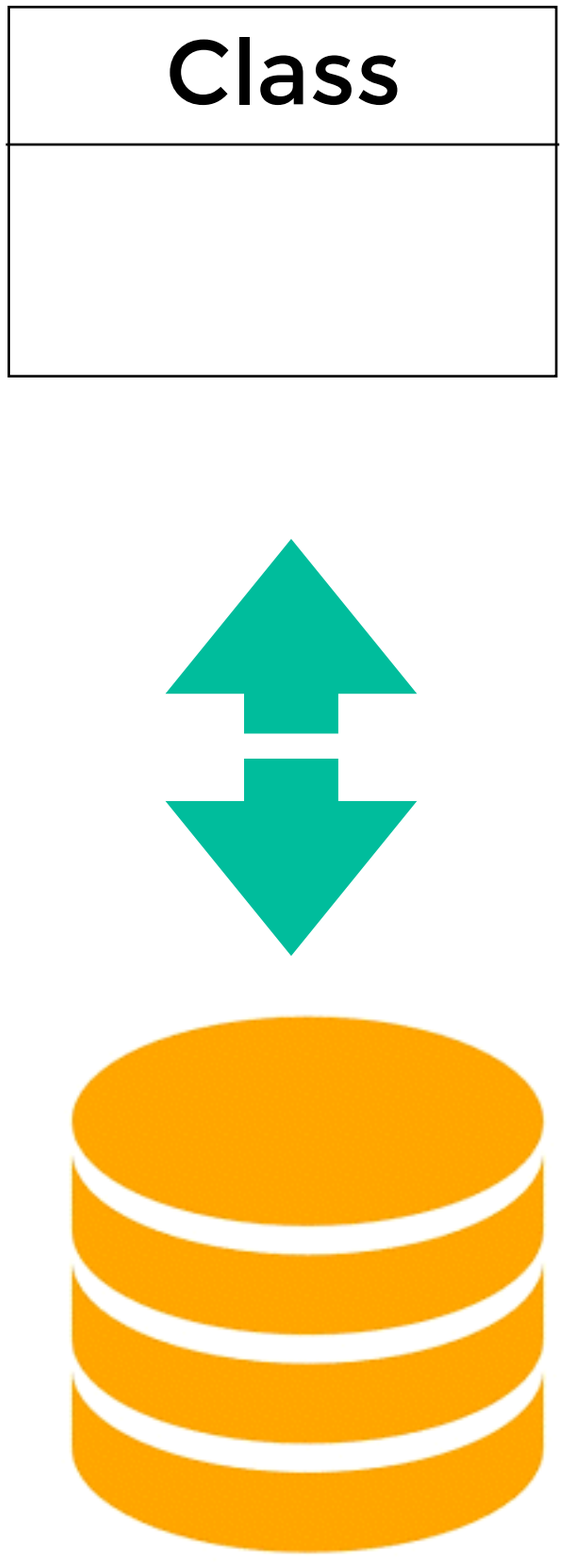
Siesta

Stored procedures based ORM for php7

Object Relational Mapper



A **Data Mapper** is a Data Access Layer that performs bidirectional transfer of data between a persistent data store (often a relational database) and an in-memory data representation (the domain layer). The goal of the pattern is to keep the in-memory representation and the persistent data store independent of each other and the data mapper itself. The layer is composed of one or more mappers (or Data Access Objects), performing the data transfer. Mapper implementations vary in scope. Generic mappers will handle many different domain entity types, dedicated mappers will handle one or a few.

The **active record pattern** is an approach to accessing data in a database. A database table or view is wrapped into a class. Thus, an object instance is tied to a single row in the table. After creation of an object, a new row is added to the table upon save. Any object loaded gets its information from the database. When an object is updated, the corresponding row in the table is also updated. The wrapper class implements accessor methods or properties for each column in the table or view.

