Apache NetBeans 10.0 New and Noteworthy

This page lists the new features available in Apache NetBeans 10.0. There is a release schedule, based on the testing that will be done as part of the NetCAT program.

- JDK 11 support
- Miscellaneous Features
- PHP

JDK 11 support

NetBeans 10.0 will include support for JDK 11, including:

- update of nbjavac module
- removal of Java EE and Corba modules from the JDK
- deprecation of Nashorn Javascript engine
- var support for implicitly typed lambda expressions

Primary tasks for JDK 11 uptake in Apache NetBeans are:

<table>
<thead>
<tr>
<th>Task</th>
<th>Tracking Bug</th>
<th>Owner</th>
<th>Comments</th>
</tr>
</thead>
</table>
- Run tests for modules java.completion, java.editor, java.editor.base, java.hints, java.source, java.source.base, lib.nbjavac  
- Upload updated nb-javac jars  
Pull Request #812  
- Uploaded nb-javac plugin 1.5 in Update Center 10 |
| 3 Java EE and CORBA modules removal: http://openjdk.java.net/jeps/320 | NETBEANS-805 Update any dependencies on Java EE and CORBA modules | Arunava | Pending  
- No dependencies found |
| 4 Dynamic Class-File Constants http://openjdk.java.net/jeps/309 | NETBEANS-804 Update any dependent code for JDK11 Class-file format change | Arunava | Pending  
- Investigating based on Jan's inputs regarding impacted code areas.  
PR 910 |
| 5 Var support for implicitly typed lambda expressions. http://openjdk.java.net/jeps/323 | NETBEANS-806 Var support for Lambda Parameters | rtaneja1/arusinha/vikasprab hakar | Feature wiki: LVTI Support for Lambda Parameters in NetBeans 10  
List of Enhancements/Bugs below. |
| 6 Deprecation of Nashorn JS Engine http://openjdk.java.net/jeps/335 | NETBEANS-1009 Deprecate the Nashorn Javascript Engine | Svata Dedic | |

Var Support in Lambda expression parameters

Identified below enhancements/bugs to support this feature in Apache NetBeans.

<table>
<thead>
<tr>
<th>Type</th>
<th>Apache Jira Id</th>
<th>Owner</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Enhancement</td>
<td>NETBEANS-860: Proposed new hint “Convert Lambda to Use 'var' Parameter Types” for Lambda expression</td>
<td>Done</td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2</td>
<td>Bug</td>
<td>NETBEANS-859: Hint “Use explicit parameter type” is not working with var type lambda parameters</td>
<td>Verified fixed w JDK-8204674</td>
</tr>
<tr>
<td>3</td>
<td>Bug</td>
<td>NETBEANS-862: Autocomplete for var keyword not supported in Lambda parameter</td>
<td>Done</td>
</tr>
<tr>
<td>4</td>
<td>Bug</td>
<td>NETBEANS-863: Refactor rename of var type Lambda parameter is not proper</td>
<td>Verified fixed w JDK-8204674</td>
</tr>
<tr>
<td>5</td>
<td>Bug</td>
<td>NETBEANS-1021: use explicit params type throws exception for invalid lambda params types</td>
<td>Done</td>
</tr>
</tbody>
</table>

## Miscellaneous Features

As the code donations from Oracle keep coming in, they are being integrated into NetBeans. NetBeans 10.0 will see the return of Javascript and Groovy, as well as the addition of JUnit 5.

- **JavaScript**: [https://github.com/apache/incubator-netbeans/pull/882](https://github.com/apache/incubator-netbeans/pull/882)
- **Groovy**: [https://github.com/apache/incubator-netbeans/pull/904](https://github.com/apache/incubator-netbeans/pull/904)
- **OpenJDK**
  - JUnit 5: [https://github.com/apache/incubator-netbeans/pull/892](https://github.com/apache/incubator-netbeans/pull/892)
    - New JUnit 5 Library
    - Default version of JUnit when creating Java 8+ applications
    - Support for Run / Debug Focused Test Methods
    - New JUnit 5 Test Template
    - Updated IDE Hints for JUnit 5's Assertion class

### JavaScript: [https://github.com/apache/incubator-netbeans/pull/882](https://github.com/apache/incubator-netbeans/pull/882)

Modules in the "webcommon" cluster are included in Apache NetBeans for the first time. I.e., all JavaScript features in Apache NetBeans GitHub are part of Apache NetBeans 10.

