P8, IBM's PHP on Java Virtual Machine™ Project

Rob Nicholson

rob_nicholson@uk.ibm.com

Why Build PHP on the JVM?

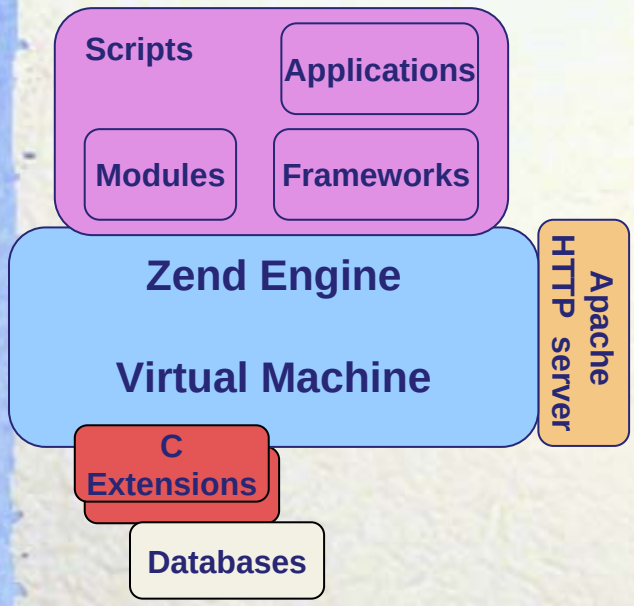
- **20M+ web domains use PHP**
- **3M+ Programmers know PHP**
- **Huge repository of reusable modules, snippets, extensions.**
- **Easy language to learn -> Mashups**
- **Language has evolved to be easy to use**

- Same-process interaction Java <-> PHP.
- Combine Java and PHP assets.
- Combine Java and PHP programmers.
  - Data sharing without copies.
  - Extend Java with PHP.
- Benefit from vast investment in Java VM
- IBM WebSphere sMash has **Groovy** +P8

TIOBE Programming Community Index (Sep 2008)

Position Sep 2008	Position Sep 2007	Delta in Position	Programming Language	Ratings Sep 2008
1	1	=	Java	20.715%
2	2	=	C	15.379%
3	5	⇈	C++	10.716%
4	3	↓	(Visual) Basic	10.490%
5	4	↓	PHP	9.243%
6	8	⇈	Python	5.012%
7	6	↓	Perl	4.841%
8	7	↓	C#	4.334%

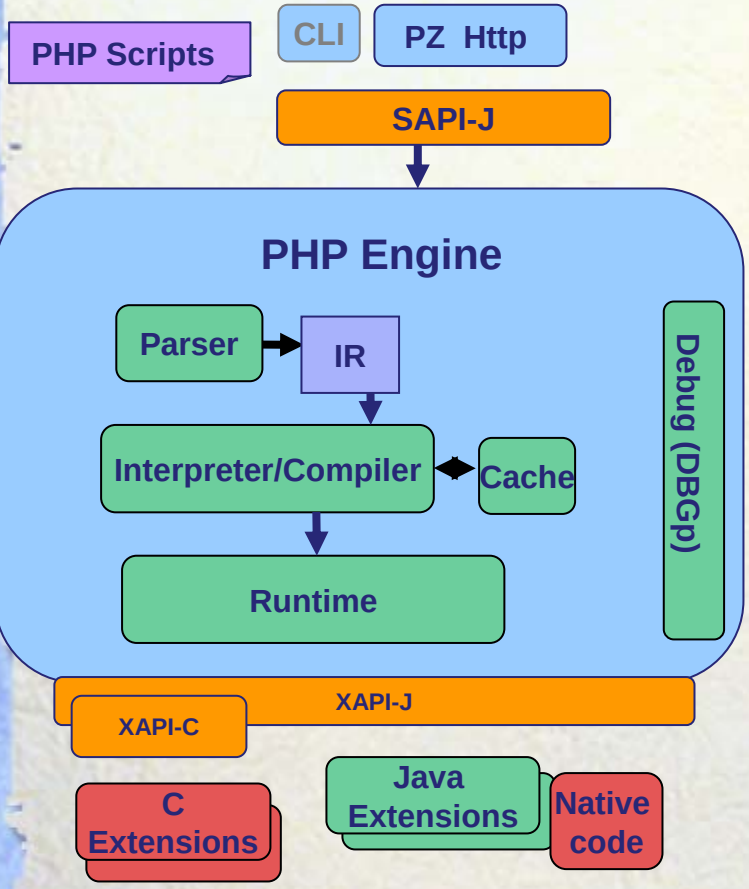
What is PHP?



