Introduction to Iterators

Marcus Börger

LinuxTag 2004 Karlsruhe

Introduction to Iterators

☑ What are Iterators

☑ The basic concepts

What are Iterators

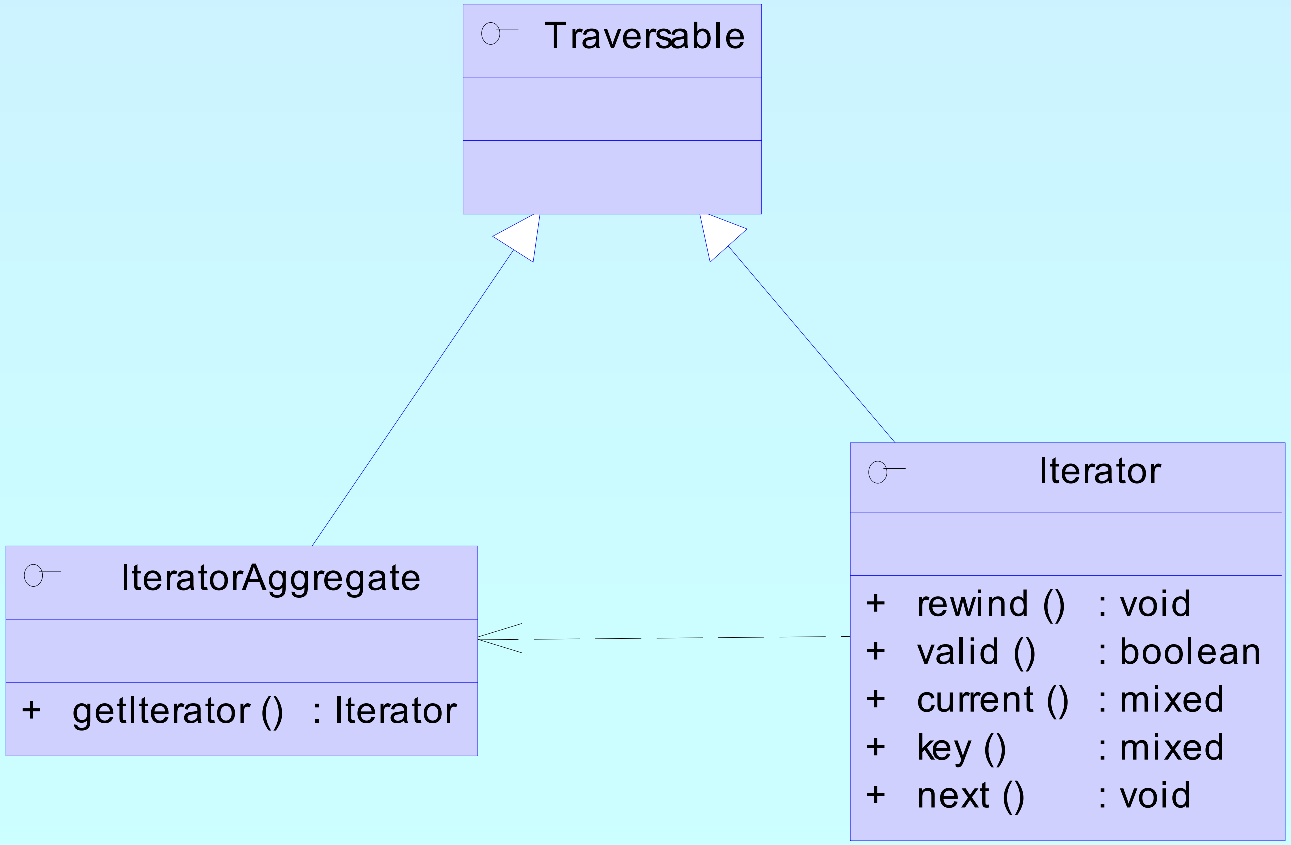
- ☑ Iterators are a concept to iterate anything that contains other things. Examples:
  - ☑ Values and Keys in an array
  - ☑ Text lines in a file
  - ☑ Database query results
  - ☑ Files in a directory
  - ☑ Elements or Attributes in XML
  - ☑ Bits in an image
  - ☑ Dates in a calendar range
- ☑ Iterators allow to encapsulate algorithms

