Namespaces

ใน PHP Namespaces คืออะไร

ใน PHP นั้น Namespaces ถูกออกแบบมาเพื่อแก้ปัญหาสองอย่างที่ผู้เขียนไลบรารีและแอปพลิเคชันพบเจอเมื่อสร้างส่วนประกอบโค้ดที่นำกลับมาใช้ใหม่ได้ เช่น คลาสหรือฟังก์ชัน

1.การชนกันของชื่อระหว่างโค้ดที่ตัวเราสร้างและคลาส/ฟังก์ชัน/ค่าคงที่ภายใน PHP หรือคลาส/ฟังก์ชัน/ค่าคงที่ของบุคคลที่สาม

2.ความสามารถในการใช้นามแฝง ซึ่งออกแบบมาเพื่อแก้ไขปัญหาแรกและทำให้การอ่านโค้ดสะดวกขึ้น

ประโยชน์ของ Namespaces

- ช่วยหลีกเลี่ยงการชนกันของชื่อระหว่างคลาส/ฟังก์ชัน/ค่าคงที่ที่ตัวเรากำหนดเองหรือคลาส/ฟังก์ชัน/ค่าคงที่ที่กำหนดโดยบุคคลที่สาม
- สามารถใช้นามแฝงหรือย่อชื่อให้สั้นลง เพื่อทำให้การอ่านโค้ดสะดวกขึ้น
- จัดกลุ่มคลาส, อินเตอร์เฟส, ฟังก์ชัน และค่าคงที่ที่เกี่ยวข้องกัน ชื่อของ Namespace ไม่คำนึงถึงตัวพิมพ์ใหญ่-พิมพ์เล็ก

การประกาศ namespace

Namespaces จะถูกประกาศไว้ที่ตอนต้นของไฟล์ด้วย keyword ว่า namespace

ตัวอย่าง: ประกาศ namespace ที่ชื่อว่า Html

```php
<?php
namespace Html;
?>
```

**การประกาศ namespace ใน PHP จะต้องเป็นสิ่งแรกในไฟล์ PHP นั้น ๆ หากมีโค้ดหรือคำสั่งอื่น ๆ มาก่อนหน้านั้น จะทำให้โค้ดนั้นไม่ถูกต้อง

ตัวอย่าง:

```php
<?php
echo "Hello World!";
namespace Html;
...
?>
```

การประกาศ Namespaces

นอกจากนี้ เรายังสามารถซ้อน (nested) namespace ได้ โดยการประกาศ namespace หนึ่งอยู่ภายใน namespace อื่น

ตัวอย่าง: ประกาศ namespace ที่ชื่อ Html ภายใน namespace ที่ชื่อ Code

```php
<?php
namespace Code\Html;
?>
```

การใช้ Namespaces

โค้ดใดก็ตามที่ตามหลังการประกาศ namespace จะทำงานภายใน namespace นั้น ดังนั้นคลาสที่อยู่ภายใน namespace นั้นสามารถถูกสร้างอินสแตนซ์ (instantiated) ได้โดยไม่ต้องมีตัวระบุเพิ่มเติม (qualifiers) แต่ถ้าต้องการเข้าถึงคลาสจากภายนอก namespace จะต้องใส่ชื่อ namespace นำหน้าคลาสนั้นด้วย

ตัวอย่าง: การใช้ classes จาก namespace ที่ชื่อ Html

```php
<?php
$table = new Html\Table();
$row = new Html\Row();
?>
```

เมื่อมีการใช้งานคลาสหลาย ๆ คลาสจาก namespace เดียวกันพร้อมกัน การใช้คำสั่ง namespace จะทำให้โค้ดดูเรียบง่ายขึ้น เพราะไม่จำเป็นต้องระบุชื่อ namespace นำหน้าคลาสที่เราจะใช้งาน

ตัวอย่าง: การใช้คลาสจาก namespace ที่ชื่อ Html โดยไม่ต้องใช้ตัวระบุ Html\

```php
<?php
namespace Html;
$table = new Table();
$row = new Row();
?>
```

การตั้งนามแฝงให้กับ Namespace

โดยใช้คำสั่ง use เพื่อกำหนด alias ให้กับ namespace นั้น ทำให้การเขียนโค้ดสะดวกยิ่งขึ้น

ตัวอย่าง: การตั้งนามแฝงให้กับ namespace

```php
<?php
use Html as H;
$table = new H\Table();
?>
```

ตัวอย่าง: การตั้งนามแฝงให้กับคลาส

```php
<?php
use Html\Table as T;
$table = new T();
?>
```

เปรียบเทียบ namespace ใน PHP กับภาษา Java, C, Python

PHP

```php
<?php
namespace MyApp\Utilities;

class Logger {
    public function log($message) {
        echo $message;
    }
}

// การใช้งาน
$logger = new \MyApp\Utilities\Logger();
$logger->log('Hello, Namespace!');
?>
```

ใน PHP การใช้ namespace ช่วยให้สามารถจัดกลุ่มของฟังก์ชัน, คลาส หรือคงค่าต่างๆ เพื่อป้องกันการชนกันของชื่อ

- การประกาศ: ใช้คำสั่ง namespace เพื่อสร้าง namespace
- การอ้างอิง: อ้างอิงคลาสหรือฟังก์ชันผ่านการใช้ชื่อเต็ม หรือใช้ use เพื่อลดการพิมพ์ชื่อเต็ม

Java

```java
package com.myapp.utilities;

public class Logger {
    public void log(String message) {
        System.out.println(message);
    }
}

// การใช้งาน
import com.myapp.utilities.Logger;

public class Main {
    public static void main(String[] args) {
        Logger logger = new Logger();
        logger.log("Hello, Namespace!");
    }
}
```

ใน Java การใช้ namespace ทำผ่านการจัดการ package ซึ่งเป็นวิธีจัดโครงสร้างของคลาส เพื่อป้องกันการชนกันของชื่อ

- การประกาศ: ใช้ package เพื่อกำหนด namespace ให้กับคลาส
- การอ้างอิง: ใช้ import เพื่ออ้างอิงคลาสจากแพ็คเกจอื่น

C

```c
#include <stdio.h>

struct MyNamespace_Logger {
    void (*log)(const char* message);
};

void MyNamespace_log(const char* message) {
    printf("%s\n", message);
}

int main() {
    struct MyNamespace_Logger logger = { MyNamespace_log };
    logger.log("Hello, Namespace!");
    return 0;
}
```

ใน C ไม่มี namespace โดยตรง แต่สามารถใช้ struct, enum, และ prefix เพื่อเลี่ยงการชนกันของชื่อได้

- การเลี่ยงการชนกันของชื่อ: ใช้ prefix เช่น MyNamespace_ เพื่อแยกแยะฟังก์ชันหรือ struct

Python

```python
# utilities/logger.py

class Logger:
    def log(self, message):
        print(message)

# main.py
from utilities.logger import Logger

logger = Logger()
logger.log("Hello, Namespace!")
```

ใน Python ใช้ module และ package เพื่อจัดการ namespace โดยธรรมชาติ

- การประกาศ: ใช้ไฟล์ Python เป็น module โดยอัตโนมัติถือเป็น namespace
- การอ้างอิง: ใช้ import เพื่อนำคลาสหรือฟังก์ชันจาก module อื่นมาใช้งาน

Thank you