- Procedural and OO language.
- Engine and Extensions Implemented in C.
- Frameworks, Modules, Applications implemented in PHP.
- Large and active open source communities.

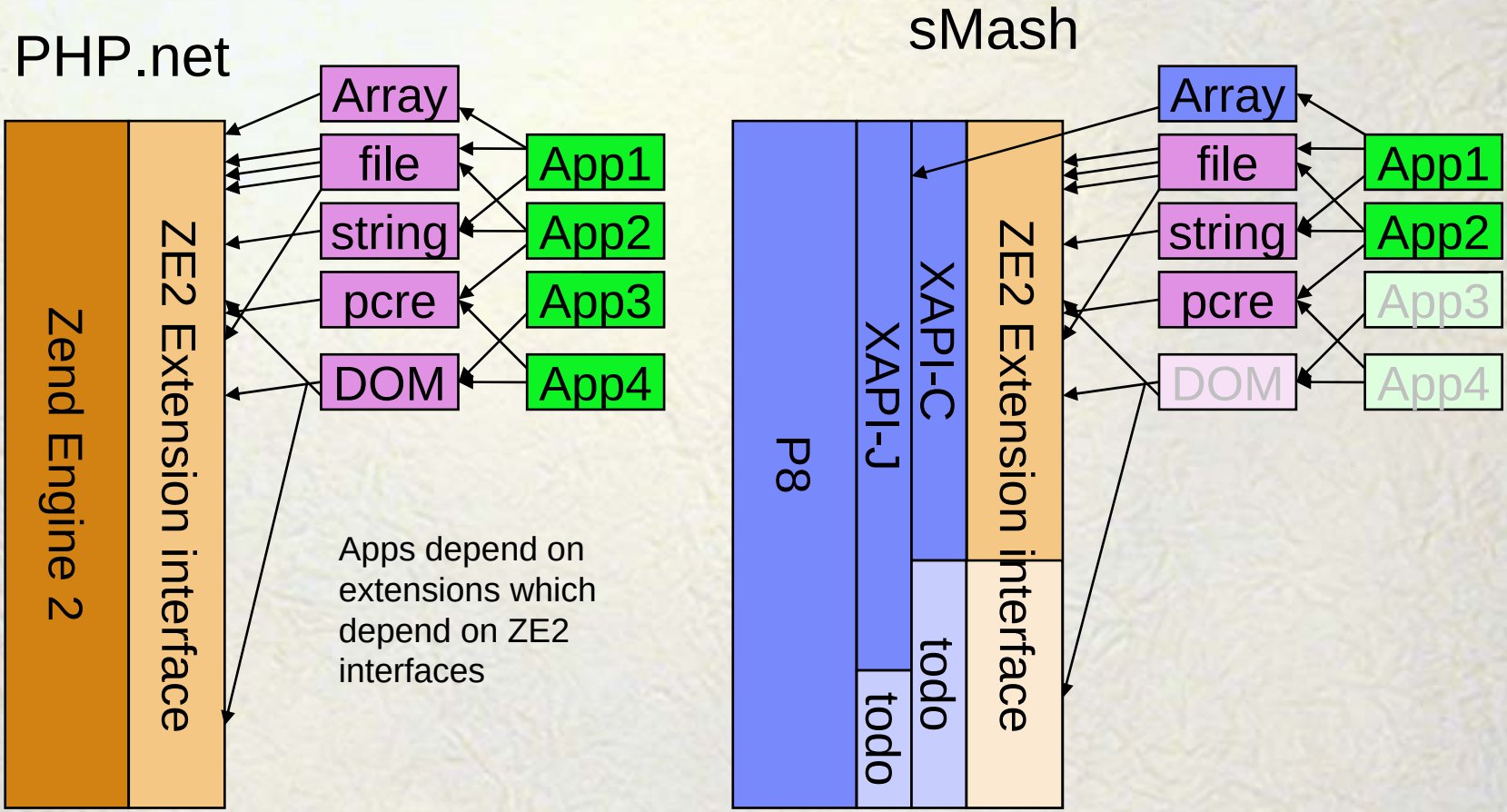
- No specification
- Incomplete documentation
- Incomplete tests
- No Unicode
- **Reference Implementation based language**