### Groovy: [https://github.com/apache/incubator-netbeans/pull/904](https://github.com/apache/incubator-netbeans/pull/904)

Modules in the "groovy" cluster are included in Apache NetBeans for the first time. I.e., all Groovy features in Apache NetBeans GitHub are part of Apache NetBeans 10.

**Plus:**
- Add simple infrastructure to write Groovy Hints: [https://github.com/apache/incubator-netbeans/pull/902](https://github.com/apache/incubator-netbeans/pull/902)
- Add trait support for Groovy: [https://github.com/apache/incubator-netbeans/pull/901](https://github.com/apache/incubator-netbeans/pull/901)
- First stab at flow typing: [https://github.com/apache/incubator-netbeans/pull/903](https://github.com/apache/incubator-netbeans/pull/903)

### OpenJDK

- Automatically detect JTReg from OpenJDK configuration: [https://github.com/apache/incubator-netbeans/pull/925](https://github.com/apache/incubator-netbeans/pull/925)
- Register the expanded JDK as a Java Platform: [https://github.com/apache/incubator-netbeans/pull/918](https://github.com/apache/incubator-netbeans/pull/918)
- Various improvements to make the OpenJDK project work better: [https://github.com/apache/incubator-netbeans/pull/897](https://github.com/apache/incubator-netbeans/pull/897)

### JUnit 5: [https://github.com/apache/incubator-netbeans/pull/892](https://github.com/apache/incubator-netbeans/pull/892)

With NetBeans existing JUnit and Maven integration, the introduction of JUnit 5 didn't need much work, as in a lot of cases, NetBeans defers to Maven, and in particular the Maven Surefire plugin to execute tests. No work has been implemented for Ant specific JUnit 5 integration.

### New JUnit 5 Library

Introducing support for JUnit 5.3.1 a new Library has been added to NetBeans.
Default version of JUnit when creating Java 8+ applications

When working with a Maven based Java 8+ project with no existing tests, or test framework defined in the project pom file, the IDE will use JUnit 5 as the default JUnit version, and add in the appropriate dependencies.

Support for Run / Debug Focused Test Methods

JUnit 5 introduced a new @Testable annotation, any method that includes an annotation that inherits from this, will be allowed to be executed with these action menu items.
New JUnit 5 Test Template

Default JUnit 5 Test Template provided out of the box. Note JUnit 5 doesn't currently support test suite's for its Jupiter engine, so the template provided will attempt to create a test suite using the Vintage engine.

- Unit Tests
  - JUnit Test
  - Test Class – JUnit 3.x
  - JUnit Test for Existing Class
  - JUnit Test Suite
  - JUnit Test Suite – JUnit 3.x
  - Test Class – JUnit 4.x
  - Test Class – JUnit 5.x

Updated IDE Hints for JUnit 5's Assertion class

Introduced updated assertArrayEquals hints to support JUnit 5 as the method signatures have changed since JUnit 4.

PHP

A big addition to NetBeans 10 is that it will once again support PHP. Features will include support for PHP 7.0 through 7.3, PHPStan and Twig, as well as editing and debugging enhancements.

All the PHP support for NetBeans 9 was contributed by our NetBeans user Junichi Yamamoto.

PHP 7.3 Support

https://lists.apache.org/thread.html/3f7fee458f5b47d5531ad42350c6e7f619636f0708b6bcb33db2f49a@%3Cdev.netbeans.apache.org%3E

- Allow a trailing comma in function calls
```php
<?php

// Function Calls
$array1 = ['baz'];
$array2 = ['qux'];
$merged = array_merge(
    $array1,
    $array2,
    ['foo', 'bar'],
);

// Method & Closure Calls
$foo = new Foo(
    'constructor',
    'foo',
);
$foo->bar(
    'method',
    'bar',
);
$bar(
    'closure',
    'bar',
);

// Language Constructs
unset(
    $param1,
    $param2,
);

* list() Referent Assignment

<?php

$array = [1, 2];
list($a, &$b) = $array;

echo $a . PHP_EOL;
echo $b . PHP_EOL;

* Flexible Heredoc and Nowdoc Syntaxes
```
<?php

