor JDBC drivers, makes work with JDBC more comfortable and ensures good performance by utilizing best practices.
- When can I use it ?
  - You can start playing with it right now, but it is not ready for use in Production yet. Final release is planned in Autumn.
- What future holds ?
  - List of features to be implemented is not small. Please visit http://midao.org/status.html details.

Why Midao JDBC ?

- It simplifies work with JDBC.
  - Transactions, metadata, type handling, input/output processing, cached/lazy queries, named parameters etc.
- It simplifies work with vendor JDBC driver implementation.
  - Supports Oracle Database, Microsoft SQL, Oracle MySQL, MariaDB, PostgreSQL etc.
    - Vendor specific type handling/metadata/exception handling.
- It handles resources and provides best practices for you.
  - Library ensures that all resources are used no longer than needed and are closed properly. Also it utilizes best practices while working with JDBC Drivers.
- It provides additional functions for you.
  - Profiling, Pooled Data Sources, Lazy query execution support.
- And with all of the above - it is still small and simple to use library.

What about *others*

- Midao JDBC vs Spring JDBC*.
  - Smaller, simpler and little more customizable/extensible.
  - Provides additional functionality not present in Spring JDBC.
  - Doesn't require Spring container to work.
  - Provides some support for Spring JDBC (templates, exception handlers, metadata).
- Apache DbUtils*.
  - Handles much more JDBC boilerplate code.
  - Provides much more functionality.
  - Configurable/extensible.
  - Midao JDBC follows principles set by DbUtils. Which results in easier migration and short learning curve for Apache DbUtils projects.
- Midao JDBC vs Hibernate(and other ORM Libraries/Frameworks).
  - Smaller and have better performance(because of direct JDBC usage).
  - Midao JDBC is not an ORM Library/Framework nor tries to be one. It's main goal to work with SQL directly while providing functionality to make it more comfortable to use object relational mapping when needed.

** - above points would be elaborated in Migration Guide (ETA August-September)*

Start using

How to start ?

Grab midao-core-jdbc.jar at midao.org

Download

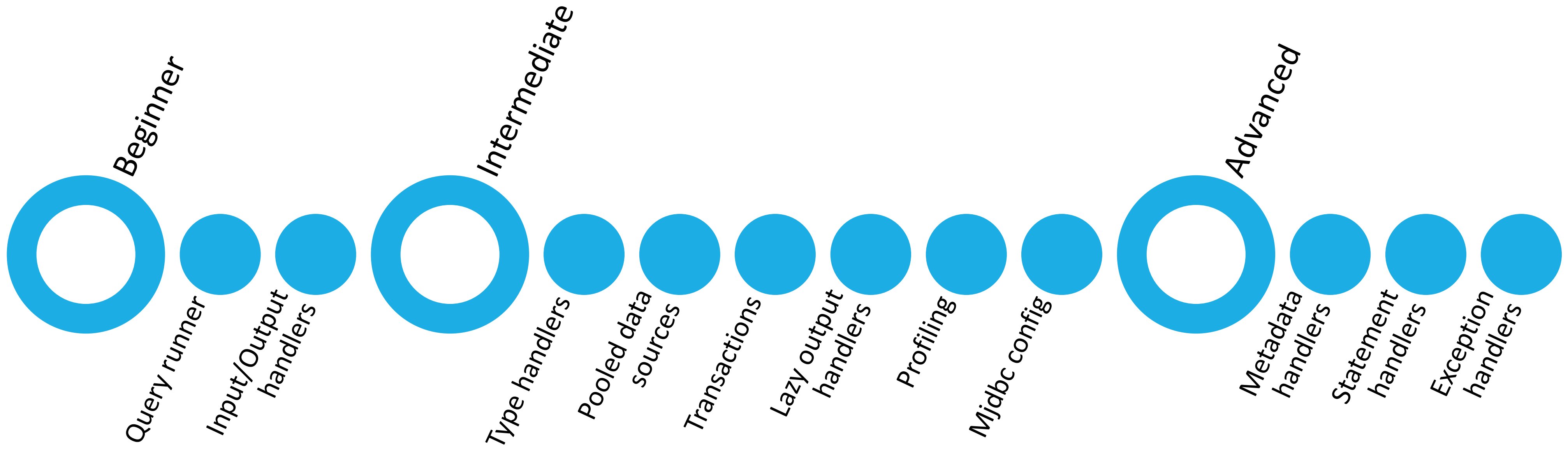
Add Maven dependency:

```
<dependency>
    <groupId>org.midao</groupId>
    <artifactId>midao-jdbc-core</artifactId>
    <version>0.9.4</version>
</dependency>
```

Maven

- Dependencies:
  - No mandatory dependencies.
  - SLF4j is optional dependency and can be used if present.
- Support:
  - Java 5&6.
  - JDBC3&JDBC4.

