

Database abstraction with eZ Components

Tobias Schlitt <ts@ez.no>

International PHP Conference 2008

About me

- Tobias Schlitt <toby@php.net>
- Qualified IT Specialist
- Studying CS at the TU Dortmund
- PHP since more than 7 years
- Open Source addicted and active
 - eZ Components
 - PEAR (currently inactive)
 - PHP / PHP projects in general

Agenda

- Motivation
- eZ Components overview
- Connecting
- SQL query abstraction
- Persistent objects
- Schema abstraction
- Outlook
- Q/A

Agenda

- Motivation
- eZ Components overview
- Connecting
- SQL query abstraction
- Persistent objects
- Schema abstraction
- Outlook
- Q/A

Database abstraction, WTF?

- Why the hell do I need this?
 - “Using MySQL is wrong?”
 - “I don't need portable apps”
 - “PHP already has PDO”
 - “DB abstraction is slow!”

Is using MySQL wrong?

- No! I like it, too. :)
- Maybe your customers don't?
 - Some companies love Oracle
 - Shared hosting users love SQLite
- How to replace DB X with Y in your App?
 - Replace the DB connection and API calls
 - Replace all necessary SQL
 - Start fiddling with lovely peculiarities

Portable applications

- The application I just started is...
 - ... tiny
 - ... just for internal use
 - ... just a hack
- I've seen pigs fly
 - Small apps grow explosingly fast
 - Managers often switch technology decisions
 - Do your developers a favor!

PHP already has PDO

- PDO is a good start...
 - ... but has some flaws...
 - ... and does not go far enough...
- Driver inconsistencies
- Quoting identifiers
- SQL differences
- ORM

DB abstraction is slow

- Yes. Abstraction is always slow.
 - Use Assembler if you need raw speed! ;)
 - We try to keep it as performant as possible
- Usually you can gather a lot by
 - Content caching
 - APC
 - Scaling your servers

Concepts, patterns and approaches

- Different levels of DB abstraction
 - Access API
 - SQL
 - ORM
- Different ORM patterns (M. Fowler)
 - Table Data Gateway
 - Row Data Gateway
 - Active Record
 - Data Mapper