Source: Wikipedia

Core building principles

Stored procedures

```
CREATE
    DEFINER=`root`@`localhost`
    PROCEDURE `ValueObject_DB_PK`(IN P_ID INT)
    MODIFIES SQL DATA
    SQL SECURITY INVOKER
    BEGIN
        DELETE FROM `ValueObject` WHERE `ID` = P_ID;
    END
```

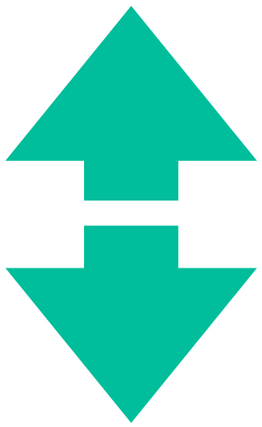
Extensibility

```
{
  "name": "Entity Generator",
  "className": "Siesta\\Generator\\EntityGenerator",
  "pluginList": [
    "Siesta\\GeneratorPlugin\\Entity\\ConstantPlugin",
    "Siesta\\GeneratorPlugin\\Entity\\MemberPlugin",
    "Siesta\\GeneratorPlugin\\Entity\\ConstructorPlugin",
    "Siesta\\GeneratorPlugin\\Entity\\SavePlugin",
    "Siesta\\GeneratorPlugin\\Entity\\FromResultSetPlugin",
```

JSON/Rest Support

Simple Mapping

PersonEntity
#id #firstName #lastName

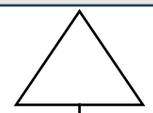


Person

id	firstName	lastName
1	Richard	Dorfmeister
2	Peter	Kruder
3	Thomas	Bangalter
4	Guillaume	Emmanuel

PersonEntity
#id #firstName #lastName

PersonService
getEntityByPrimaryKey() deleteEntityByPrimaryKey() batchSave

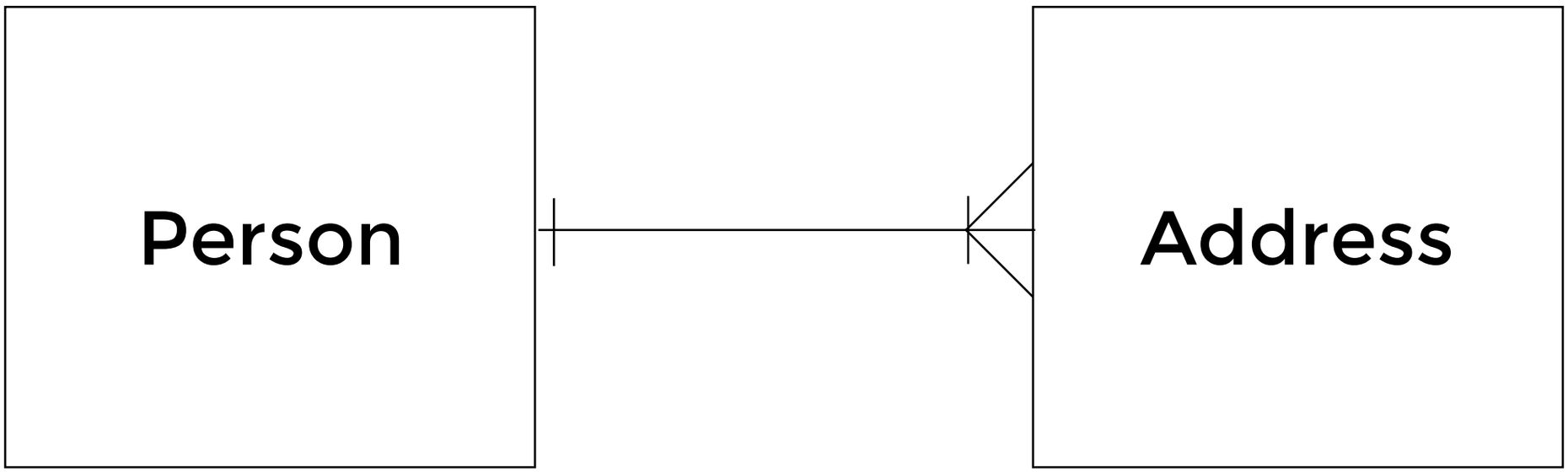


Person
+getFullName()