The basic concepts

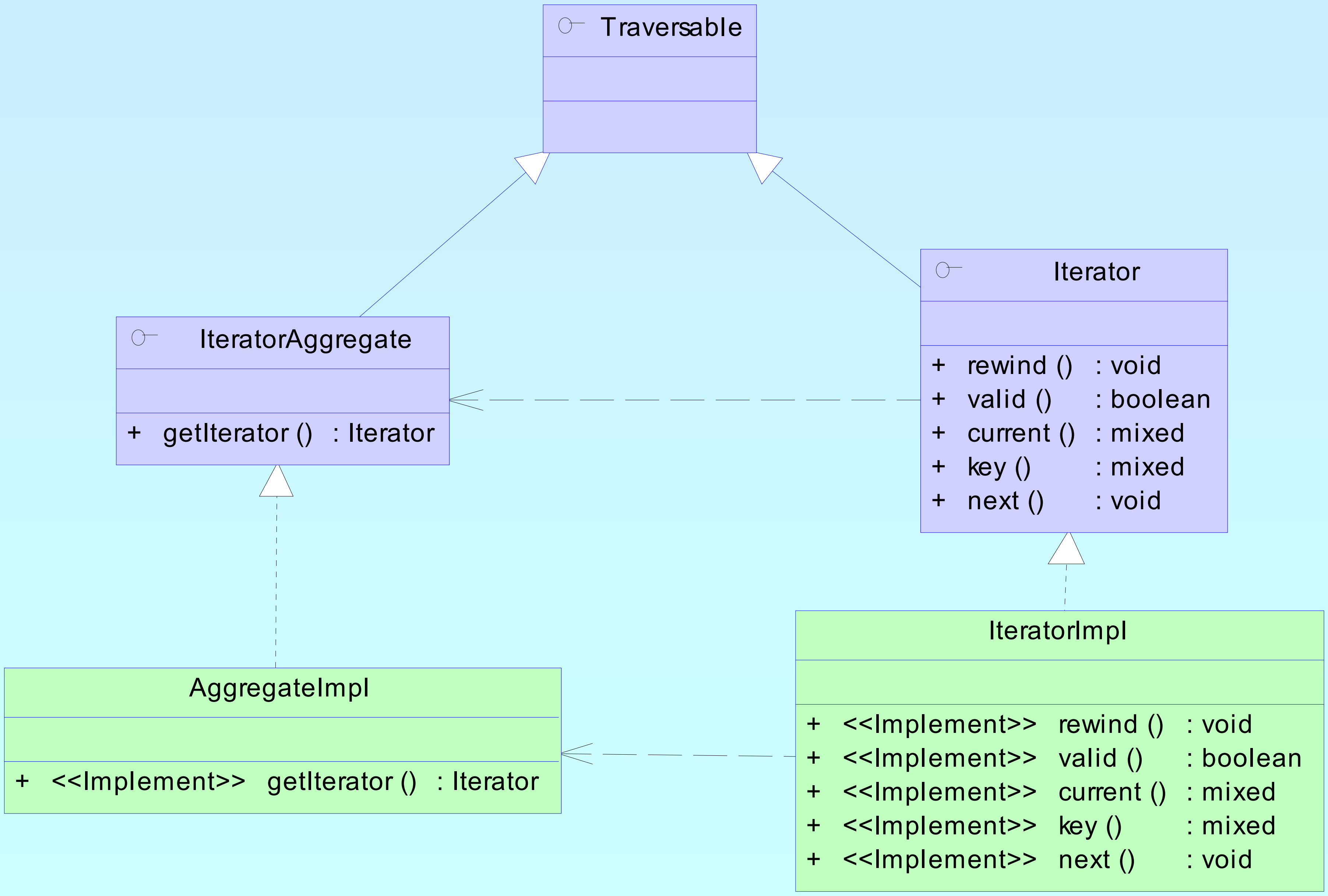
- ☑ Iterators can be internal or external also referred to as active or passive
- ☑ An internal iterator modifies the object itself
- ☑ An external iterator points to another object without modifying it
- ☑ PHP always uses external iterators at engine-level

