Performing Magic with PHP

Metaprogramming and Reflection

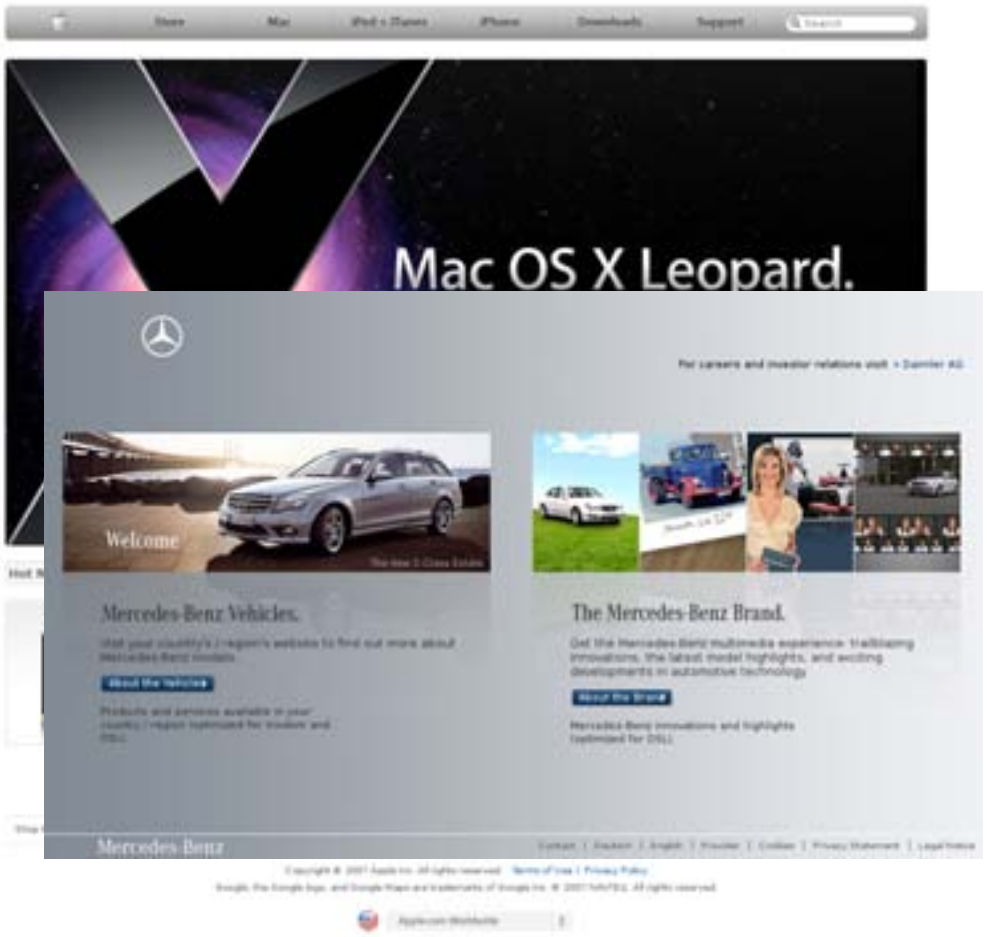
Gregor Gabrysiak Stefan Marr Falko Menge

Hasso Plattner Institute
University of Potsdam

15 January 2008

We all know what PHP is...

We all know what PHP is...

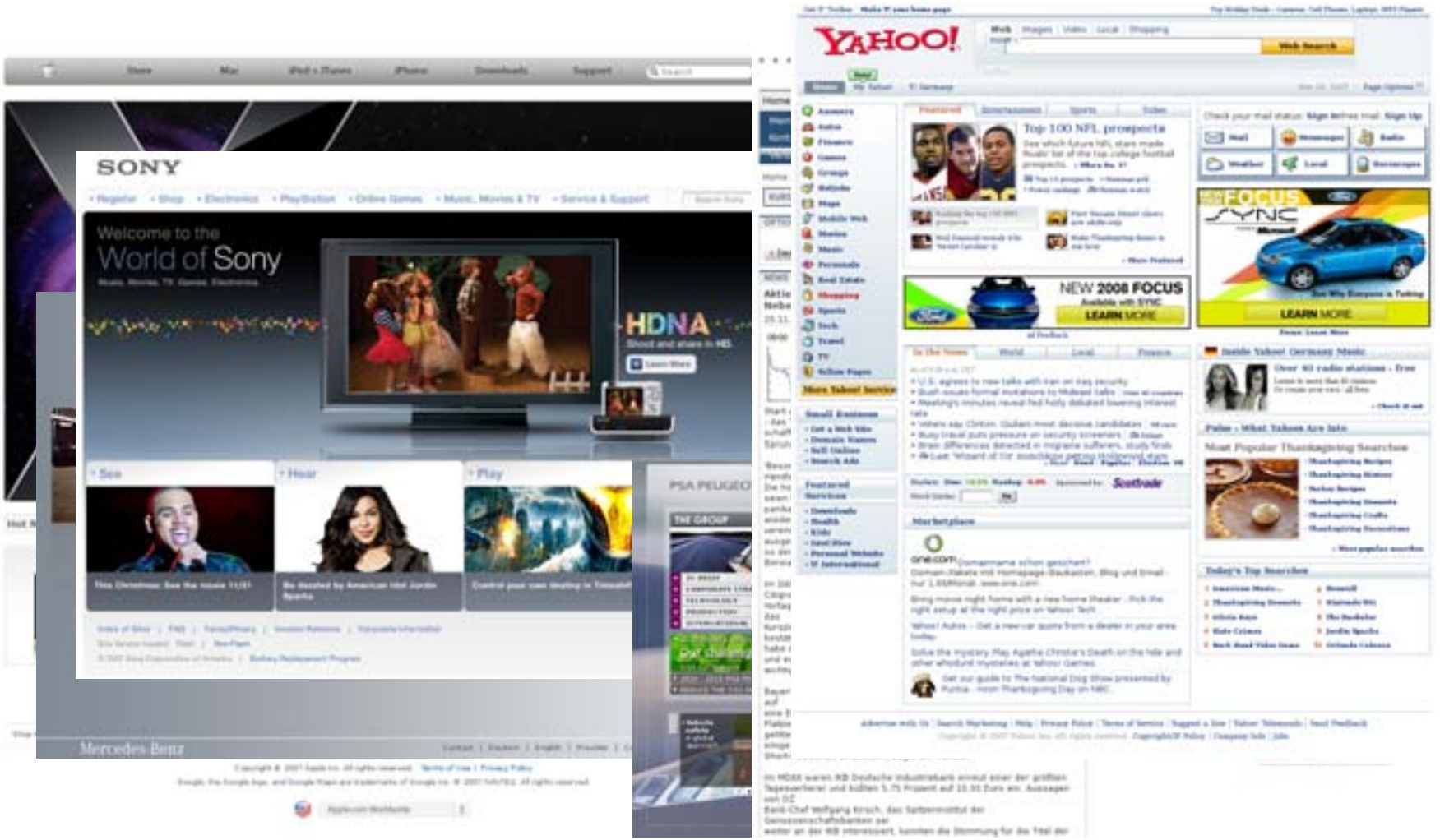


We all know what PHP is...