Agenda

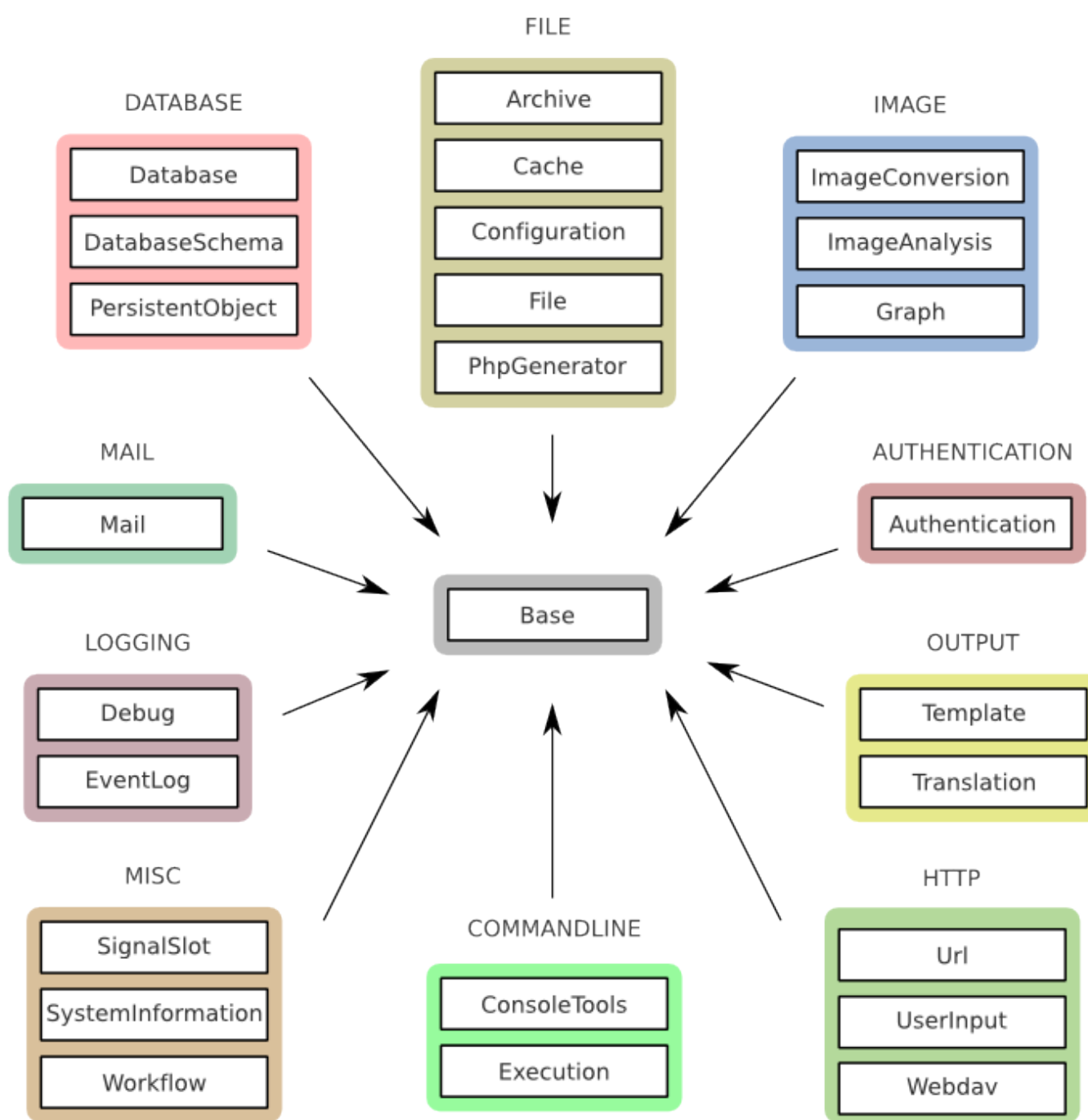
- Motivation
- eZ Components overview
- Connecting
- SQL query abstraction
- Persistent objects
- Schema abstraction
- Outlook
- Q/A

eZ Components introduction

- Open source library for PHP 5
 - New BSD license
 - General purpose
- Loosly coupled components
- Well documented
- Professional support available
 - See: <http://ez.no>