PHP Iterators

- ☑ Anything that can be iterated implements **Traversable**
- ☑ User classes cannot implement **Traversable**
- ☑ **Aggregate** is used for objects that use external iterators
- ☑ **Iterator** is used for internal traversal or external iterators



Implementing Iterators


Traversable
IteratorAggregate
+ getIterator () : Iterator
Iterator
+ rewind () : void
+ valid () : boolean
+ current () : mixed
+ key () : mixed
+ next () : void
AggregateImpl
+ <<Implement>> getIterator () : Iterator
IteratorImpl
+ <<Implement>> rewind () : void
+ <<Implement>> valid () : boolean
+ <<Implement>> current () : mixed
+ <<Implement>> key () : mixed
+ <<Implement>> next () : void

How Iterators work

- ☑ Iterators can be used manually
- ☑ Iterators can be used implicitly with **foreach**

```
<?php
$o = new ArrayIterator(array(1, 2, 3));
$o->rewind();
while ($o->valid()) {
    $key = $o->key();
    $val = $o->current();
    // some code
    $o->next();
}
?>
```

```
<?php
$o = new ArrayIterator(array(1, 2, 3));
foreach($o as $key => $val) {
    // some code
}
?>
```

Debug Session

```php
<?php
class ArrayIterator {
    protected $ar;
    function __construct(Array $ar) {
        $this->ar = $ar;
    }
    function rewind() {
        rewind($this->ar);
    }
    fucntion valid() {
        return !is_null(key($this->ar));
    }
    function key() {
        return key($this->ar);
    }
    fucntion current() {
        return current($this->ar);
    }
    function next() {
        next($this->ar);
    }
}
?>
```