We all know what PHP is...

We all know what PHP is...

We all know what PHP is...



History of PHP

Timeline

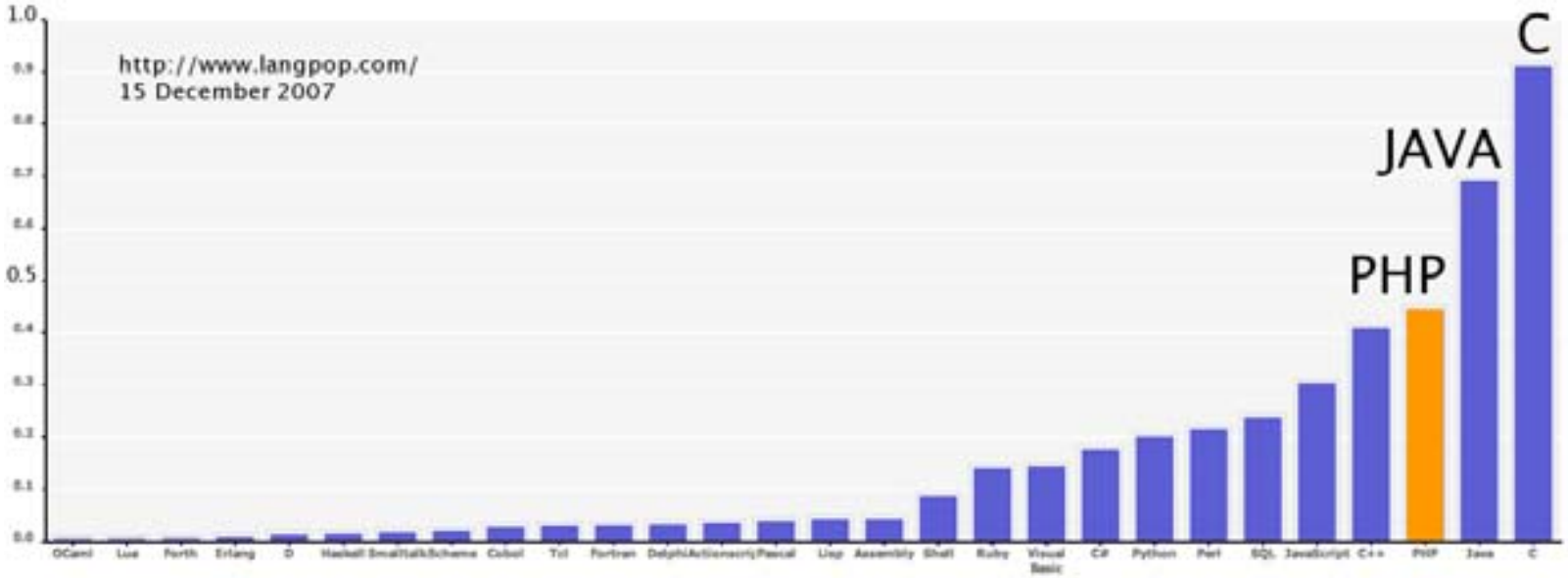
- 1995 "Personal Home Page Tools"
  - Developed by Rasmus Lerdorf
- 1996 PHP: Hypertext Preprocessor Version 2
  - Already used by approx. 1% of all domains (in 1997)
- 1998 PHP 3
  - Available on 10% of all Web servers (1998)
- 2000 PHP 4
  - Zend Engine released, first rudimentary object model
- 2004 PHP 5
  - New object model, improved XML, database abstraction
- PHP 6 is on the way
  - Unicode support, legacy features dropped

Application Domains of PHP

- Server-side scripting - Traditional domain of PHP
- Command-line scripting, partly replacing Perl, Python, awk, or shell scripting

```
php -r "sleep(7*60); echo chr(7);"
```

- Bindings to GUI libraries, such as GTK+, Qt and Win32



Outline

Language Features of PHP

Object Model

- Dynamically typed
- PHP4 has already offered an object model
  - Objects were handled like primitive types
  - Passed by value
- PHP5 has reintroduced classes and interfaces
  - Objects are passed by reference
  - Visibilities
  - Modifiers: `static`, `final`, `abstract`
- Garbage collection basically using reference counting
- Type hinting

```php
interface iDog {
    public function setName($name);
    public function sit();
}
abstract class Dog implements iDog {
    protected $name = "unnamed dog";
    public function setName($name) {
        $this->name = $name;
    }
    final public function sit() {
        echo "$this->name sat down, barking: ";
        echo  $this->bark();
    }
    protected function bark() {
        print "WOOF";
    }
}
class Poodle extends Dog implements iDog {
    final protected function bark() {
        echo " wiif ";
    }
}
$b = new Poodle();
$b->setName("small poodle");
$b->sit();
```

Magic Methods

Dynamic Object Properties

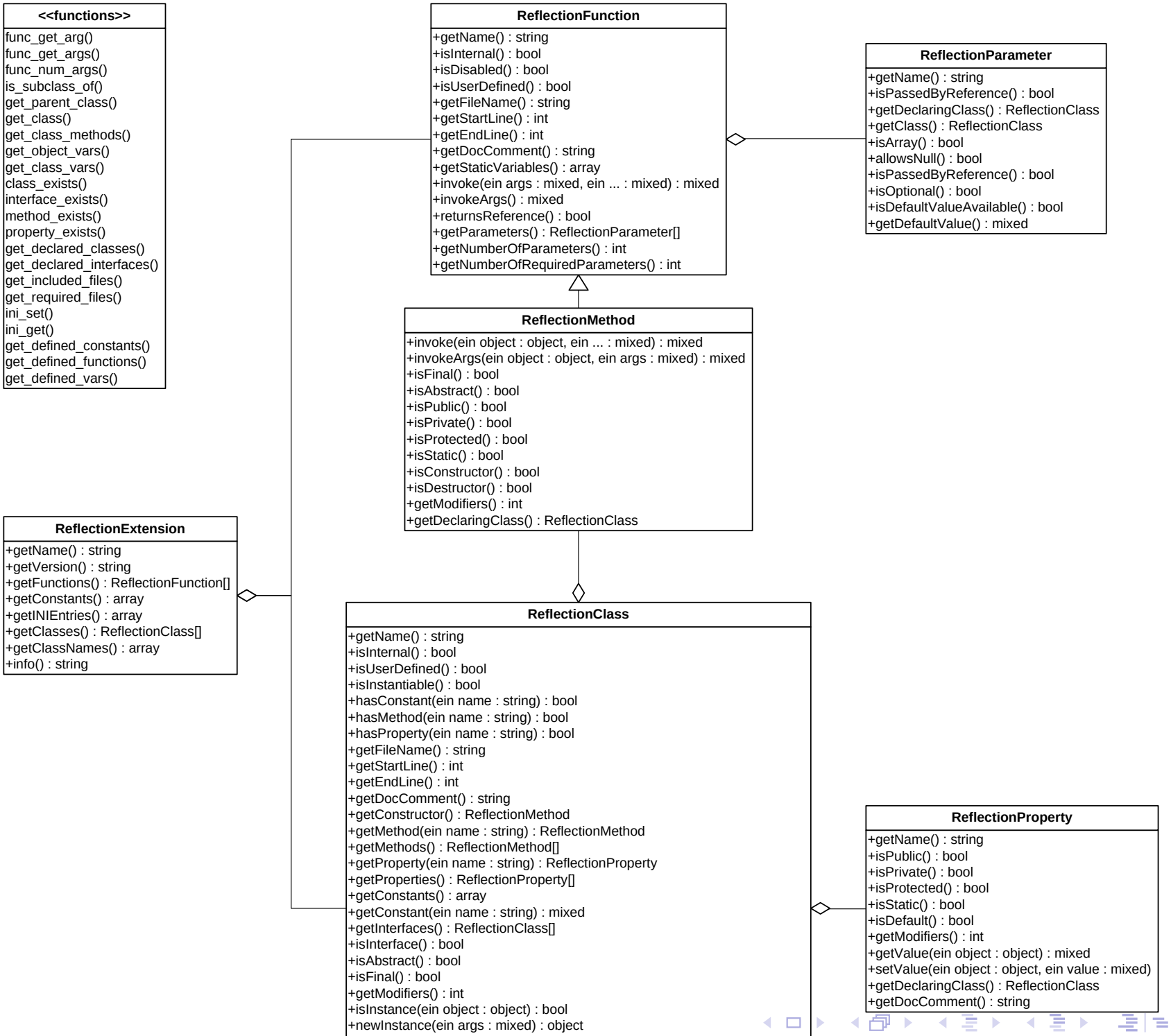
- `__get()`, `__set()`
  - Called when property to get or to set is not available
  - May be overloaded to enforce different behavior
- `__isset()`: tests whether a variable is set or nil
- `__unset()`: destroys the values of certain properties
- `__clone()`: defines how an instance is to be duplicated