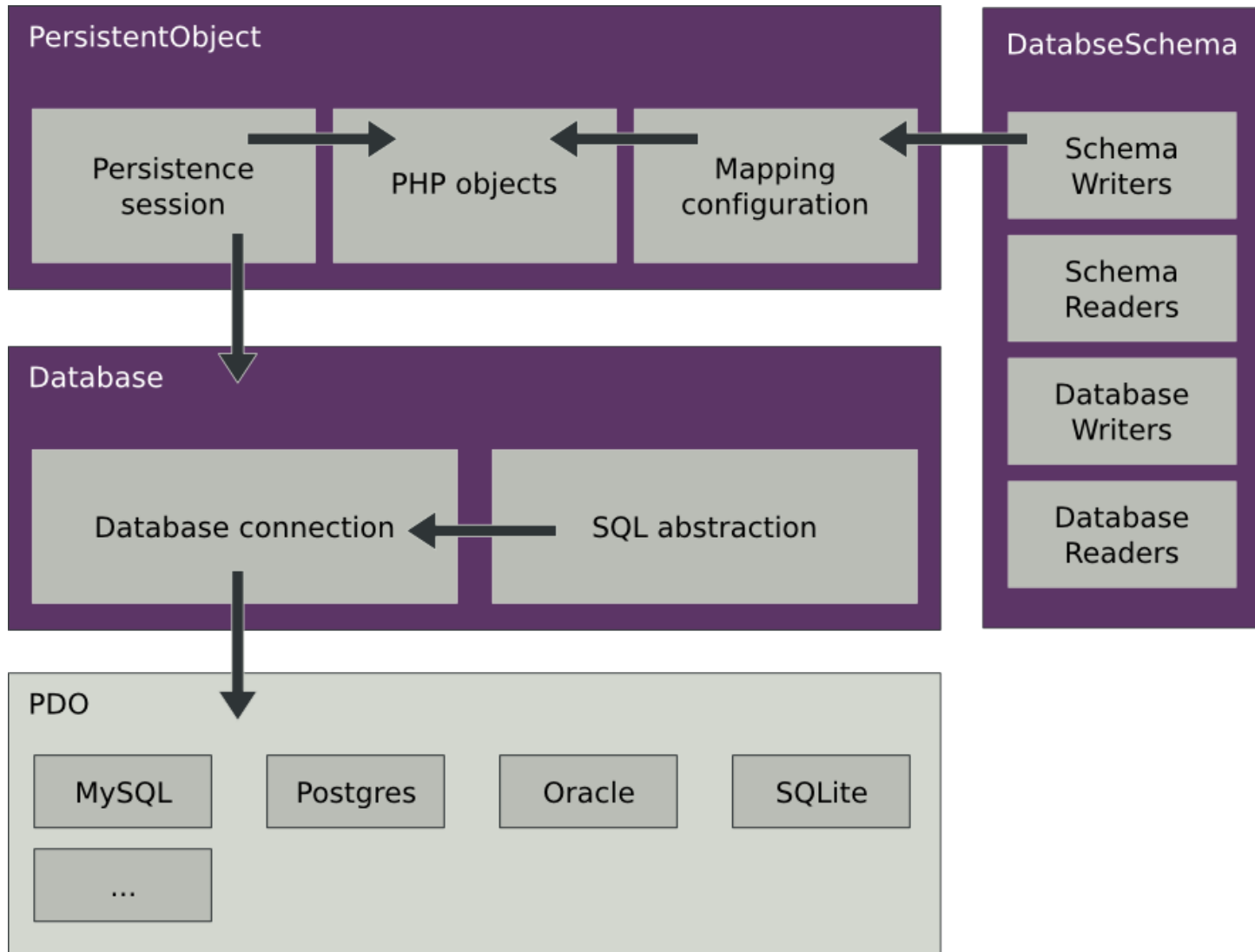
eZ Components



eZ Components goals

- Solid platform for PHP application development
 - No framework
 - Top notch API
 - High code quality
 - BC maintainance
- Clean IP
- Open source friendly