What is P8?



- **PHP 5 on Java 5 SE or later.**
- Hand crafted lexer, LPG generated parser.
- Started as Interpreter -> transitioning to compiler.
  - Maintain the illusion of interpreter.
- Extensibility via XAPI
  - XAPI-C for C extensions from php.net
  - XAPI-J for Java extensions, native libraries invoked over JNI and Project Zero interface
- Java Bridge
  - Extend Java classes in PHP.
  - Implement Java Interfaces in PHP.
  - PHP proxies over Java Classes
- Debug using via DBGp using Eclipse with PDT

How applications are supported.



PHP Characteristics

- Nothing persists request to request
- Dynamic runtime inclusion:
  - Many versions of function foo() or class bar
    - In scope version depends on includes at runtime
- Dynamically declared/used symbols
  - Function/classes are conditionally declared
  - Variables may indirectly referred using string values
- Both copy semantics and references allowed
  - Arrays and strings are copied for each assignment to any variable
  - References can be created and used almost transparently
- Heavy use of foreign language functions
  - Heavy calculations are often done in libraries written in C/C++
  - So-called extensions written in C help PHP's rich language constructs
- OO PHP becoming popular

  ~ 10x slower than equivalent Java
- Functions of unconstrained length
  - Exceed JVM method length limit