- `__toString()`: describes a class' string representation
- `__sleep()`, `__wakeup()`, `__set_state()`

Magic Methods

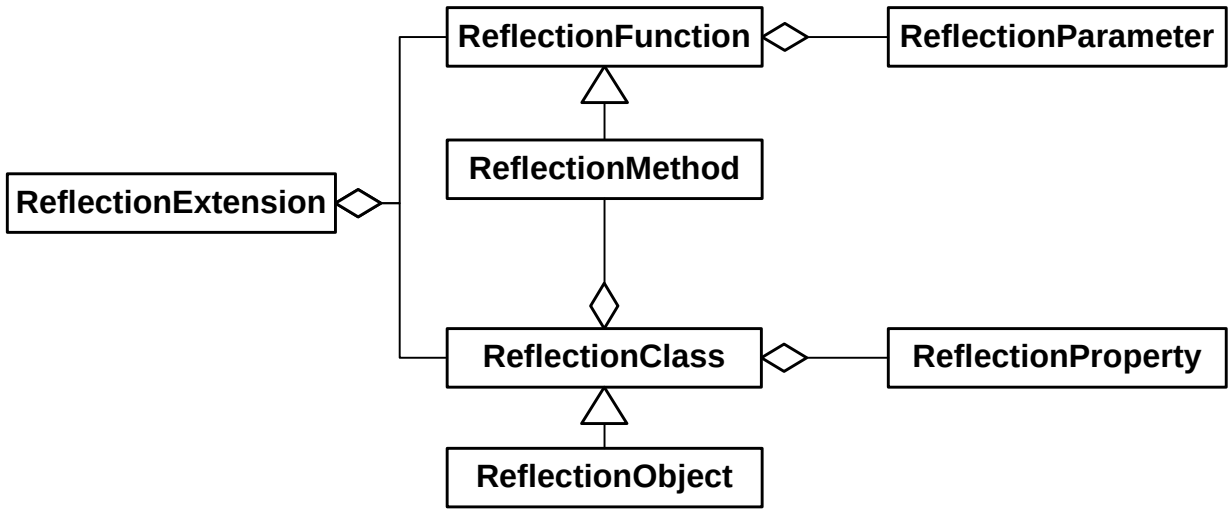
- `__autoload()`
  - Used to load all necessary classes once
  - Saves parse time and memory
- `__call()`
  - Captures the invocation of non-existing methods
  - Behavior can be adjusted for every class
  - Similar to Smalltalk's `#doesNotUnderstand: aMessage`

```
class s {
  private $default = "default answer";
  public function __call($name, $arguments) {
    echo "Method $name called:\n";
    var_dump($arguments);
    return $this->default;
  }
}
```

