-Mỗi module có thể có 1 file global và 1 file area-specific di.xml. Ứng dụng đọc tất cả các file cấu hình di.xml được khai báo trong hệ thống và hợp nhất chúng lại bằng cách append tất cả các node

-Theo luật thông thường, các area-specific di.xml files nên cấu hình dependencies cho presentation layer và file global di.xml của module nên cấu hình các dependencies còn lại

-Ứng dụng tải cấu hình theo các giai đoạn sau:

1.initial (app/etc/di.xml)

2.Global (<moduleDir>/etc/di.xml)

3.Area-specific (<moduleDir>/etc/<area>/di.xml>)

\*các areas là adminhtml, frontend, graphql, webapi\_rest, webapi\_soap, crontab

-Type configurations miêu tả lifestyle của object và cách khởi tạo nó

VD:

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/config.xsd">

<virtualType name="moduleConfig" type="Magento\Core\Model\Config">

<arguments>

<argument name="type" xsi:type="string">system</argument>

</arguments>

</virtualType>

<type name="Magento\Core\Model\App">

<arguments>

<argument name="config" xsi:type="object">moduleConfig</argument>

</arguments>

</type>

</config>

\*Trong đó: moduleConfig là 1 virtual type kế thừa lại type Magento\Core\Model\Config và tất cả các instances của type Magento\Core\Model\App sẽ nhận 1 instance của moduleConfig như 1 dependency

-Virtual Type: cho phép thay đổi arguments của 1 injectable dependency cụ thể và thay đổi hành vi của 1 class cụ thể. Điều này cho phép bạn sử dụng 1 class tùy chỉnh mà không ảnh hưởng tới các classes khác có 1 phụ thuộc ở class gốc. Ví dụ trên tạo 1 virtual type cho Magento\Core\Model\Config và sử dụng system như là 1 constructor argument cho type

-Constructor argument: bạn có thể cấu hình constructor argument của class trong argument node trong file di.xml. Object manager sẽ inject những arguments này vòa trong class khi khởi tạo. Tên của argument được cấu hình trong file XML phải tương ứng với tên của các tham số trong constructor của class được cấu hình

Ví dụ sau tạo instance của Magento\Core\Model\Session với constructor argument $sessionName được set giá trị là adminhtml

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/config.xsd">

<type name="Magento\Core\Model\Session">

<arguments>

<argument name="sessionName" xsi:type="string">adminhtml</argument>

</arguments>

</type>

</config>

1 số ví dụ khác:

<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="urn:magento:framework:ObjectManager/etc/config.xsd">

<type name="Magento\Example\Type">

<arguments>

<!-- Pass simple string -->

<argument name="stringParam" xsi:type="string">someStringValue</argument>

<!-- Pass instance of Magento\Some\Type -->

<argument name="instanceParam" xsi:type="object">Magento\Some\Type</argument>

<!-- Pass true -->

<argument name="boolParam" xsi:type="boolean">1</argument>

<!-- Pass 1 -->

<argument name="intParam" xsi:type="number">1</argument>

<!-- Pass application init argument, named by constant value -->

<argument name="globalInitParam" xsi:type="init\_parameter">Magento\Some\Class::SOME\_CONSTANT</argument>

<!-- Pass constant value -->

<argument name="constantParam" xsi:type="const">Magento\Some\Class::SOME\_CONSTANT</argument>

<!-- Pass null value -->

<argument name="optionalParam" xsi:type="null"/>

<!-- Pass array -->

<argument name="arrayParam" xsi:type="array">

<!-- First element is value of constant -->

<item name="firstElem" xsi:type="const">Magento\Some\Class::SOME\_CONSTANT</item>

<!-- Second element is null -->

<item name="secondElem" xsi:type="null"/>

<!-- Third element is a subarray -->

<item name="thirdElem" xsi:type="array">

<!-- Subarray contains scalar value -->

<item name="scalarValue" xsi:type="string">ScalarValue</item>

<!-- and application init argument -->

<item name="globalArgument " xsi:type="init\_parameter">Magento\Some\Class::SOME\_CONSTANT</item>

</item>

</argument>

</arguments>

</type>

</config>

-Ánh xạ abstraction-implementaion: khi 1 class yêu cầu interface trong hàm \_\_construct(), Magento cần biết phải dùng class nào để tạo đối tượng tương ứng với interface đó. Magento Object Manager sẽ tra các cấu hình <preference> để biết nên dùng class nào làm “implementation” cho interface đó

Ví dụ:

<!-- File: app/etc/di.xml -->

<config>

<preference for="Magento\Framework\UrlInterface" type="Magento\Framework\Url" />

</config>

\*Ý nghĩa: bất cứ class nào cần Magento\Framework\UrlInterface thì magento sẽ inject Magento\Framework\Url vào đó, vì file này ở app/etc/di.xml nên áp dụng toàn hệ thống (global)

-Ghi đè 1 phương thức bằng ‘preference’ nodes: nếu muốn ghi đè 1 phương thức public hay protected từ 1 core class, sử dụng preperence node trong di.xml

**1. Virtual Type là gì?**

* Là cách tạo một **"phiên bản ảo"** của 1 class gốc.
* Giúp bạn **tùy chỉnh các tham số constructor** mà không ảnh hưởng đến class gốc.
* **Không cần tạo file PHP mới** (nhưng thường người ta sẽ có class cụ thể luôn cho dễ hiểu).

<virtualType name="MyCustomGreeting" type="Vendor\HelloWorld\Model\Greeting">

<arguments>

<argument name="greetingPrefix" xsi:type="string">Xin chào</argument>

</arguments>

</virtualType>

**2. Override method bằng <preference>**

* Dùng để **thay thế hoàn toàn 1 class bằng 1 class khác**.
* Cho phép bạn **ghi đè (override)** method có sẵn trong class gốc.

<preference for="Magento\Checkout\Block\Onepage\Success"

type="Vendor\Override\Block\Onepage\Success" />

→ Khi đó, bất kỳ chỗ nào Magento cần Magento\Checkout\Block\Onepage\Success thì sẽ thay bằng class bạn viết.

| **Mục đích** | **Virtual Type** | **Preference** |
| --- | --- | --- |
| Tùy chỉnh constructor | ✅ Có | ❌ Không |
| Override method | ❌ Không | ✅ Có |
| Tạo "phiên bản khác" của 1 class | ✅ Có | ❌ Không |
| Thay thế hoàn toàn class gốc | ❌ Không | ✅ Có |