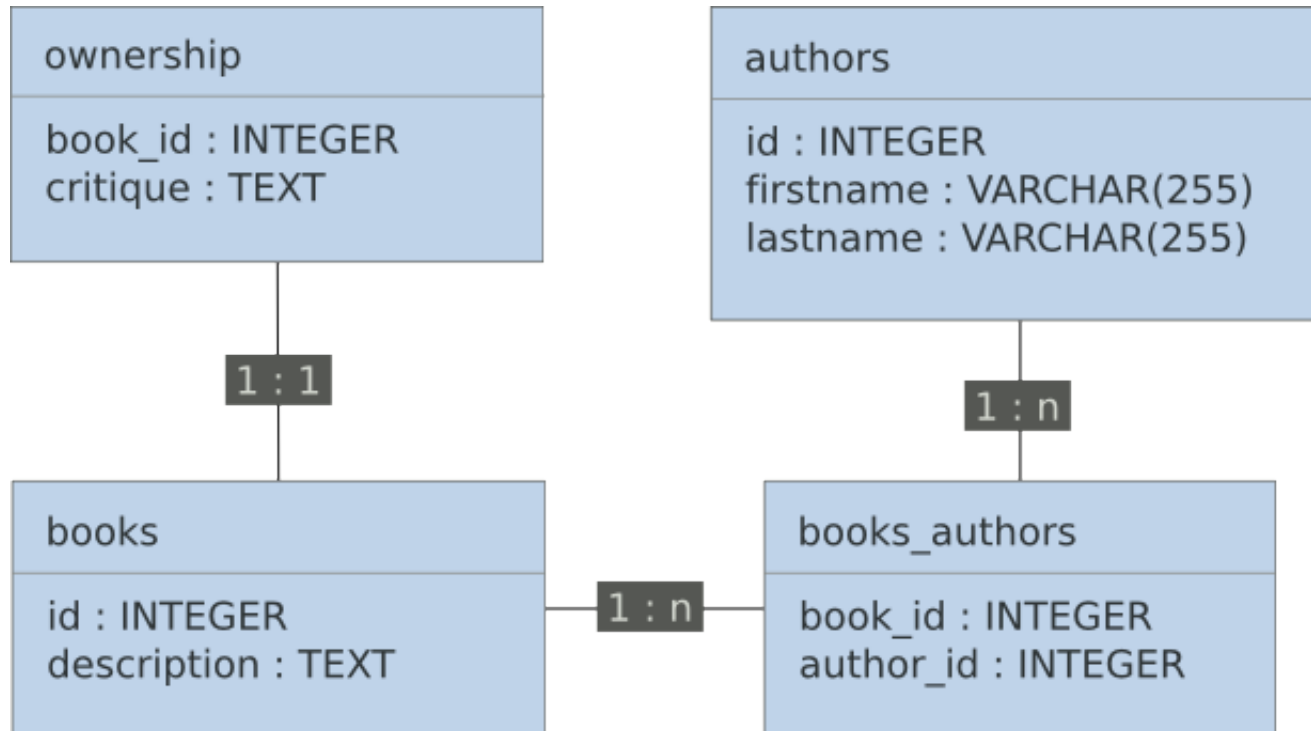
Database component stack



Agenda

- Motivation
- eZ Components overview
- **Connecting**
- SQL query abstraction
- Persistent objects
- Schema abstraction
- Outlook
- Q/A

Example database



Database connections

- Provided by: Database
- Dependencies: Base
- Idea: Enhance PDO
- Supports
 - MySQL
 - PostgreSQL
 - SQLite
 - Oracle
 - MSSQL

Creating a connection

```
// SQLite
$db = ezcdbFactory::create(
    "sqlite://$dbFile"
);

// MySQL
$db = ezcdbFactory::create(
    "mysql://user:password@localhost/database"
);

// Using the singleton/registry mechanism
ezcdbInstance::set( $db );

// ...
$db = ezcdbInstance::get( $db );
```

Usage like PDO

- ezcDbHandler extends PDO
- Adds some missing bits
 - Nested transactions
 - Identifier Quoting
 - Fix driver / DB flaws
 - SQLite function emulation
 - Oracle quoting in OCI8

Performing a SELECT

```
// Simple SQL
$sql = 'SELECT *
        FROM authors
        ORDER BY lastname, firstname';

// Complex SQL
$sql = 'SELECT books.title, authors.firstname,
            authors.lastname
        FROM books AS b
        LEFT JOIN books_authors AS ba ON ( b.id =ba.book_id )
        LEFT JOIN authors AS a ON ( ba.author_id = a.id )
WHERE b.id NOT IN( SELECT o.book_id FROM ownership AS o )
        LIMIT 1';

$stmt = $db->prepare( $sql );
$stmt->execute();
```

Agenda

- Motivation
- eZ Components overview
- Connecting
- **SQL query abstraction**
- Persistent objects
- Schema abstraction
- Outlook
- Q/A

SQL abstraction

- Provided by: Database
- Dependencies: Base
- Idea: Create DB independent SQL
 - Ensures that you stick to common sub set
 - Emulates certain DB features
 - Integrated binding

Abstracted SELECT 1/2

```
// Simple SELECT
$query = $db->createSelectQuery();
$query->select( '*' )
    ->from(
        $db->quoteIdentifier( 'authors' )
    )
    ->orderBy(
        $db->quoteIdentifier( 'lastname' )
    )
    ->orderBy(
        $db->quoteIdentifier( 'firstname' )
    );
$stmt = $query->prepare();
$stmt->execute();

// SELECT * FROM "authors" ORDER BY "lastname", "firstname"
// SELECT * FROM `authors` ORDER BY `lastname`, `firstname`
```

Abstracted SELECT 2/2

```
// Complex SELECT
$query->select(
    $db->quoteIdentifier( 'books' ) . '.'
    . $db->quoteIdentifier( 'title' ),
    // ...
)
->from(
    $db->quoteIdentifier( 'books' )
)
->leftJoin(
    $db->quoteIdentifier( 'books_authors' ),
    $query->expr->eq(
        $db->quoteIdentifier( 'books' ) . '.'
        . $db->quoteIdentifier( 'id' ),
        // ...
    )
)
// ...
->limit( 1 );
```