// Closing Marker Indentation
indentation = <<< 'END'
    closing
        marker
    indentation
END;

// Closing Marker New Line
$array = <<< END
    closing
    marker
    new
    line
    $test
    END, $test];

PHP 7.2 Support

* Trailing Commas In List Syntax

<?php

namespace A;

class Foo {
}

class Bar {
}

class Baz {
}

namespace B;

use A\{
    Foo,
    Bar,
    Baz,
};

$foo = new Foo();
$bar = new Bar();
$baz = new Baz();
• Coloring for object type

```php
10  class ObjectTypeHint {
11  }
12
13  // This function can be statically analysed to conform to the
14  // return type
15  function correctFunction(): object {
16      $obj = json_decode('{}');
17  
18      return $obj;
19  }
20
21  // This function can be statically analysed to contain an error
22  // and will also fail at runtime.
23  function errorFunction(): object {
24      return [];
25  }
26
27  function parameterType(object $object) {
28  }
29
30
• PHP version in Project Properties dialog
PHP 7.1 Support

- Class constant visibility
<?php

class ParentClass {
    const IMPLICIT_PUBLIC_PARENT_CONST = 0;
    public const PUBLIC_PARENT_CONST = 0;
    private const PRIVATE_PARENT_CONST = 0;
    protected const PROTECTED_PARENT_CONST = 0;

    public function test() {
        ParentClass::PRIVATE_PARENT_CONST;
        self::PROTECTED_PARENT_CONST;
    }
}

interface TestInterface {
    public const PUBLIC_INTERFACE_CONST = 0;
    const IMPLICIT_INTERFACE_PUBLIC = 1;
}

class ChildClass {
    const IMPLICIT_PUBLIC_CHILD_CONST = 0;
    public const PUBLIC_CHILD_CONST = 0;
    private const PRIVATE_CHILD_CONST = 0;
    protected const PROTECTED_CHILD_CONST = 0;

    public function ChildClass::IMPLICIT_PUBLIC_CHILD_CONST;
}

class TestInterface1 {
    const IMPLICIT_PUBLIC_PARENT_CONST = 0;
    public const PUBLIC_PARENT_CONST = 0;
    private const PRIVATE_PARENT_CONST = 0;
    protected const PROTECTED_PARENT_CONST = 0;

    public const @class 
ChildClass Magic Constant

    * Multi catch exception handling
<php

class ExceptionType1 extends Exception {
    public function something1() {
    }
}

class ExceptionType2 extends Exception {
    public function something2() {
    }
}

try {
    // Some code...
    catch (ExceptionType1 | ExceptionType2 $ex) {
        echo $ex->getMessage();
    } catch (Exception $e) {
        echo $e->getMessage();
    }

• Nullable types

<?php

class Foo {
}

class NullableType {

    public function returnType(): ?Foo {
        return new Foo();
    }

    public function parameterType(?Foo $foo, ?string $string) {
    }

• Support for keys in list()
```php
<?php
$data = [
    ['id' => 1, "name" => 'Tom'],
    ['id' => 2, "name" => 'Fred'],
];

// list() style
list("id" => $id1, "name" => $name1) = $data[0];

// [] style
[$id => $id1, "name" => $name1] = $data[0];

// list() style
foreach ($data as list("id" => $id, "name" => $name)) {
    // something...
}

// [] style
foreach ($data as ["id" => $id, "name" => $name]) {
    // something...
}
```

* Coloring for new keywords (void, iterable)

```php
<?php
function iterator(iterable $iter): void {
    foreach ($iter as $val) {
        // something...
    }
}
```

* PHP version in Project Properties dialog

<table>
<thead>
<tr>
<th>Encoding:</th>
<th>UTF-8</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHP Version:</td>
<td>PHP 7.1</td>
</tr>
</tbody>
</table>

- PHP version is used only for hints
- Allow short tags (<?)
- Allow ASP tags (<% and %>)

**PHP 7.0 Support**

* Context sensitive lexer
<?php

class ContextSensitiveLexer
{
    const CONST = [1,3], GOTO = 2;
    const ECHO = "echo";
    const NEW = [self::GOTO, self::ECHO], FOR = 2;

    public function echo() {
        echo self::GOTO . PHP_EOL;
        echo ContextSensitiveLexer::CONST[0] . PHP_EOL;
        echo ContextSensitiveLexer::ECHO . PHP_EOL;
    }

    public static function new() {
        return new ContextSensitiveLexer();
    }
}

$instance = ContextSensitiveLexer::new();

---

PHPStan Support

- Options (Tools > Options > PHP > Code Analysis > PHPStan)

- Inspect (Source > Inspect... > Configuration: PHPStan)
• Hint for void return types
• Hint for incorrect non-abstract methods