Generated Methods

```php
public function save(bool $cascade = false, ...)

public function fromResultSet(ResultSet $resultSet)

public function getAdditionalColumn(string $key)

public function delete(string $connectionName = null)


public function fromArray(array $data)

public function toArray(CycleDetector $cycleDetector = null)


public function fromJSON(string $jsonString)

public function toJSON(CycleDetector $cycleDetector = null) : string


public function getId(bool $generateKey = false, ...)
```

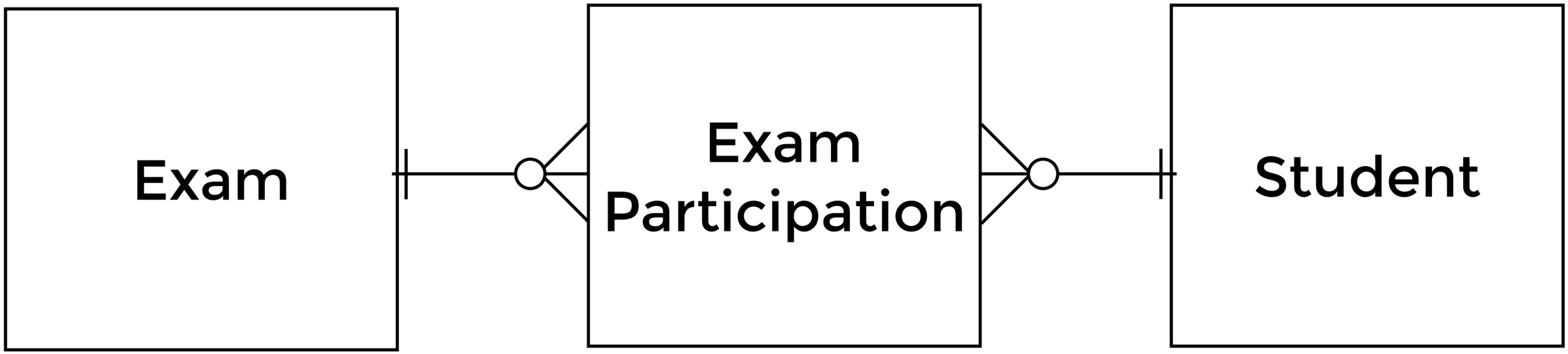
Reference & Collection




```
getAddressList() : Address[]
deleteFromAddressList(int $aid)
addToAddressList(Address $a)
```

```
setPerson(Person $p)
setPersonId(int $pid)
getPerson() : Person
getPersonId() : int
```

Collection Many2Many




```
getStudentList(): Exam[]
addToStudentList(Student $s)
deleteFromStudentList(int $sid)
deleteAssignedStudent(int $sid)
```

Siesta Generator

</>

</>

</>

</>

Siesta

Entity

Service

...

...

{}

{}

{MainConfig}

{GeneratorConfig}

Stored Procedures

```
<storedProcedure name="getPersonByFirstName" modifies="false" resultType="list">
    <parameter name="firstName" spName="P_FNAME" type="string" dbType="VARCHAR(100)"/>
    <parameter name="limit" spName="P_LIMIT" type="int" dbType="INT"/>
    <sql>
        SELECT * FROM !TABLE! WHERE `firstName` = P_FNAME LIMIT P_LIMIT;
    </sql>
</storedProcedure>
```

```
public function getPersonByFirstName(string $firstName=null, int $limit) : Person[]
```

ResultType

none	entity	list	resultSet
Update Statements	expected 0..1 Entities	expected 0..n Entities	custom queries

ResultSet

```php
$resultSet = $artistService->getMyCustomList("Parameter1", "parameter2");

// hasNext() method will tell if there is another row available.
while($resultSet->hasNext()) {

    // getter allow to access row data
    // using the generated const for every column
    $isCool = $resultSet->getBooleanValue(Artist::COLUMN_IS_COOL);
    $numberOfAlbums = $resultSet->getIntegerValue(Artist::COLUMN_NUMBER_OF_ALBUMS);

    // you can also use name of a column directly
    $custom = $resultSet->getFloatValue("custom_column_name");
    $artistName = $resultSet->getStringValue(Artist::COLUMN_NAME);
    $dob = $resultSet->getDateTime(Artist::COLUMN_DOB);

}
// mandatory(!!!)
$resultSet->close();
```