Abstract INSERT

```
// Insert
$insert = $db->createInsertQuery();
$insert->insertInto( $db->quoteIdentifier( 'books' ) )
    ->set(
        $db->quoteIdentifier( 'title' ),
        $insert->bindValue( 'Understanding Exposure' )
    )
    ->set(
        $db->quoteIdentifier( 'description' ),
        $insert->bindValue( 'How to Shoot Great Photo...' )
    );
```

```
// SQLite
INSERT INTO "books" ( "title", "description" )
    VALUES ( :ezcValue1, :ezcValue2 )
```

Abstract DELETE

```
// Delete
$delete = $db->createDeleteQuery();
$delete->deleteFrom( $db->quoteIdentifier( 'books' ) )
    ->where(
        $delete->expr->eq(
            $db->quoteIdentifier( 'id' ),
            $delete->bindParam( $bookId )
        )
    );
```

```
// Debugging...
echo $delete->getQuery() . "\n";
```

```
// SQLite
DELETE FROM "books" WHERE "id" = :ezcValue1
```

Agenda

- Motivation
- eZ Components overview
- Connecting
- SQL query abstraction
- **Persistent objects**
- Schema abstraction
- Outlook
- Q/A

Persistent objects

- Provided by PersistentObject
- Dependencies: Base, Database
- Idea: ORM, in some way
- Don't touch the database manually
- Provide standard DB operations
 - SELECT, INSERT, DELETE, UPDATE
 - Relation support
- Ease SQL usage

Simple model class

```
class tsBook implements ezcPersistentObject
{
    public $id;
    public $title;
    public $description;
    public function setState( array $state )
    {
        foreach ( $state as $attribute => $value )
        { $this->$attribute = $value; }
    }
    public function getState()
    {
        return array(
            'id'          => $this->id,
            'title'       => $this->title,
            'description' => $this->description,
        );
    }
}
```

Simple persistence definition

```
$def = new ezcPersistentObjectDefinition();
$def->table = 'books';
$def->class = 'tsBook';

$def->idProperty = new ezcPersistentObjectIdProperty();
$def->idProperty->columnName = 'id';
$def->idProperty->propertyName = 'id';
$def->idProperty->generator = new
    ezcPersistentGeneratorDefinition(
        'ezcPersistentSequenceGenerator'
    );
$def->properties['title'] = new ezcPersistentObjectProperty();
$def->properties['title']->columnName = 'title';
$def->properties['title']->propertyName = 'title';
$def->properties['title']->propertyType =
    ezcPersistentObjectProperty::PHP_TYPE_STRING;
// ...
return $def;
```

Why PHP configurations?

- PHP developers tend to know PHP
- Definition objects perform instant sanity checks
- The PHP parser is optimized to parse this code
- Can even be sped up using APC
- Other config formats can be compiled to PHP code (`var_export()`)
 - e.g. XML / YAML / JSON / ...

Persistence session

```
// tsBook class:    classes/book.php
// tsBook def:     persistent/tsbook.php

$session = new ezcPersistentSession(
    $db,
    new ezcPersistentCacheManager(
        new ezcPersistentCodeManager(
            dirname( __FILE__ ) . '/persistent'
        )
    )
);
```

Loading / searching objects

```
// Loading an object
$someBook = ezcPersistentSession::load( 23 );

// Searching for books
$query = $session->createFindQuery( 'tsAuthors' );
$query->orderBy( 'lastname' )->orderBy( 'firstname' );

$authors = $session->find( $query, 'tsAuthor' );
foreach ( $authors as $author )
{
    echo "{$author->lastname}, {$author->firstname}\n";
}

// SQLite
SELECT "authors"."id", "authors"."firstname",
       "authors"."lastname"
FROM "authors"
ORDER BY "authors"."lastname", "authors"."firstname"
```

Relation configuration 1/2

```
// ...
$def->properties['title']->propertyType =
    ezcPersistentObjectProperty::PHP_TYPE_STRING;

$def->relations['tsAuthor'] = new
    ezcPersistentManyToManyRelation(
        'books',
        'authors',
        'books_authors'
    );
$def->relations['tsAuthor']->columnMap = array(
    new ezcPersistentDoubleTableMap(
        'id', 'book_id',
        'author_id', 'id'
    )
);
```