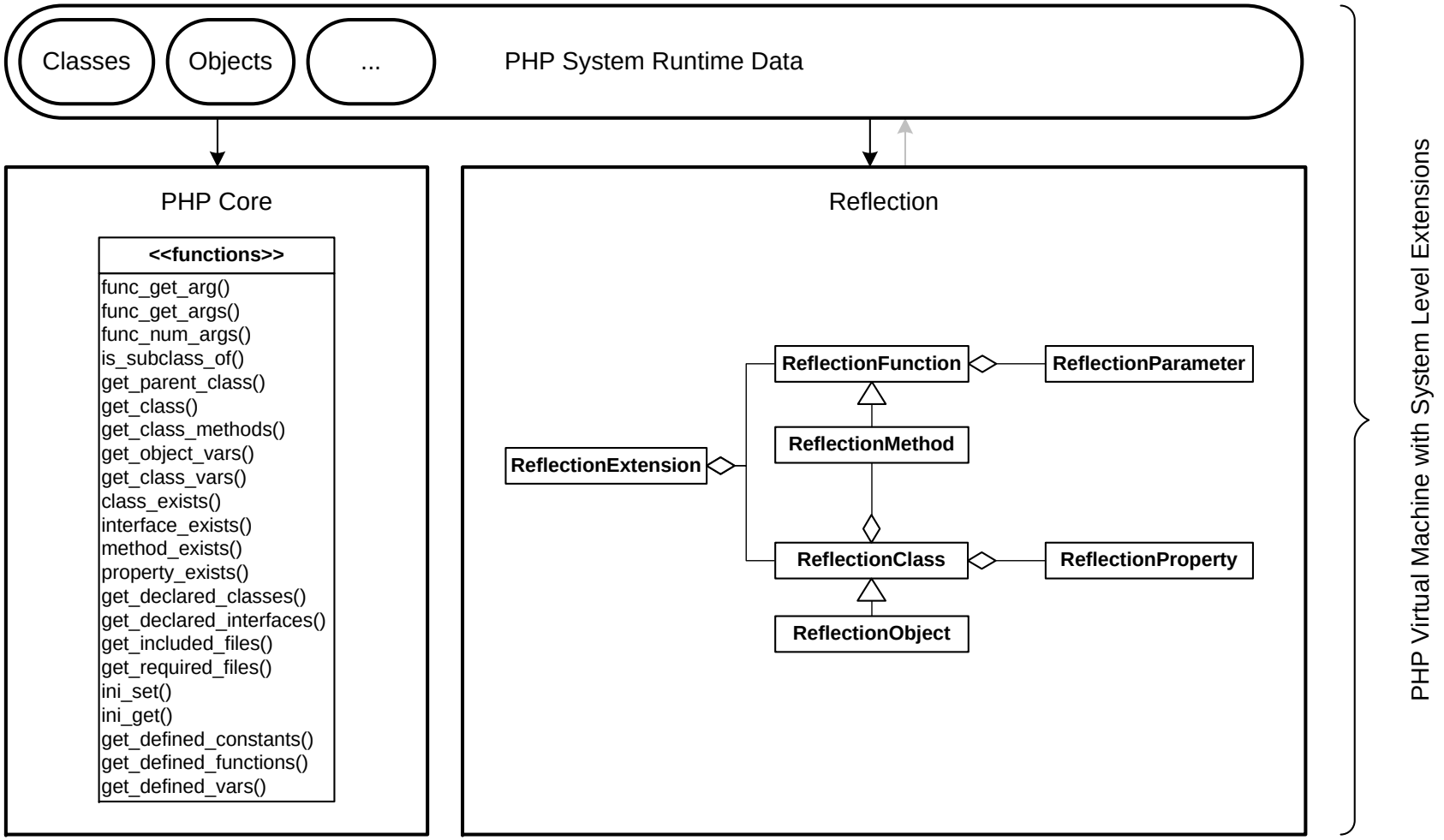

<<functions>>
func_get_arg()
func_get_args()
func_num_args()
is_subclass_of()
get_parent_class()
get_class()
get_class_methods()
get_object_vars()
get_class_vars()
class_exists()
interface_exists()
method_exists()
property_exists()
get_declared_classes()
get_declared_interfaces()
get_included_files()
get_required_files()
ini_set()
ini_get()
get_defined_constants()
get_defined_functions()
get_defined_vars()
ReflectionFunction
+getName() : string
+isInternal() : bool
+isDisabled() : bool
+isUserDefined() : bool
+getFileName() : string
+getStartLine() : int
+getEndLine() : int
+getDocComment() : string
+getStaticVariables() : array
+invoke(ein args : mixed, ein ... : mixed) : mixed
+invokeArgs() : mixed
+returnsReference() : bool
+getParameters() : ReflectionParameter[]
+getNumberOfParameters() : int
+getNumberOfRequiredParameters() : int
ReflectionParameter
+getName() : string
+isPassedByReference() : bool
+getDeclaringClass() : ReflectionClass
+getClass() : ReflectionClass
+isArray() : bool
+allowsNull() : bool
+isPassedByReference() : bool
+isOptional() : bool
+isDefaultValueAvailable() : bool
+getDefaultValue() : mixed
ReflectionMethod
+invoke(ein object : object, ein ... : mixed) : mixed
+invokeArgs(ein object : object, ein args : mixed) : mixed
+isFinal() : bool
+isAbstract() : bool
+isPublic() : bool
+isPrivate() : bool
+isProtected() : bool
+isStatic() : bool
+isConstructor() : bool
+isDestructor() : bool
+getModifiers() : int
+getDeclaringClass() : ReflectionClass
ReflectionExtension
+getName() : string
+getVersion() : string
+getFunctions() : ReflectionFunction[]
+getConstants() : array
+getINIEntries() : array
+getClasses() : ReflectionClass[]
+getClassNames() : array
+info() : string
ReflectionClass
+getName() : string
+isInternal() : bool
+isUserDefined() : bool
+isInstantiable() : bool
+hasConstant(ein name : string) : bool
+hasMethod(ein name : string) : bool
+hasProperty(ein name : string) : bool
+getFileName() : string
+getStartLine() : int
+getEndLine() : int
+getDocComment() : string
+getConstructor() : ReflectionMethod
+getMethod(ein name : string) : ReflectionMethod
+getMethods() : ReflectionMethod[]
+getProperty(ein name : string) : ReflectionProperty
+getProperties() : ReflectionProperty[]
+getConstants() : array
+getConstant(ein name : string) : mixed
+getInterfaces() : ReflectionClass[]
+isInterface() : bool
+isAbstract() : bool
+isFinal() : bool
+getModifiers() : int
+isInstance(ein object : object) : bool
+newInstance(ein args : mixed) : object
ReflectionProperty
+getName() : string
+isPublic() : bool
+isPrivate() : bool
+isProtected() : bool
+isStatic() : bool
+isDefault() : bool
+getModifiers() : int
+getValue(ein object : object) : mixed
+setValue(ein object : object, ein value : mixed)
+getDeclaringClass() : ReflectionClass
+getDocComment() : string

PHP5 Reflection API

<<functions>>
func_get_arg()
func_get_args()
func_num_args()
is_subclass_of()
get_parent_class()
get_class()
get_class_methods()
get_object_vars()
get_class_vars()
class_exists()
interface_exists()
method_exists()
property_exists()
get_declared_classes()
get_declared_interfaces()
get_included_files()
get_required_files()
ini_set()
ini_get()
get_defined_constants()
get_defined_functions()
get_defined_vars()