Customer Generator & Plugins

Generator

Plugin

```
{
  "name": "Entity Generator",
  "className": "Siesta\\Generator\\EntityGenerator",
  "pluginList": [
    "Siesta\\GeneratorPlugin\\Entity\\ConstantPlugin",
    "Siesta\\GeneratorPlugin\\Entity\\MemberPlugin",
    "Siesta\\GeneratorPlugin\\Entity\\ConstructorPlugin",
    "Siesta\\GeneratorPlugin\\Entity\\SavePlugin",
    "Siesta\\GeneratorPlugin\\Entity\\FromResultSetPlugin",
    "Siesta\\GeneratorPlugin\\Entity\\DeletePlugin",
    "Siesta\\GeneratorPlugin\\Entity\\ArrayConverterPlugin",
    "Siesta\\GeneratorPlugin\\Entity\\JSONConverterPlugin",
    "Siesta\\GeneratorPlugin\\Entity\\AttributeGetterSetterPlugin"
    "Siesta\\GeneratorPlugin\\Entity\\ReferenceGetterSetter",
    "Siesta\\GeneratorPlugin\\Entity\\CollectorGetterSetter",
    "Siesta\\GeneratorPlugin\\Entity\\CollectionManyAccess",
    "Siesta\\GeneratorPlugin\\Entity\\ArePKIdenticalPlugin",
    "Siesta\\GeneratorPlugin\\Entity\\ValueObjectPlugin",
    "SiestaDemo\\Plugin\\ToStringPlugin"
  ],
  "config": {
  },
  "validatorList": [
    "Siesta\\Validator\\DefaultDataModelValidator",
    "Siesta\\Validator\\DefaultEntityValidator",
    "Siesta\\Validator\\DefaultAttributeValidator",
    "Siesta\\Validator\\DefaultReferenceValidator",
    "Siesta\\Validator\\DefaultIndexValidator",
    "Siesta\\Validator\\DefaultCollectionValidator",
    "Siesta\\Validator\\DefaultCollectionManyValidator",
    "Siesta\\Validator\\DefaultStoredProcedureValidator",
    "Siesta\\Validator\\DefaultValueObjectValidator"
  ]
},
```

```php
interface Plugin
{

    /**
     * @param Entity $entity
     *
     * @return array
     */
    public function getUseClassNameList(Entity $entity) : array;

    /**
     * @return string[]
     */
    public function getDependantPluginList() : array;

    /**
     * @return array
     */
    public function getInterfaceList() : array;

    /**
     * @param Entity $entity
     * @param CodeGenerator $codeGenerator
     */
    public function generate(Entity $entity, CodeGenerator $codeGenerator);

}
```

In Memory replication

Modifying Stored Procedure

Reading Stored Procedure

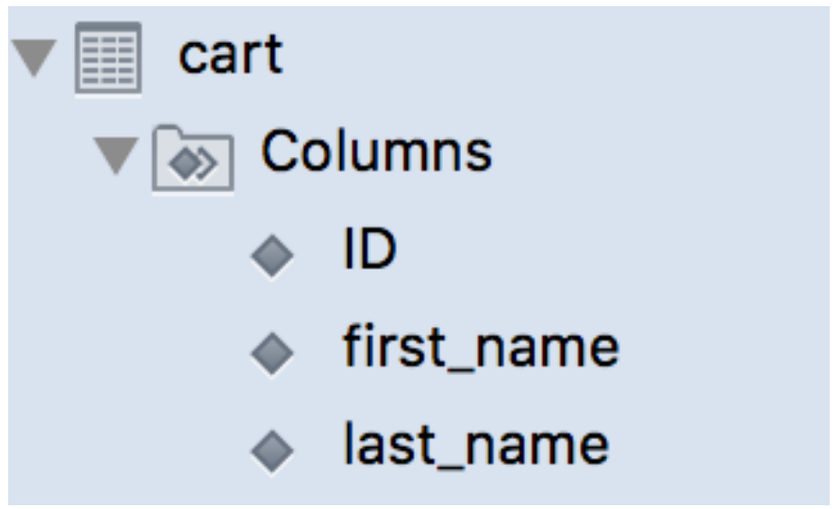
Replication table (MEMORY)

