

Zend_Db_Expr Extension - Benjamin Eberlei

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Db_Expr Function Abstraction Component Proposal

Proposed Component Name	Zend_Db_Expr Function Abstraction
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Db_Expr Function Abstraction
Proposers	Benjamin Eberlei
Zend Liaison	TBD
Revision	1.0 - 3 August 2009: Initial Draft. (wiki revision: 7)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Currently Zend_Db_Select does not allow to write completely portable applications when it comes to abstraction of SQL functions. This proposal aims to close the gap as much as possible by offering an Expression object inside Zend_Db_Select and Zend_Db_Adapter_Abstract. The largest possible subset of functions between all the current supported vendors is integrated, possible a subset of ANSI-SQL 92.

2. References

- [SQL Functions programmers reference](#)
- [ezComponents Query Expression Object API](#)

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **MUST** abstract as much SQL functions as possible that are common between: MySQL, PgSql, Sqlite, Oracle, IBM Db2, SqlSrv

- This component **MUST** be nested inside each Zend_Db_Select instance and globally inside the Adapter
- The Adapter **SHOULD** enforce the expression object to be created only once.
- This component **WILL** offer a non-compliant modus where additional vendor-specific functions are nested in
- This component **WILL** use quoting/escaping facilities internally.
- This component **WILL** enforce escaping of all variables and columns in each expression.
- Use of this component with Zend_Db_Select is **OPTIONAL**, the old way of implementing expressions stays the same.

4. Dependencies on Other Framework Components

- Zend_Db_Adapter_Abstract
- Zend_Db_Select

5. Theory of Operation

You can use the Query Expression object to generate Zend_Db_Expr instances with specific supported SQL functions.

The question is, how would this query expression object be available to the consumer? There are several possible APIs:

```
$select->e()->upper("columnName");
$select->fn()->upper("columnName");
$select->expr()->upper("columnName");
$select->func()->upper("columnName");
$select->e->upper("columnName");
$select->fn->upper("columnName");
//...
```

I think the expr() method is the best choice although its the longest, because other programming languages like C# or Java use Expression as a name for objects that perform actions on values.

Use of it would follow the lines of:

```
$select = $db->select();
$select->from("table")
    ->where( $select->expr()->eq("columnName", $value) );
    ->where( $select->expr()->like("columnName2", $value2."%") );
    ->where( $select->expr()->notEq($select->expr()->upper("col3", $value3)) );
```

In a first version I would opt to implement a common subset of functions accross all major database vendors, probably ANSI SQL 92 is a good start. Additionally if "non" compatible modus will be included that abstracts as much functions from each vendor as possible.

Because some functions have conflicting names in different vendor languages, i want to go away from sql naming in some cases and use programming language names on some issues. I guess the hardest part of this proposal is the naming issue, implementation should be easy if everything is agreed on.

6. Milestones / Tasks

- Milestone 1: Community Review
- Milestone 2: Zend Acceptance
- Milestone 3: Implementation & Documentation

7. Class Index

- Zend_Db_Expr_ExpressionAbstract
- Zend_Db_Expr_Mysql
- Zend_Db_Expr_Sqlite
- Zend_Db_Expr_Oracle
- Zend_Db_Expr_IbmDb2
- Zend_Db_Expr_PgSql
- Zend_Db_Expr_MsSqlServer

8. Use Cases

9. Class Skeletons

```
]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>
```