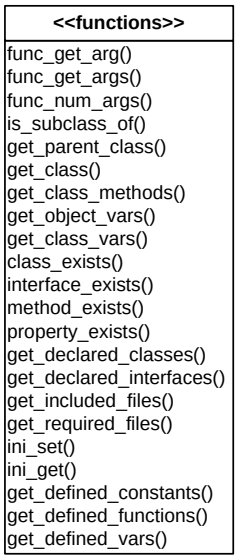


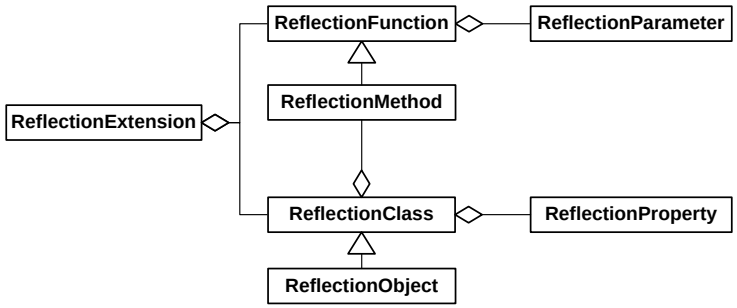
PHP5 Reflection API



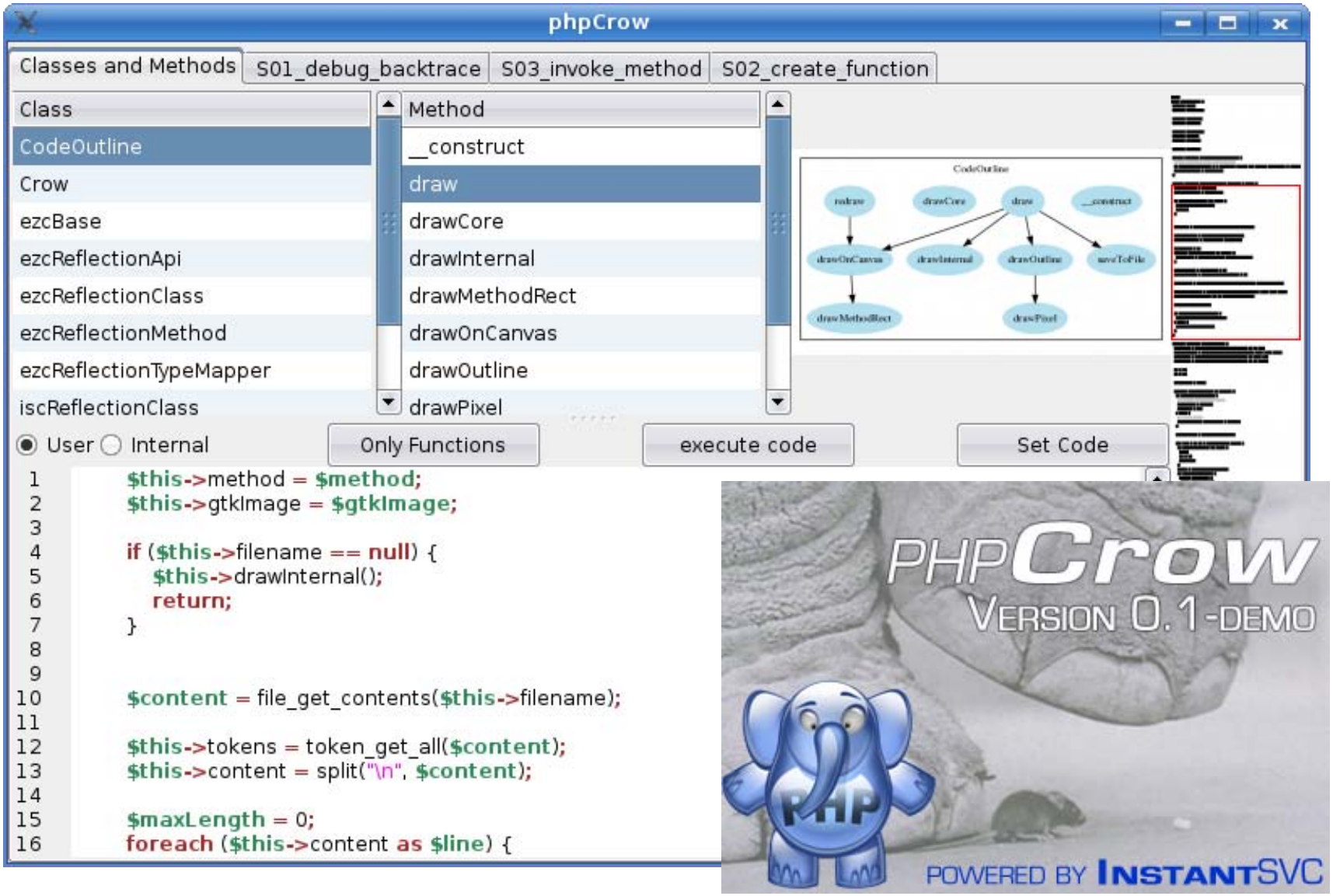
PHP5 Reflection API

- Wraps underlying VM data structures
- Provides introspection and method/function invocation capabilities
- Can be deactivated (to reduce VM memory footprint)
- Respects the *Mirrors* design principle [1] to a high degree





Reflection in Action



Reflection in Action: debug_backtrace

```
class DebugBacktrace {
    public function callee() {
        echo 'Caller of this Function was ',
            DebugBacktrace::getCaller(), ".\n";
    }
    public static function getCaller() {
        $trace = debug_backtrace();
        return $trace[2]['class'] . $trace[2]['type']
            . $trace[2]['function'];
    }
    public function test1() {
        $this->callee();
    }
    public static function test2($t) {
        $t->callee();
    }
}

$d = new DebugBacktrace();
$d->test1(1, 2, 3);
DebugBacktrace::test2($d);
```

Common Usage of Reflection in PHP

- XML/JSON to PHP mapping
- Frameworks for dependency injection and AOP
- PHPCallGraph –> visualization
- PHPUnit: Mockup objects, generation of test skeletons, code metrics

Example for Annotations

```
class Calculator {
    /**
     * @assert (0, 0) == 0
     * @assert (0, 1) == 1
     * @assert (1, 1) == 2
     */
    public function add($a, $b) { return $a + $b; }
}
```

Basic Intercession

Execute Generated Code

- `eval()`
- `create_function()`

Dynamically Call a Method or Function

- `call_user_function()`
- `call_user_function_array()`
- `call_user_method()`
- `call_user_method_array()`
- `$functionName($a, $b);`

Basic Intercession: `create_function`

`create_function`

```
$func = create_function('$foo,$bar', 'echo "CreatedFunction:
    $foo $bar\n";');
$func('Hello', 'World!');
var_dump(addslashes($func));

call_user_func($func, 'FOO', 'baz');

//common usage example
$av = array("the ", "a ", "that ", "this ");
array_walk($av, create_function('&$v,$k', '$v = $v . "mango";'))
    ;
print_r($av);
```

Basic Intercession: Invoke Methods

Invoke Methods

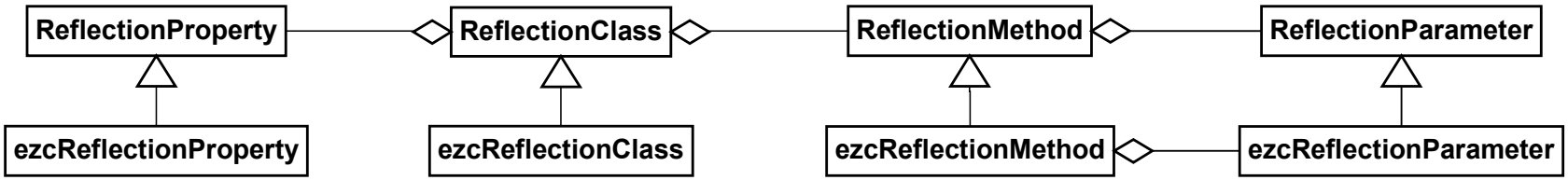
```
class InvokeTest {
  public function callMe() {
    echo "Yeah, you called me!\n";
  }
}

$o = new InvokeTest();
$class = new ReflectionClass($o);
$method = $class->getMethod('callMe');
$method->invoke($o, 'callMe');

function fooFunc($bar) {
   echo "$bar\n";
}

$func = 'fooFunc';
$func('Hello World');
```