Persistent table (Innodb …)

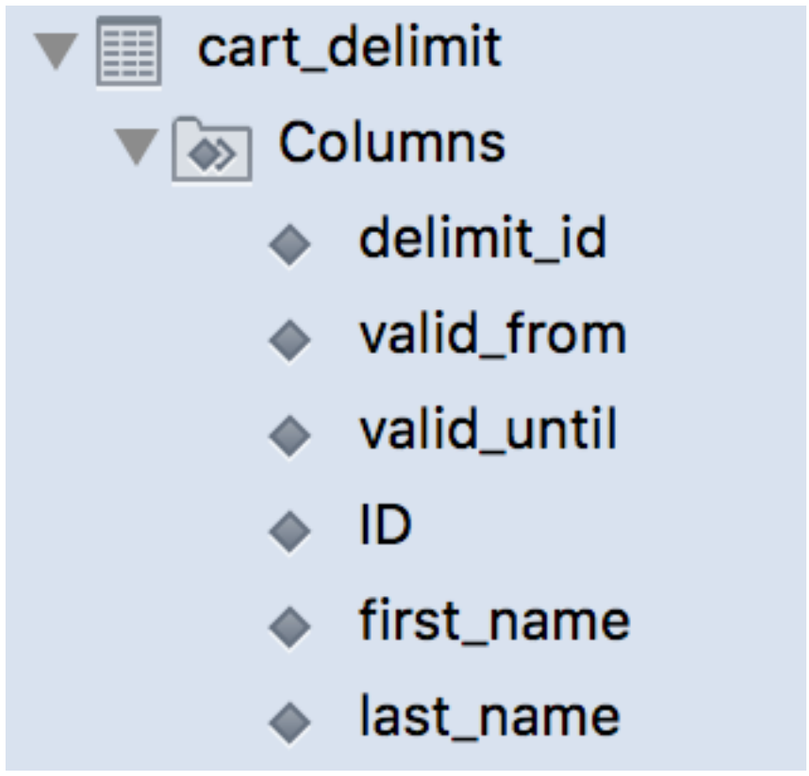
Delimiting Tables

Modifying Stored Procedure

Main Table



Delimit table



Test coverage

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		84.25%	4620 / 5484		81.70%	1112 / 1361		56.97%	94 / 165
CodeGenerator		89.87%	204 / 227		81.97%	50 / 61		50.00%	2 / 4
Config		76.49%	218 / 285		81.33%	61 / 75		40.00%	2 / 5
Console		0.00%	0 / 175		0.00%	0 / 19		0.00%	0 / 3
Contract		100.00%	0 / 0		100.00%	0 / 0			0 / 0
Database		47.96%	94 / 196		44.00%	22 / 50		58.33%	7 / 12
Driver		89.25%	855 / 958		81.01%	192 / 237		43.75%	14 / 32
Exception		0.00%	0 / 21		0.00%	0 / 3		50.00%	1 / 2
Generator		87.38%	90 / 103		94.74%	18 / 19		83.33%	5 / 6
GeneratorPlugin		96.28%	1036 / 1076		96.23%	153 / 159		84.62%	22 / 26
Logger		0.00%	0 / 100		0.00%	0 / 28		0.00%	0 / 3
Main		60.38%	64 / 106		47.22%	17 / 36		0.00%	0 / 2
Migration		92.35%	374 / 405		90.00%	63 / 70		28.57%	2 / 7
Model		94.48%	616 / 652		90.91%	240 / 264		57.89%	11 / 19
NamingStrategy		78.12%	25 / 32		69.23%	9 / 13		60.00%	3 / 5
Sequencer		81.48%	22 / 27		66.67%	6 / 9		66.67%	2 / 3
Util		60.76%	96 / 158		53.19%	25 / 47		11.11%	1 / 9
Validator		96.91%	408 / 421		93.83%	76 / 81		80.00%	8 / 10
XML		95.57%	518 / 542		94.74%	180 / 190		82.35%	14 / 17

https://github.com/gperler/siesta

https://gperler.github.io