Pulugin là 1 class dùng thể thay đổi 1 phương thức của các lớp public bằng cách chặn cuộc gọi hàm và mã chạy trước đó,sau hoặc xung quanh hàm đó, -> overide hoặc extends hành vi của các phương thức gốc, công khai cho bất kỳ lớp hoặc giao diện nào .

* Không thể sử dụng plugin trong trường hợp
  + Phương thức cuối
  + Class cuối
  + Phương thức không phải dạng public
  + Các phương thức tĩnh
  + \_\_contruct
  + Các đối tượng được khởi tạo trước đó Magento\Framework\Interception được khởi động

Cách tiếp cận đánh chặn này làm giảm xung đột giữa các phần mở rộng làm thay đổi hành vi của cùng một lớp hoặc phương thức. Việc Plugin thực hiện lớp của bạn thay đổi hành vi của một hàm lớp, nhưng nó không thay đổi chính lớp đó. Magento gọi các bộ chặn này theo tuần tự theo thứ tự sắp xếp được cấu hình, do đó chúng không xung đột với nhau.

**Khai báo một plugin**

di.xml

<config>

<type name="{ObservedType}">

<plugin name="{pluginName}" type="{PluginClassName}" sortOrder="1" />

</type>

</config>

#### **Required option**

* type name. Một lớp hoặc giao diện mà plugin quan sát.
* plugin name. Tên plugin tùy ý xác định plugin. Cũng được sử dụng để hợp nhất các cấu hình cho plugin.
* plugin type. Tên của class của plugin hoặc loại ảo của nó. Sử dụng quy ước đặt tên sau khi bạn xác định yếu tố này: \Vendor\Module\Plugin\<ClassName>.

#### **Optional options**

* plugin sortOrder. Các plugin gọi cùng một phương thức sẽ chạy chúng bằng cách sử dụng thứ tự này.
* plugin disabled. Để tắt plugin, hãy đặt thành phần này thành true. Giá trị mặc định là false.

### Xác định plugin

3 phương pháp trong Plugin

* before - - beforeDispatch()

**Trước khi phương thức** là **phương pháp** đầu tiên để chạy trong một phương pháp quan sát được, và các phương thức này phải có cùng tên với tên của quan sát được trong khi nhãn tiền tố là before.

Để áp dụng các phương thức trước để sửa đổi các đối số của một phương thức được quan sát, bạn có thể trả về một đối số đã sửa đổi. Nếu có nhiều đối số, sự trở lại sẽ được thực hiện theo một loạt các đối số đó. Nếu trả về không hợp lệ, điều đó có nghĩa là các đối số cho phương thức quan sát không được sửa đổi.

<?php

namespace Vendor\Modeule\Plugin;

class ExamplePlugin

{

public function beforeSetTitle(\Mageplaza\HelloWorld\Controller\Index\Example $subject, $title)

{

$title = $title . " to ";

echo \_\_METHOD\_\_ . "</br>";

return [$title];

}

}

* around - - aroundDispatch()

**Các phương thức xung quanh** cho phép mã chạy trước và sau phương thức quan sát được, vì vậy bạn có thể ghi đè lên một phương thức. Các phương thức này phải có cùng tên với tên của người được quan sát trong khi nhãn tiền tố là “around”.

Trước khi sắp xếp đối số của phương thức gốc, một callable phương thức xung quanh sẽ được gọi đến phương thức tiếp theo trong chuỗi, điều đó có nghĩa là plugin tiếp theo hoặc hàm quan sát được gọi.

**Lưu ý:** Trong trường hợp callable không được khai báo, việc gọi đến cả plugin tiếp theo lẫn phương thức gốc đều không đạt được.

<?php

namespace Mageplaza\HelloWorld\Plugin;

class ExamplePlugin

{

public function aroundGetTitle(\Mageplaza\HelloWorld\Controller\Index\Example $subject, callable $proceed)

{

echo \_\_METHOD\_\_ . " - Before proceed() </br>";

$result = $proceed();

echo \_\_METHOD\_\_ . " - After proceed() </br>";

return $result;

}

}

* after - - afterDispatch()

**Sau khi các phương thức** bắt đầu chạy ngay sau khi phương thức quan sát được hoàn thành, và các phương thức này phải có cùng tên với tên của người quan sát được trong khi nhãn tiền tố là “after”.

Sau khi các phương thức chịu trách nhiệm chỉnh sửa kết quả của một phương pháp được quan sát một cách chính xác và được yêu cầu có giá trị trả lại.