Annotations in PHP

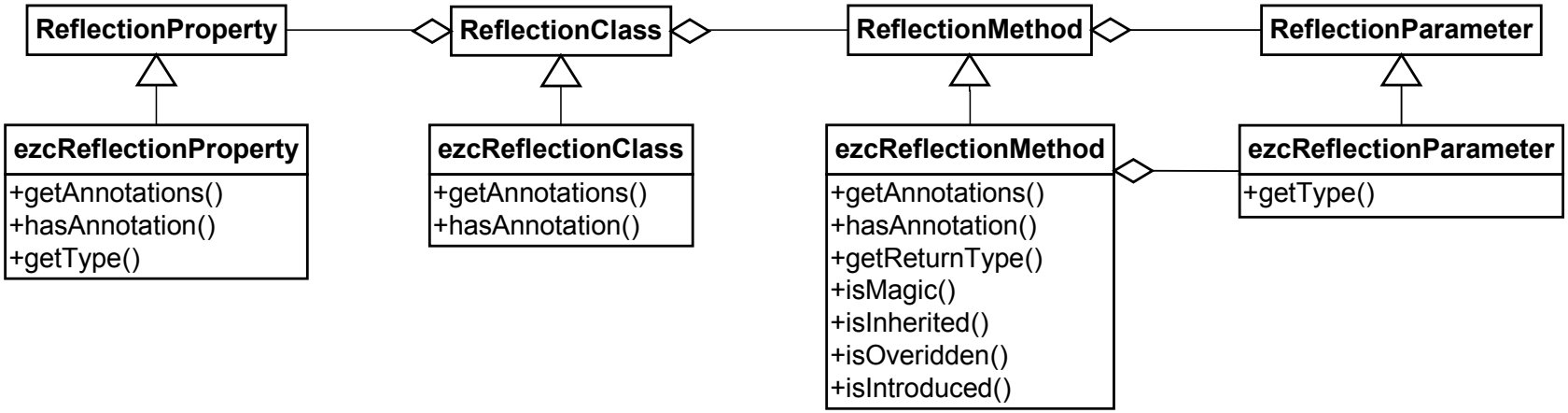
Class with Type Annotations

```
class HelloWorld {

    /**
     * @var string
     */
    private $hello = 'Hello';

    /**
     * @param string $name
     * @return string
     */
    public function getGreeting($name) {
        return "$this->hello $name!";
    }

}
```