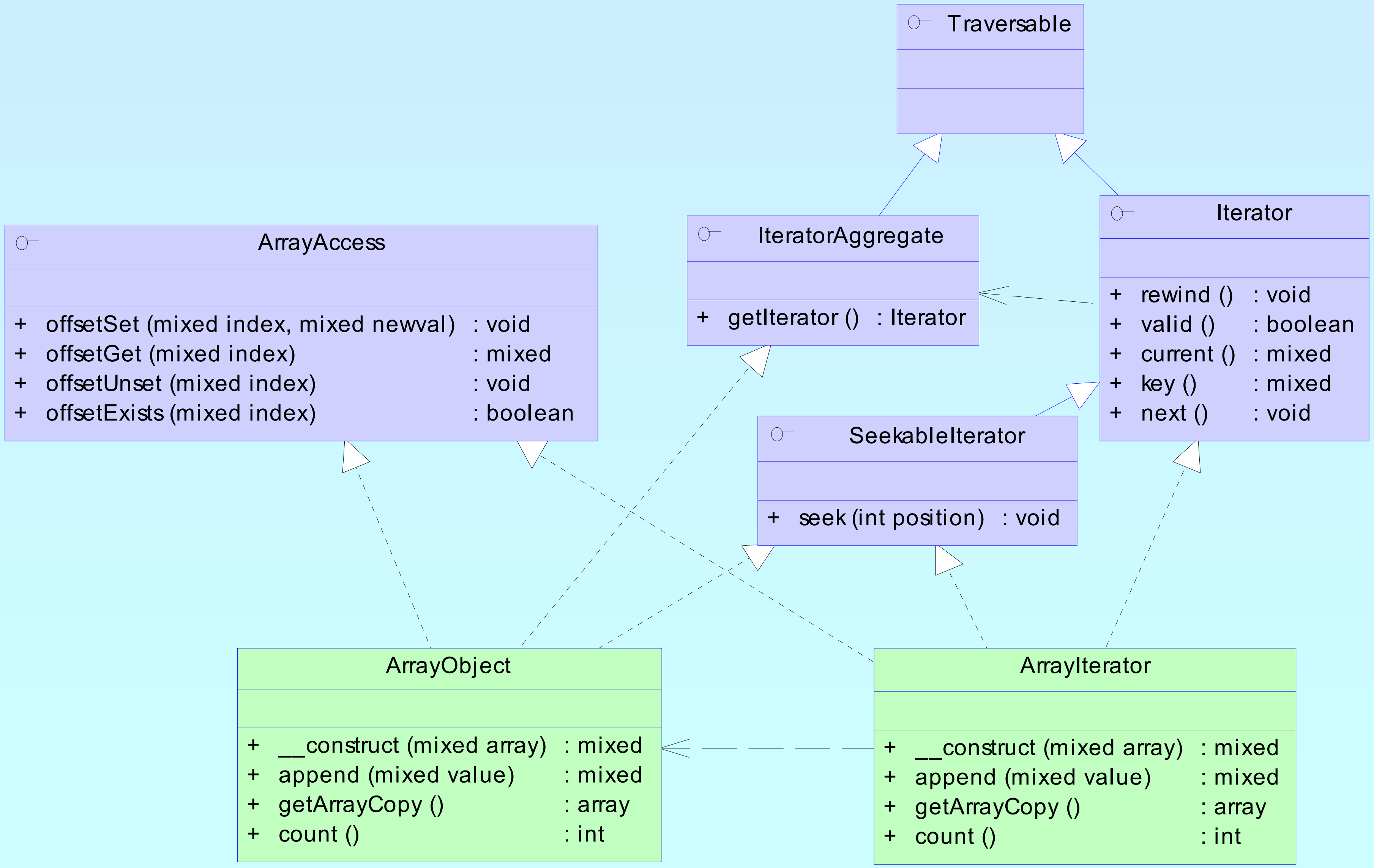
```php
<?php
$a = array(1, 2, 3);
$o = new ArrayIterator($a);
foreach($o as $key => $val) {
    echo "$key => $va\n";
}
?>
```

```
0 => 1
1 => 2
2 => 3
```

Array and property traversal

- ☑ **ArrayObject** allows external traversal of arrays and object properties
- ☑ **ArrayObject** creates **ArrayIterator** instances for iteration
- ☑ Multiple **ArrayIterator** instances can reference the same target with different states

Array and property traversal


Traversable
IteratorAggregate
+ getIterator () : Iterator
Iterator
+ rewind () : void
+ valid () : boolean
+ current () : mixed
+ key () : mixed
+ next () : void
ArrayAccess
+ offsetSet (mixed index, mixed newval) : void
+ offsetGet (mixed index) : mixed
+ offsetUnset (mixed index) : void
+ offsetExists (mixed index) : boolean
SeekableIterator
+ seek (int position) : void
ArrayObject
+ __construct (mixed array) : mixed
+ append (mixed value) : mixed
+ getArrayCopy () : array
+ count () : int
ArrayIterator
+ __construct (mixed array) : mixed
+ append (mixed value) : mixed
+ getArrayCopy () : array
+ count () : int

Algorithms in Iterators

- ☑ Recursive traversal
  - ☑ Arrays
  - ☑ XML data
  - ☑ Directories
- ☑ Filtering values
  - ☑ Numerical calculations
  - ☑ String comparisons
- ☑ Limiting/Extending input iterators
  - ☑ Preventing rewind calls
  - ☑ Concatenation
  - ☑ Repetition...Infinity
  - ☑ Vacuity

References

- ☑ Documentation and Sources to PHP5
  http://php.net
- ☑ Documentation to ext/spl
  http://cvs.php.net/co.php/php-src/ext/spl/spl.php?r=HEAD
  http://somabo.de/php/ext/spl/html/
- ☑ Sourcecode for examples
  ext/spl/examples
- ☑ These slides
  http://somabo.de/talks/