BÁO CÁO TASK 4

# Thư viện Castle Windsor

Nó có thể cung cấp cho bạn các đối tượng với các phụ thuộc được tạo. Toàn bộ đồ thị đối tượng được tạo thông qua phản ánh và cấu hình chứ không phải là toán tử "new" và bạn có thể viết ít lỗi hơn với DRY code, được thiết kế tốt hơn theo cách có thể kiểm tra và lặp lại được.

## DependsOn – Chỉ định sự phụ thuộc

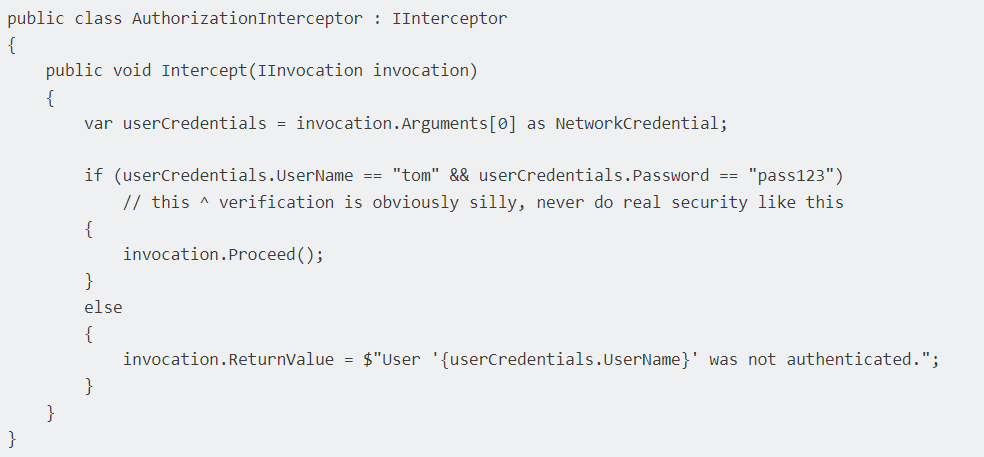
Dependency injection là một lớp không khởi tạo các phụ thuộc của nó mà yêu cầu chúng (thông qua contructor hoặc property). Sử dụng Castle để chỉ định cách giải quyết phụ thuộc là sử dụng DependsOn:

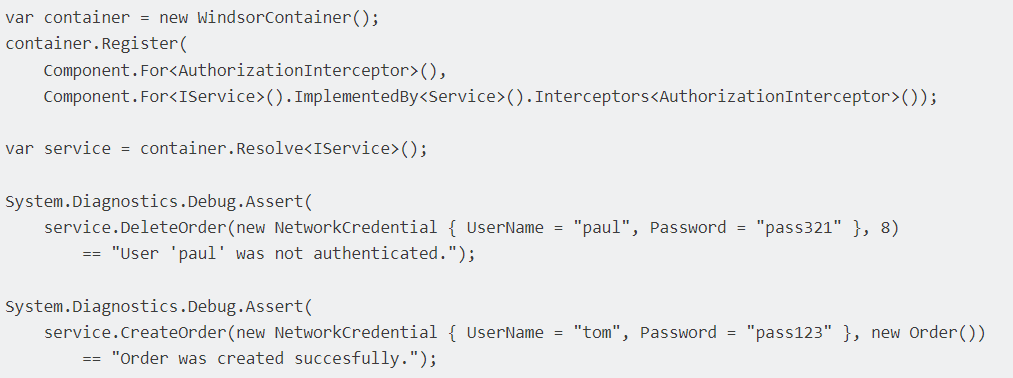


## Interceptors

Các Interceptor là để quan sát, điều chỉnh và có khả năng chặn các yêu cầu và những phản hồi.

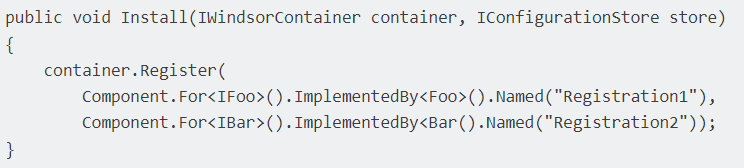
* Interceptors yêu cầu Iinterceptor interface.
* Interceptors có thể được xâu chuỗi theo một thứ tự cố định.
* Interceptors có thể có các phụ thuộc.
* Interceptors không phải gọi Proceed mỗi lần.





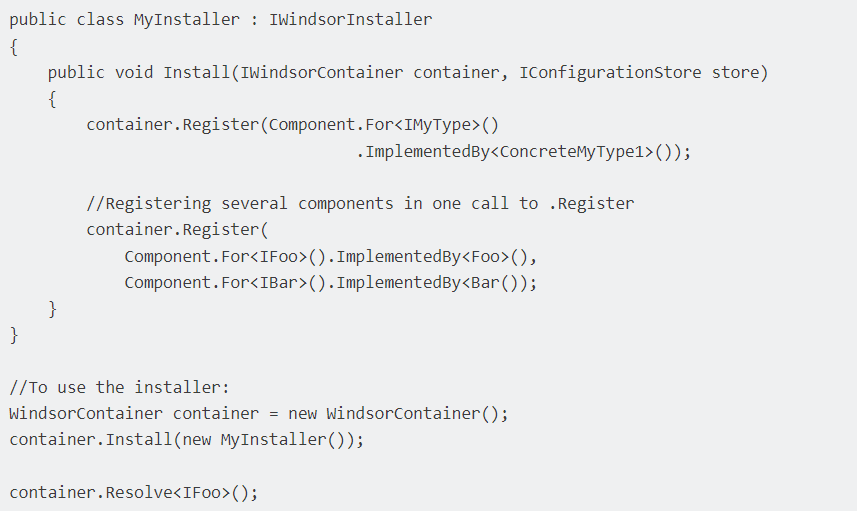
## Register một số loại Interface giống nhau

Khi đăng ký các loại cho container Castle sử dụng loại của lớp để resolve. Trong trường hợp có nhiều hơn một đăng ký cho một loại cụ thể, thuộc tính Name phải được đặt:



## Installers

Installers là loại tùy chỉnh triển khai IWindsorInstaller interface và được sử dụng để đăng ký component vào container.



Một cách khác để Installers là dùng FromAssembly class của Castle.



Loại này cho thấy một số phương pháp hữu ích để định vị assembly.



## Lifestyles

Lifestyle là "cách" Castle kiểm soát phạm vi mà một thành phần được sử dụng và thời điểm clean nó. Các lifestyles tích hợp là Singleton, Transient, PerWebRequest, Scoped, Bound, PerThread và Pooled.

* Transient: Mỗi khi thành phần được resolved, một phiên bản mới của nó được tạo ra bởi container. (không live forever).



* Singleton: Mỗi khi thành phần được resolved, cùng một thể hiện sẽ được vùng chứa trả về. (live forever – tồn tại đến khi kết thúc chương trình)