```
<?php

function shouldReturnNothing(): void {
    return 1;

    "void" cannot return anything
    ----
    (Alt-Enter shows hints)
```

• Suggestion for the strict types declaration. It is suggested when the caret is on the line which contains "<?php" and the file doesn't have the declaration.

```
<?php

class IncorrectNonAbstractMethodsHint
{
    public function correctNonAbstractMethod() {
    }

    public function incorrectNonAbstractMethod();
    ! Add body of the method: incorrectNonAbstractMethod
```

• Lowercase code completion for TRUE, FALSE, and NULL constants. Enabled by default.

```
Change project default to HTML 3.2
Change project default to HTML 4.0 Strict
Change project default to HTML 4.0 Transitional
Change project default to HTML 4.0 Frameset
Change project default to HTML 4.01 Strict
Change project default to HTML 4.01 Transitional
Change project default to HTML 4.01 Frameset
Change project default to HTML 5
! Add declare(strict_types=1)
! Customize Hints in Options Dialog
```

• Lowercase code completion for TRUE, FALSE, and NULL constants. Enabled by default.
Code folding for arrays

```php
<?php

$isNull = true;

true true PHP Platform
```

- Code folding for arrays
```php
$array1 = [
    'nested1' => ['test',
                  'nested2-1' => 'test',
                  'nested2-2' => 'test',
                  'nested2-3' => ['nested3-1' => 'test',
                                  'nested3-2' => 'test'],
    ],
];
$array2 = array(...10 lines);

$array1 = [
    'nested1' => ['test',
                  'nested2-1' => 'test',
                  'nested2-2' => 'test',
                  'nested2-3' => ['nested3-1' => 'test',
                                  'nested3-2' => 'test'],
    ],
];

$array2 = array(...10 lines);
```

```php
$array1 = [
    'test1' => ['test2' => ['test3' => 'test4']],
];
$array2 = array("test1" => array(
    "test2" => array("test3" => "test4")
));
```

- Code foldings for use statements
```php
namespace A;

use App\Test1;
use App\Test2;

use App\Test3;
use App\{
    Test4,
    Test5
};

function functionName($param) {
}

use App\{
    Test6
};

use App\Test7;

new Test1();
new Test2();
new Test3();
new Test4();
new Test5();
new Test6();
new Test7();

namespace B;

use App\Test2;

function functionName($param) {
}

use const App\Test3\CONSTANT;
use function App\{
    test1,
    test2
};
```

* Code folding for php tags

```php
<?php
    echo "code folding for php tags";
?>

<?php
    echo "code folding for php tags";
?>
```
* New vardoc support (/** @var VarType $varType */)

```php
<?php

class VarType
{
    public $test = "";
    public function test()
    {
    }
}

/** @var VarType */
ivarType = getVerType();
ivarType->test();
```

* Option for comment completion. Enabled by default. (Tools > Options > Editor > Code Completion > Language: PHP)

![Options settings for code completion](image)
Debugger

- Conditional breakpoints

```php
<?php
/*
 * Enable the option
 */

/*
 * Disable the option
 */
```

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
    $array = ["foo", "bar", "baz", "qux"];
    foreach ($array as $value) {
      echo $value;
    }
    ?>
  </body>
</html>
```
Twig

- Autocompletion for delimiters, brackets, and quotes

**Options**

Language: **Twig File**

Quotes and Delimiters Completion:
- Use Smart Quotes (' and ') Completion (after text selection, e.t.c.)
- Use Smart Delimiters ({ and %) Completion

`#{ Autocomplete delimiters, brackets, and quotes #}`

- Support for the palette
{# empty Twig template #}