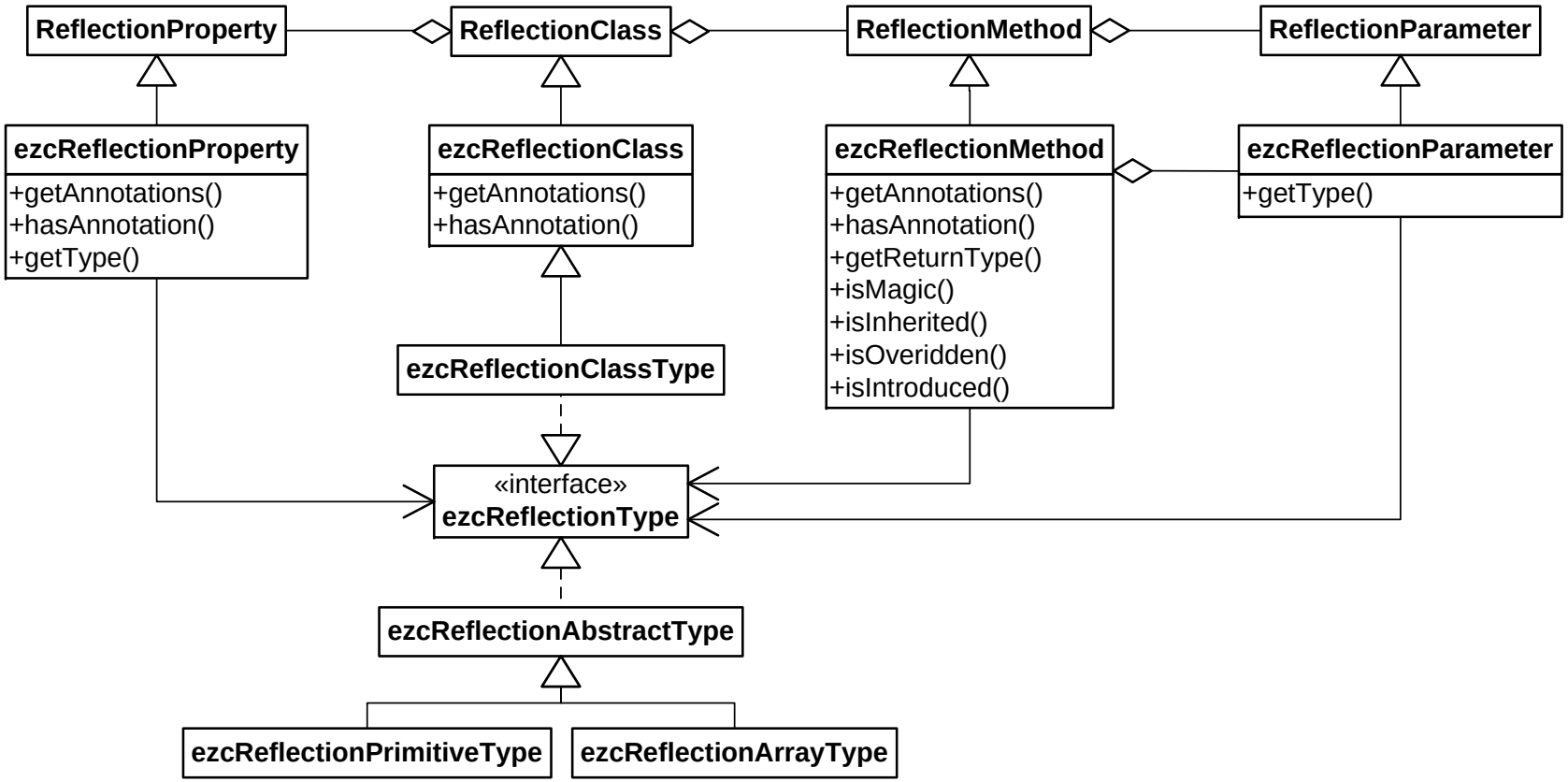
eZ Components: Reflection Component



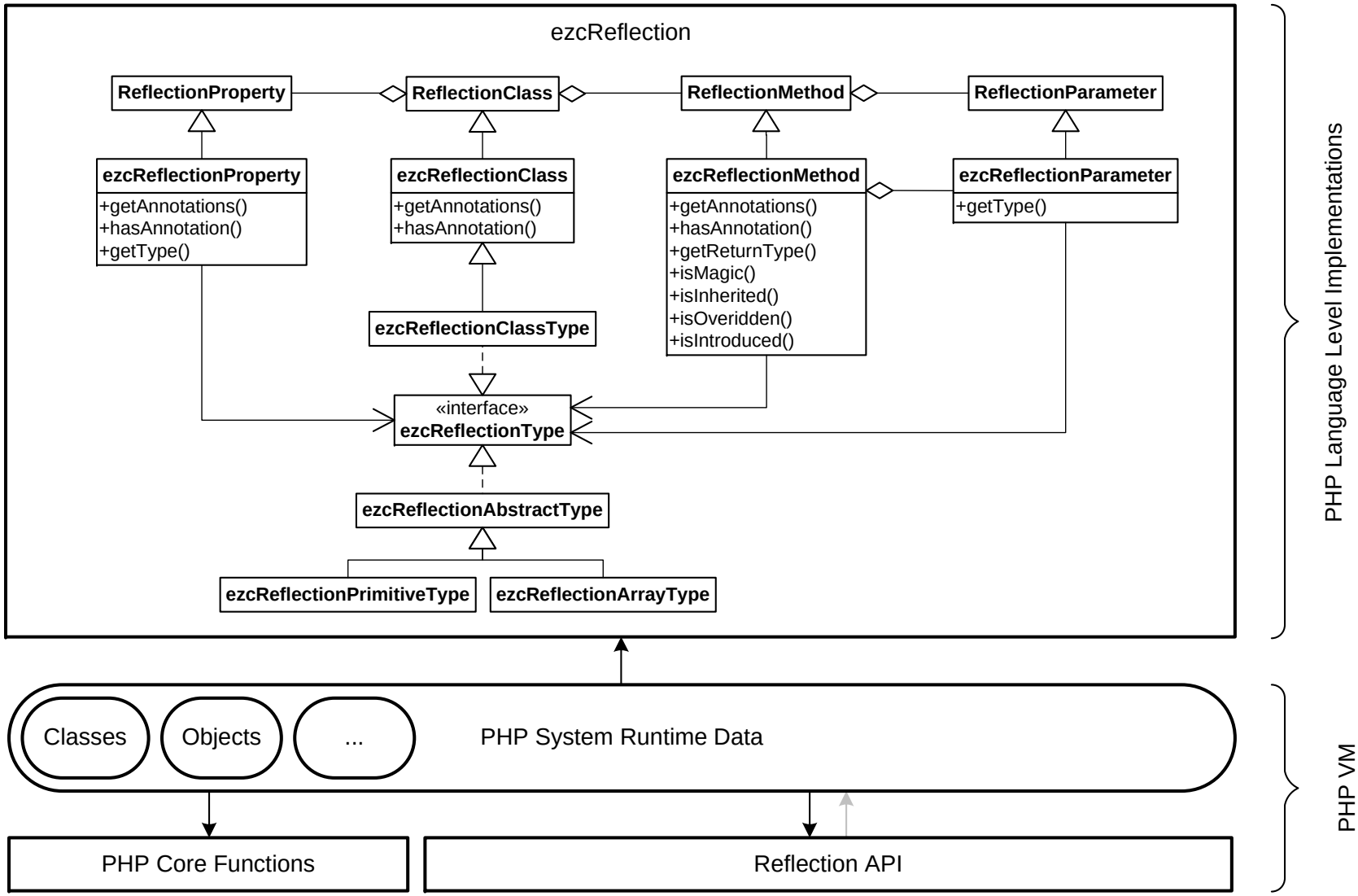
eZ Components: Reflection Component



eZ Components: Reflection Component



eZ Components: Reflection Component



eZ Components: Reflection Component

Feature Summary

- Introduces support for PHPDoc-style annotations
- Type annotations are accessible at runtime for introspection
- No runtime constraints
- Can leverage other reflection implementations as a data source, e.g. StaticReflection
- Used e.g. for WSDL generation

Annotations for WSDL Generation

Hello World Web Service

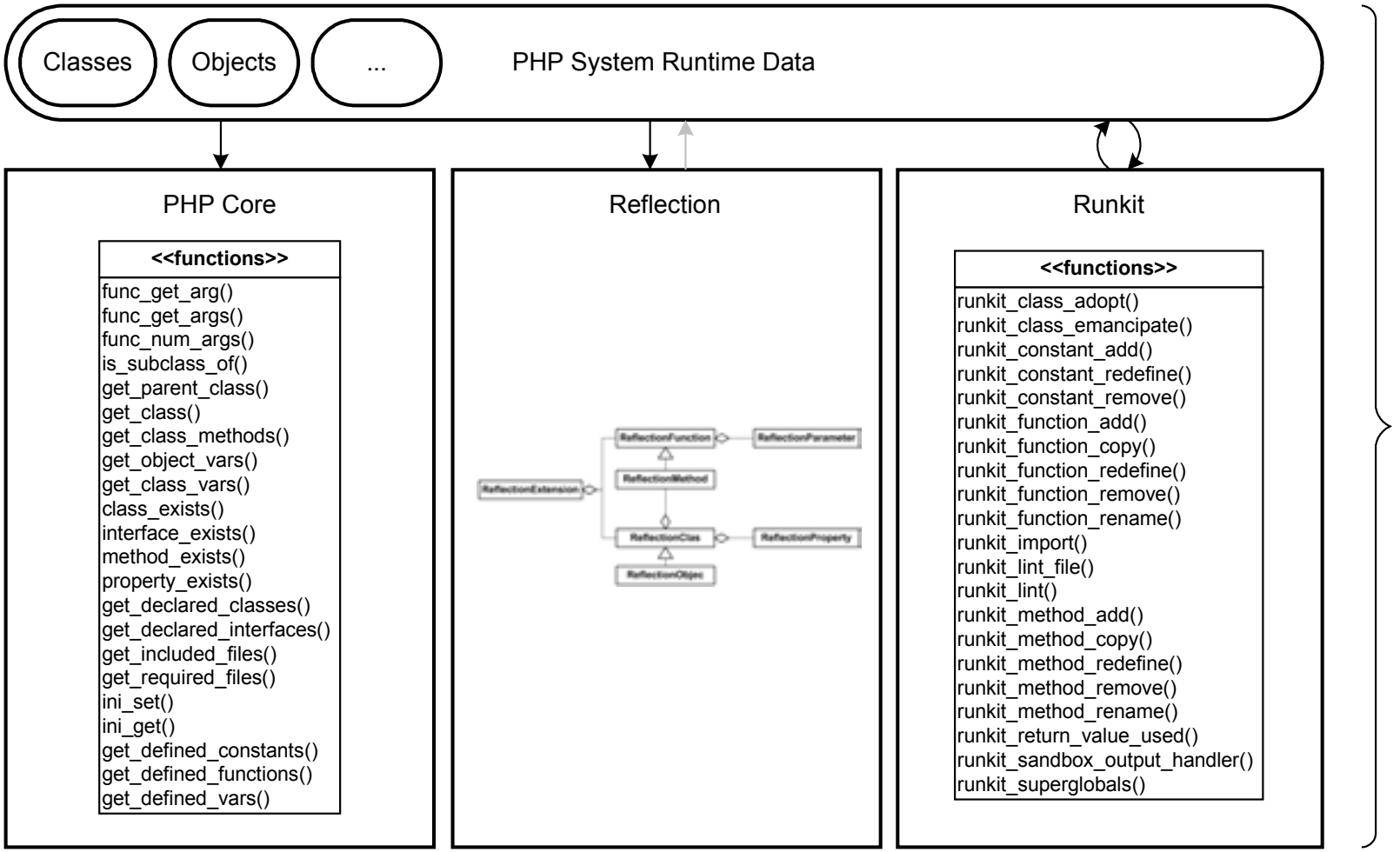
```
/**
 * @webservice
 */
class HelloWorld {

    /**
     * @param string $name
     * @return string
     * @webmethod
     */
    public function getGreeting($name) {
        return "Hello $name!";
    }

}
```