Relation configuration 2/2

```
// ...
$def->relations['tsAuthor']->columnMap = ...

$def->relations['tsOwnership'] = new
ezcPersistentOneToOneRelation(
    'books',
    'ownership'
);
$def->relations['tsOwnership']->columnMap = array(
    new ezcPersistentSingleTableMap( 'id', 'book_id' )
);
$def->relations['tsOwnership']->cascade = true;

return $def;
```

Adding new objects

```
$db->beginTransaction();  
  
$book = new tsBook();  
$book->title = 'Understanding Exposure';  
$book->description = 'How to Shoot Great Photographs...';  
  
$session->save( $book );  
// echo $book->id;  
  
$author = new tsAuthor();  
$author->firstname = 'Bryan';  
$author->lastname = 'Peterson';  
  
$session->save( $author );  
  
$session->addRelatedObject( $book, $author );  
  
$db->commit();
```

Removing objects

```
$db->beginTransaction();  
  
// $session->removeRelatedObject( $book, $author );  
  
$session->delete( $book );  
  
$session->delete( $author );  
  
$db->commit();
```

Agenda

- Motivation
- eZ Components overview
- Connecting
- SQL query abstraction
- Persistent objects
- **Schema abstraction**
- Outlook
- Q/A

Schema abstraction

- Provided by: DatabaseSchema
- Dependencies: Base, Database
- Idea: Store abstract schemas
 - Read from DB X
 - Store on disc (PHP / XML)
 - Write to DB Y
 - Compare DB X and Y schemas
 - Store diff
 - Patch DB Y / Z

Reading schemas

```
// Instance of ezcdbSchema
$schema = ezcdbSchema::createFromDb( $db );

$schema->writeToFile(
    'array',
    dirname( __FILE__ ) . '/schema.php'
);

$schema->writeToFile(
    'xml',
    dirname( __FILE__ ) . '/schema.xml'
);
```

Diffing schemas

```
$modifiedSchema = ezcDbSchema::createFromFile(
    'xml',
    dirname( __FILE__ ) . '/schema_modified.xml'
);

$diff = ezcDbSchemaComparator::compareSchemas(
    $schema,
    $modifiedSchema
);

$diff->writeToFile(
    'xml',
    dirname( __FILE__ ) . '/schema_diff.xml'
);

$diff->applyToDb( $db );
```

PersistentObject stubs

- Provided by:
PersistentObjectDatabaseSchemaTiein
- Dependencies: Base, DatabaseSchema, ConsoleTools
- Idea: Database -> PersistentObject
 - Generates definition stubs
 - Generates class stubs
 - Supports class prefixes

Agenda

- Motivation
- eZ Components overview
- Connecting
- SQL query abstraction
- Persistent objects
- Schema abstraction
- Outlook
- Q/A

Outlook

- 2008.2 (about christmas, alpha in 2 weeks)
 - Webdav (authentication, authorization, locking)
 - Document (wiki dialects, rst, docbook, xhtml)
 - MvcTools (brand new!)
- In the future
 - Identity map for PersistentObject
 - Better join support
- What would you like to see?

Agenda

- Motivation
- eZ Components overview
- Connecting
- SQL query abstraction
- Persistent objects
- Schema abstraction
- Outlook
- Q/A

Q/A

- Any questions left?
- Thank you for listening!
- Slides will be online soonish
 - Watch <http://blog.schlitt.info>
- Further questions
 - Tobias Schlitt <ts@ez.no>

The end...

Shameless plug:



- Introduction to the main eZ Components
- Extend eZ Components with own code
- Integrate and combine components
- Patterns and application architecture
- Practical application example

ISBN: 978-3-8362-1073-7

<http://www.galileocomputing.de/katalog/buecher/titel/gp/titelID-1545>

License

- This set of slides and the source code included in the download package is licensed under the

Creative Commons Attribution-Noncommercial-Share Alike 2.0 Generic License

- <http://creativecommons.org/licenses/by-nc-sa/2.0/deed.en>