Bạn có thể sử dụng các phương pháp này để thay đổi kết quả của một phương pháp được quan sát bằng cách sửa đổi kết quả ban đầu và trả về nó ở cuối phương thức.

<?php

namespace Mageplaza\HelloWorld\Plugin;

class ExamplePlugin

{

public function afterGetTitle(\Mageplaza\HelloWorld\Controller\Index\Example $subject, $result)

{

echo \_\_METHOD\_\_ . "</br>";

return '<h1>'. $result . '[Mageplaza.com](http://mageplaza.com/)' .'</h1>';

}

}

Nếu plugin của bạn được gọi bằng một phương thức khớp với các đối số, nó cũng phải khớp với chúng và đồng thời, bạn cần phải tuân theo chúng một cách cẩn thận. Trong quá trình này, hãy chú ý đến chữ ký gốc của phương thức cũng như các tham số mặc định và loại đề xuất.

Ví dụ: áp dụng mã bên dưới để xác định tham số loại SomeType là nullable:

<?php

namespace Mageplaza\HelloWorld\Model;

class MyUtility

{

public function save(SomeType $obj = null)

{

//do something

}

}

Nếu bạn gói phương thức này bằng một plugin như dưới đây:

<?php

namespace Mageplaza\HelloWorld\Plugin;

class MyUtilityPlugin

{

public function aroundSave(\Mageplaza\HelloWorld\Model\MyUtility $subject, \callable $proceed, SomeType $obj)

{

//do something

}

}

**Lưu ý** : Thiếu = Null

Nếu bạn gọi phương thức cùng với null, PHP sẽ [gây ra lỗi nghiêm trọng](https://www.mageplaza.com/kb/exception-printing-is-disabled-by-default-for-security-reasons-magento-1-2.html) vì plugin của bạn không được cho phép trong plugin của bạn. Bên cạnh đó, điều quan trọng là phải làm theo các đối số từ plugin được gọi bởi phương thức. Nhưng nếu không quan tâm đến các đối số, vui lòng sử dụng variadics và đối số giải nén để hoàn tất việc này:

<?php

namespace Mageplaza\HelloWorld\Plugin;

class MyUtilityPlugin

{

public function aroundSave(\Mageplaza\HelloWorld\Model\MyUtility $subject, \callable $proceed, ...$args)

{

//do something

$proceed(...$args);