Doing Real Intercession: The Runkit Extension

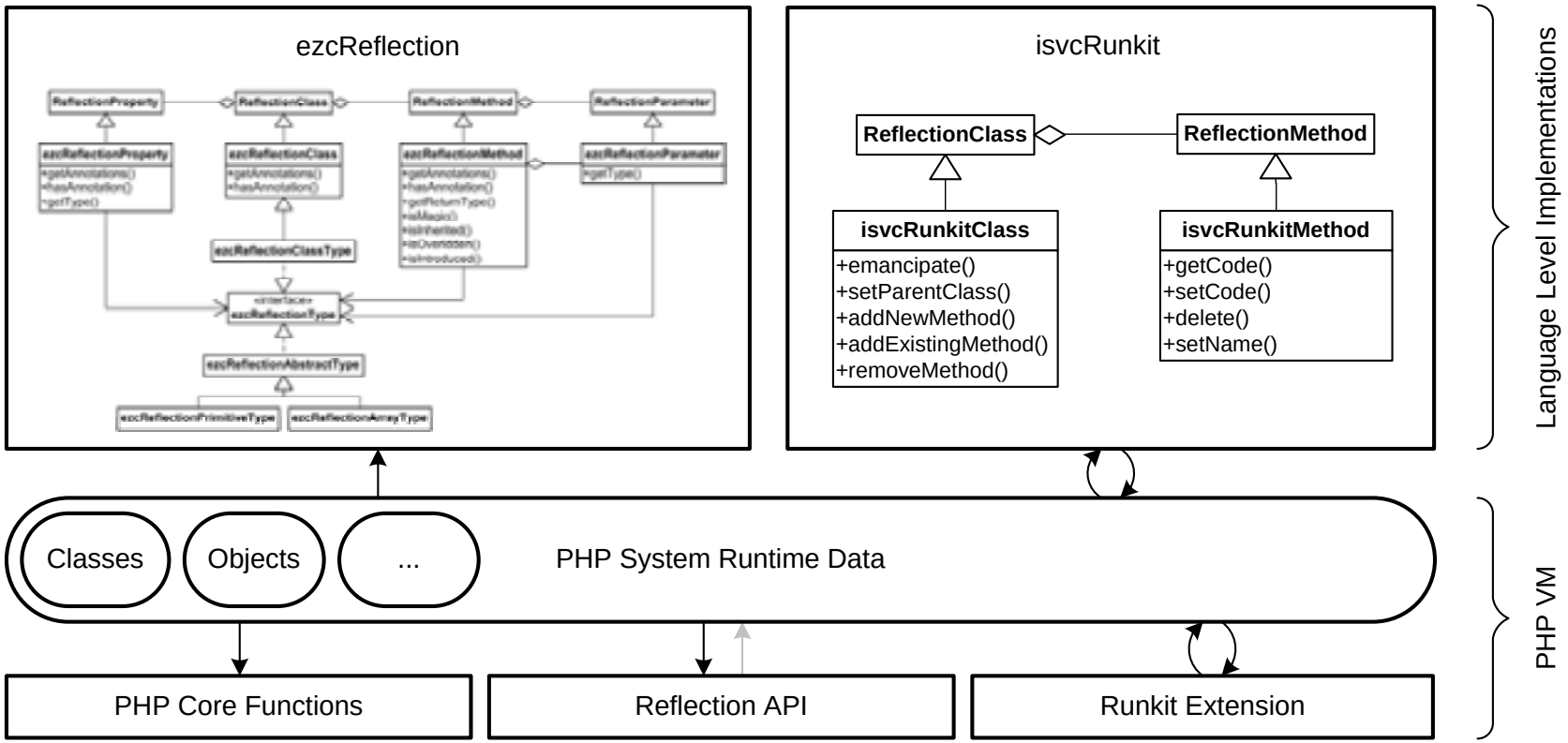
Features of Runkit

- Modify functions and methods
  - add, copy, redefine, rename and remove
- Modify constants and class constants
  - add, redefine and remove
- Modify the inheritance structure
  - adopt and emancipate classes
- Check if a return value will be used
- Sandbox classes
- Lint functions

Doing Real Intercession: The Runkit Extension



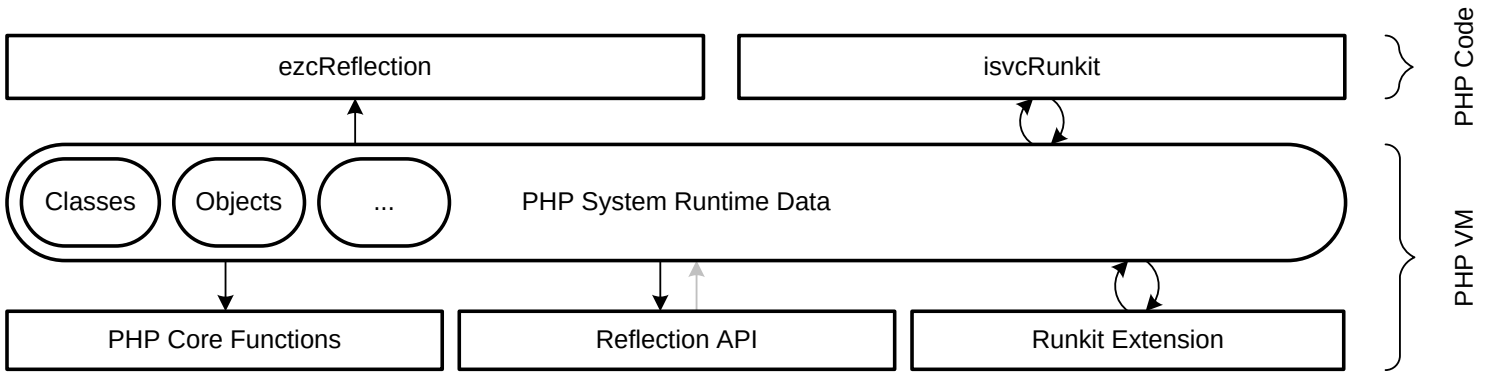
Object-oriented Runkit



Other Metaprogramming Extensions for PHP

- Advanced PHP Debugger (APD)
  - `override_function()`
  - `rename_function()`
  - Similar features provided by Runkit
- Tokenizer and `parse_tree`
  - Not reflective but very helpful for metaprogramming
- Several aspect-oriented programming (AOP) frameworks
  - Static aspect weaving
  - Using PHPDoc-style annotations and sometimes XML

Conclusion



- PHP has many reflective features
  - MOP divided into object-oriented and procedural API
  - Additional features provided by extensions
- Main use case: Web applications
  - Extensive intercession makes not much sense for the usual PHP execution model
  - In general more useful in long running programs, e.g. GTK+, QT, MFC or command line applications

For Further Reading I


Gilad Bracha and David Ungar.
Mirrors: design principles for meta-level facilities of object-oriented programming languages.
In *OOPSLA '04: Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 331–344, New York, NY, USA, 2004. ACM.


Sara Golemon.
Runkit.
PHP Extension, The PHP Group, January 2008.
`http://pecl.php.net/package/runkit`.

For Further Reading II


Stefan Marr and Falko Menge.
ezcReflection.
SVN Repository, eZ Components, January 2008.
http://svn.ez.no/svn/ezcomponents/experimental/Reflection.


PHP Documentation Group.
PHP Manual.
Documentation, November 2007.
http://www.php.net/manual/en/language.oop5.reflection.php.

For Further Reading III

The PHP Group.
PHP Extension Community Library.
Project Site, January 2008.
http://pecl.php.net/.

David Welton.
Programming Language Popularity.
Web Site, DedaSys LLC, 2008.
http://www.langpop.com/.