Typical PHP

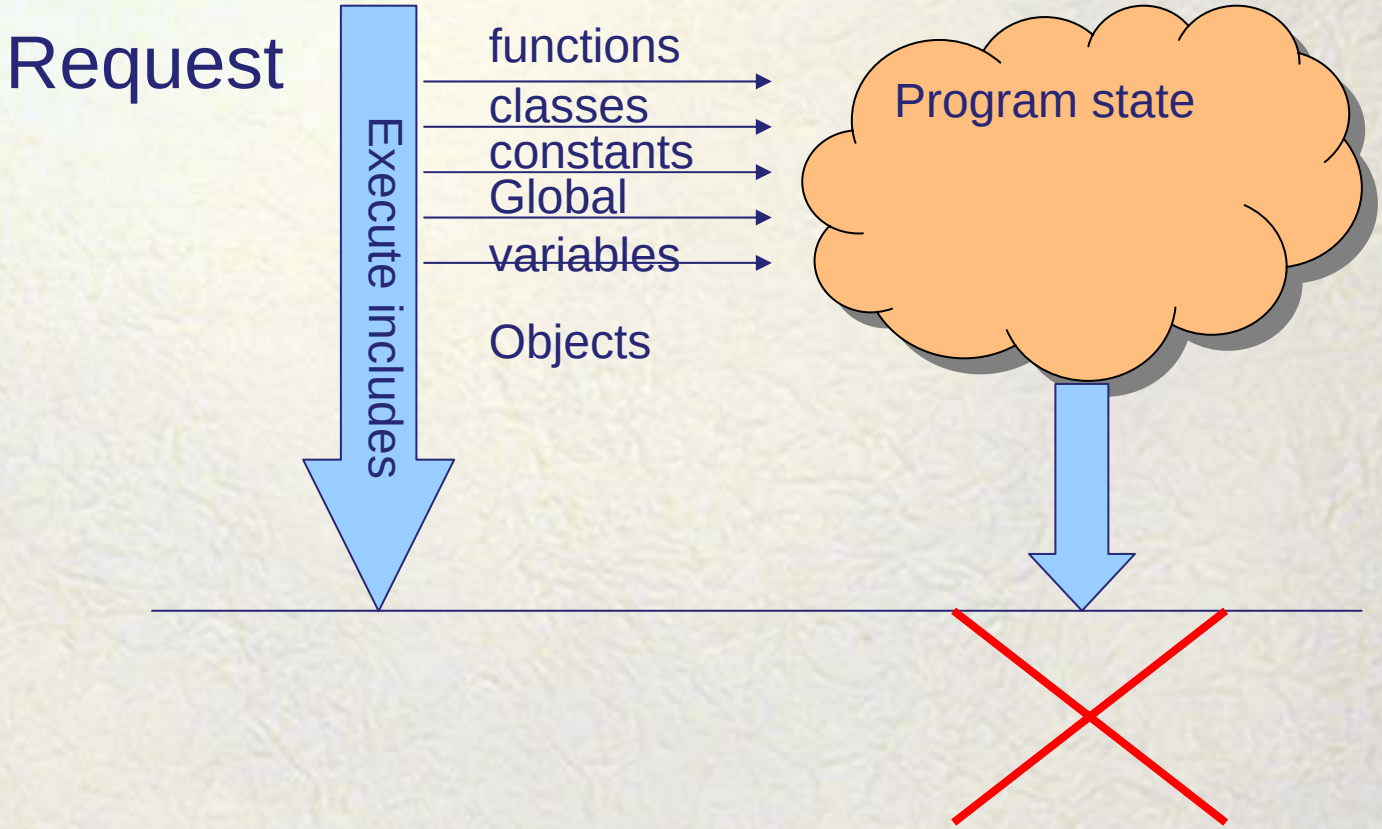
index.php:

```
<?php
        include “inc1.php”;
        $var=1;
        foo (“bar.php”,$var);
        include $var;
?>
```

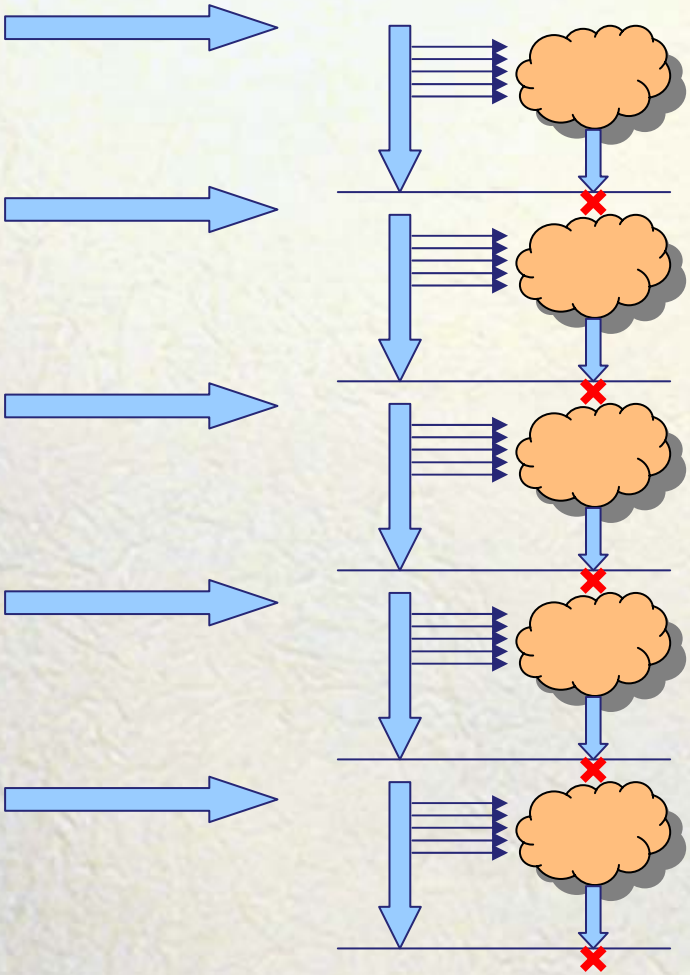
inc1.php:

```
<?php
        function foo($a, &$b) {  $b=$a; }
?>
```

PHP Request Processing.