}

}

## **Đặt ưu tiên cho plugin**

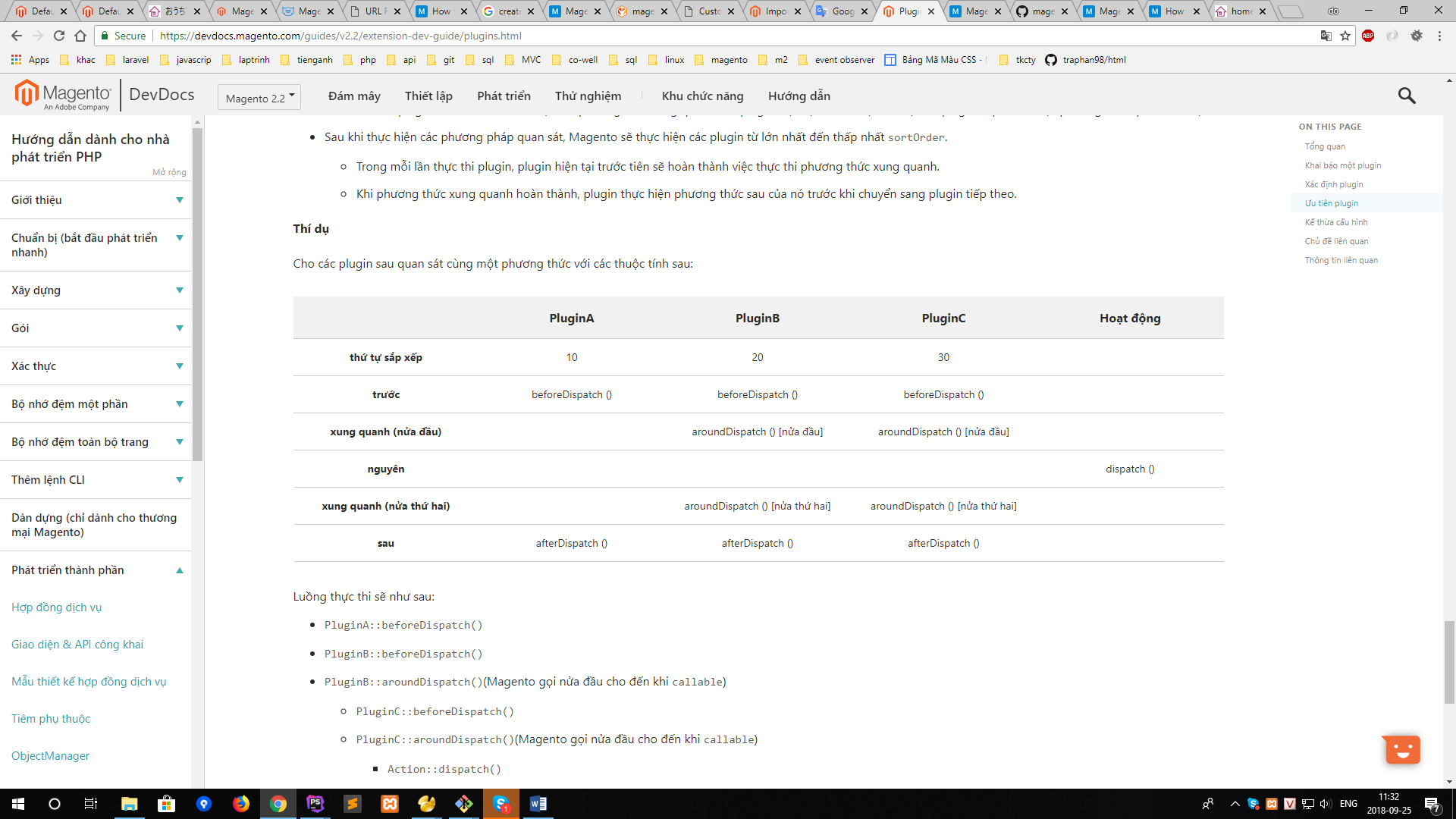
Các sortOrder bất động sản cho các plugins xác định khi nào họ trước, sau hoặc xung quanh phương pháp được gọi khi nhiều hơn một plugin đang quan sát cùng một phương pháp.

Các quy tắc ưu tiên cho việc đặt hàng các plugin:

* Trước khi thực hiện phương pháp quan sát, Magento sẽ thực hiện các plugin từ thấp nhất đến lớn nhất sortOrder.
  + Trong mỗi lần thực thi plugin, Magento thực thi phương thức trước của plugin.
  + Sau khi plugin trước hoàn thành thực thi, phương thức xung quanh của plugin hiện tại sẽ bao bọc và thực thi plugin tiếp theo hoặc phương thức quan sát được.
* Sau khi thực hiện các phương pháp quan sát, Magento sẽ thực hiện các plugin từ lớn nhất đến thấp nhất sortOrder.
  + Trong mỗi lần thực thi plugin, plugin hiện tại trước tiên sẽ hoàn thành việc thực thi phương thức xung quanh.
  + Khi phương thức xung quanh hoàn thành, plugin thực hiện phương thức sau của nó trước khi chuyển sang plugin tiếp theo.

**Thí dụ**

Cho các plugin sau quan sát cùng một phương thức với các thuộc tính sau:



Luồng thực thi sẽ như sau:

* PluginA::beforeDispatch()
* PluginB::beforeDispatch()
* PluginB::aroundDispatch()(Magento gọi nửa đầu cho đến khi callable)
  + PluginC::beforeDispatch()
  + PluginC::aroundDispatch()(Magento gọi nửa đầu cho đến khi callable)
    - Action::dispatch()
  + PluginC::aroundDispatch()(Magento gọi nửa sau callable)
  + PluginC::afterDispatch()
* PluginB::aroundDispatch()(Magento gọi nửa sau callable)
* PluginB::afterDispatch()
* PluginA::afterDispatch()

### Kế thừa cấu hình

Các lớp và giao diện được triển khai hoặc kế thừa từ các lớp có trình cắm thêm cũng sẽ kế thừa các plugin từ lớp cha.

Magento sử dụng các plugin được định nghĩa trong phạm vi toàn cục khi hệ thống nằm trong một khu vực cụ thể (ví dụ: giao diện người dùng, phụ trợ, v.v.). Bạn cũng có thể mở rộng hoặc ghi đè lên cấu hình plugin toàn cầu này thông qua khu vực di.xml.