Often less is better



- At **beginning** focus on *Query runner* and *Input/Output* handlers.
- As you **progress** you might be interested in *Type* handlers, *Pooled data sources*, *Transactions*, *Lazy output* handlers, *Profiling* and configuration of library.
- When you will arrive to the point when you think you want to **improve** it – it is worth looking at *Metadata* handlers, *Statement* handlers and *Exception* handlers.

Assumptions

Before we start you need to set up your environment.

In below examples we assume that:

1. JDBC drivers were added to project Classpath via Eclipse or Maven/Ant.
2. You have SQL Connection received from DataSource or DriverManager.
3. Midao JDBC Core was added to project Classpath via Eclipse or Maven/Ant.

For full examples please consider viewing:
1. http://midao.org/home.html#examples
2. https://github.com/pryzach/midao/tree/master/midao-jdbc-examples/src/main/java/org/midao/jdbc/examples

Query runner, transactions and profiling

All queries are executed via Query Runner. In order to receive Query Runner instance Factory can be used:

```
QueryRunnerService runner = MjdbcFactory.getQueryRunner(conn);
```

After runner instance is received – queries can be executed via:

```
runner.batch()/runner.query()/runner.update()/runner.call();
```

Transactions are handled automatically by library (using auto-commit mode), but if you are interested in manual handling you can do:

```
runner.setTransactionManualMode(true);
...
runner.commit()/runner.rollback();
```

Profiling is switched off by default. In order to switch it on:

```
MjdbcConfig.setProfilerEnabled(true);
```

Input and Output handlers

Input handlers are responsible for handling query input: (named) SQL string and input parameters:

```
Map<String, Object> queryParameters = new HashMap<String, Object>();
queryParameters.put("id", 1);
MapInputHandler input = new MapInputHandler(
        "SELECT name FROM students WHERE id = :id", queryParameters);
```

Output handlers are responsible for handling query output (ResultSets):

```
Map<String, Object> result = runner.query(input, new MapOutputHandler());

System.out.println("Query output: " + result.get(“name”));
```

Hello World JDBC *style*

```
QueryRunnerService runner = MjdbcFactory.getQueryRunner(conn);

Map<String, Object> queryParameters = new HashMap<String, Object>();
queryParameters.put("id", 1);

MapInputHandler input = new MapInputHandler(
        "SELECT name FROM students WHERE id = :id", queryParameters);

Map<String, Object> result = runner.query(input, new MapOutputHandler());
```

Map input example

```
QueryRunnerService runner = MjdbcFactory.getQueryRunner(conn);

BeanInputHandler<Student> input = null;
Student student = new Student();
student.setId(2);

// :id is IN parameter and :name and :address are OUT parameters
input = new BeanInputHandler<Student>("{call TEST_NAMED(:id, :name, :address)}", student);

// result would be filled student object searched by ID = 2. All values would come from OUT param.
Student result = runner.call(input);
```

Bean input example

What's *next* ?

- Visit http://www.midao.org/
- Browse Midao JDBC code on GitHub.
- Browse Midao JDBC JavaDoc.
- Read getting started guide www.midao.org/mjdbc-getting-started.html
- Follow us on Freecode midao.

Feedback, suggestions and questions

- stackoverflow.com still is the best place to start asking questions.
- Java Ranch (JDBC forum) is monitored for questions related to Midao JDBC.
- Mailing lists can be used: questions@midao.org
- midao.org have feedback section(below) where you can provide one.
- And if everything else fails - you can always contact me directly:

  pryzach@gmail.com.

Wow, you are still reading ?

Hope this library would be useful for you.

THANK YOU