Shared nothing architecture



Compiling PHP Functions.

- Every function creates a class with single known method.

```
public class Bytecode1_rob_php extends AbstractBytecode
{
    public PHPValue run(Runtime runtime)
    {
            // currently still fetching arguments interpreter style
            return null;
    }
}
```

- Function table is Hash “Name”=>Invocable
- Invoke using InvokeVirtual
  - Use receiver to select which function to invoke.

Function Invoke w/o methodHandles

```
<?php
  function foo(){}
  foo();
  bar();
?>
```

```
public class Bytecode1_rob_php extends AbstractBytecode
{
    private static Invocable invocable1;
    private Invocable invocable2;

    public PHPValue run(Runtime runtime)
    {
        if(invocable1 == null)
            invocable1 = Op.jhGetInvocable(runtime, "foo");
        Op.jhCALL(null, invocable1, runtime, true);
            // PHPValue returnVal = invocable1.bytecode.run(runtime);
        if(invocable2 == null)
            invocable2 = Op.jhGetInvocable(runtime, "bar");
        Op.jhCALL(null, invocable2, runtime, true);
            // PHPValue returnVal = invocable2.bytecode.run(runtime);
        return null;
    }
}
```

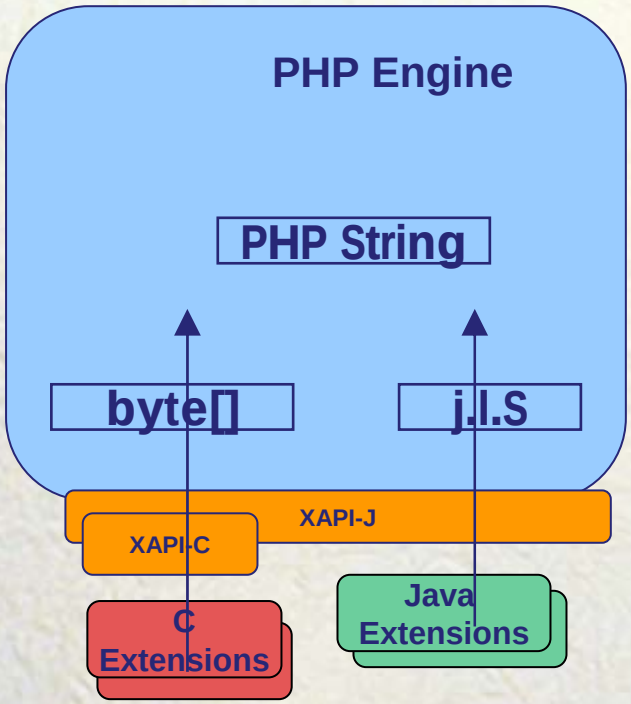
JSR292

- Initially use MethodHandles to avoid creating separate class for every PHP function.
- Not yet clear that InvokeDynamic helps for PHP Function Invocation.
- Will use InvokeDynamic for PHP method invocation.

PHPValue

- PHP native VM heavily exploits ability to change types at runtime.
- PHP values change types during arithmetic.
- PHP retains references to values.
  - Hard to devise efficient Value representation w/o ability to change class of object at runtime.
- Currently we use indirection to resolve.
  - Creates heap pressure.

Unicode



- In PHP5, a string is a sequence of bytes.
- App programmer knows encoding (if any)
- So….Strings hold characters AND jpegs.
- Runtime does not know the difference.

BUT

- P8 wants to deal seamlessly with Java Libraries.
- C extensions from PHP.net use byte[].

SO

- P8 PHPString can be either j.l.S or byte[]
- On demand conversion using runtime encoding.
- Comparisons coerce byte[] to j.l.S **iff** no decoding errors.

Questions…?