

IoT/DI вопросы применения

авторы: Садо́мов Алексей,
Они́кийчук Антон

Чем плох new

- Дублирование кода.
- Сложность поддержки.
- Сложность изменения.
- Малая гибкость программы.
- Зависимость от конкретной реализации.
- Высокая связанность объектов.

Что такое IoC

IoC - это принцип!

IoC - это передача управления чем-то какому-то фреймворку.

```
class X:
    y = Y()
    def Foo(self):
        self.y.foo()
class Y:
    def foo(self)
def __main__():
    x = X()
    x.Foo()
```

```
class Assembler:
    def RegClass(sefl,name,class)
    def Create(self,name)
class X:
    y
    def __init__(self,assembler):
        y=assembler.Create("y")
    def Foo(self):
        self.y.foo()
class Y
    def foo(self)
def __main__()
    a = Assembler();
    a. RegClass('y',Y)
    x = X(a)
    x.Foo()
```

Паттерн DI

Преимущества:

- Паттерн легковесен.
- Не накладывает ограничения на структуру приложения
- Вводится минимальное количество дополнительных абстракций.
- Легче проследить зависимость компонентов

Недостатки:

- Сложность отладки если среда разработки не поддерживает используемый контейнер
- Нет единого механизма доступа к сервисам.
- Невозможность динамического поиска сервисов

Типы DI

- Constructor Injection - фреймворк инициализирует объект с помощью конструктора.
- Setter Injection - фреймворк инициализирует объект с помощью setters.
- Interface Injection - для настройки создается специальный интерфейс.

Паттерн Service Locator

Преимущества:

- Позволяет просто получать доступ к сервисам.
- Простой способ добавления сервисов.
- Единое сервис-API.

Недостатки:

- Сложность отладки если среда разработки не поддерживает используемый контейнер.
- Несовместимость сервис-API.
- Непрозрачность зависимостей для поставляемого объекта.

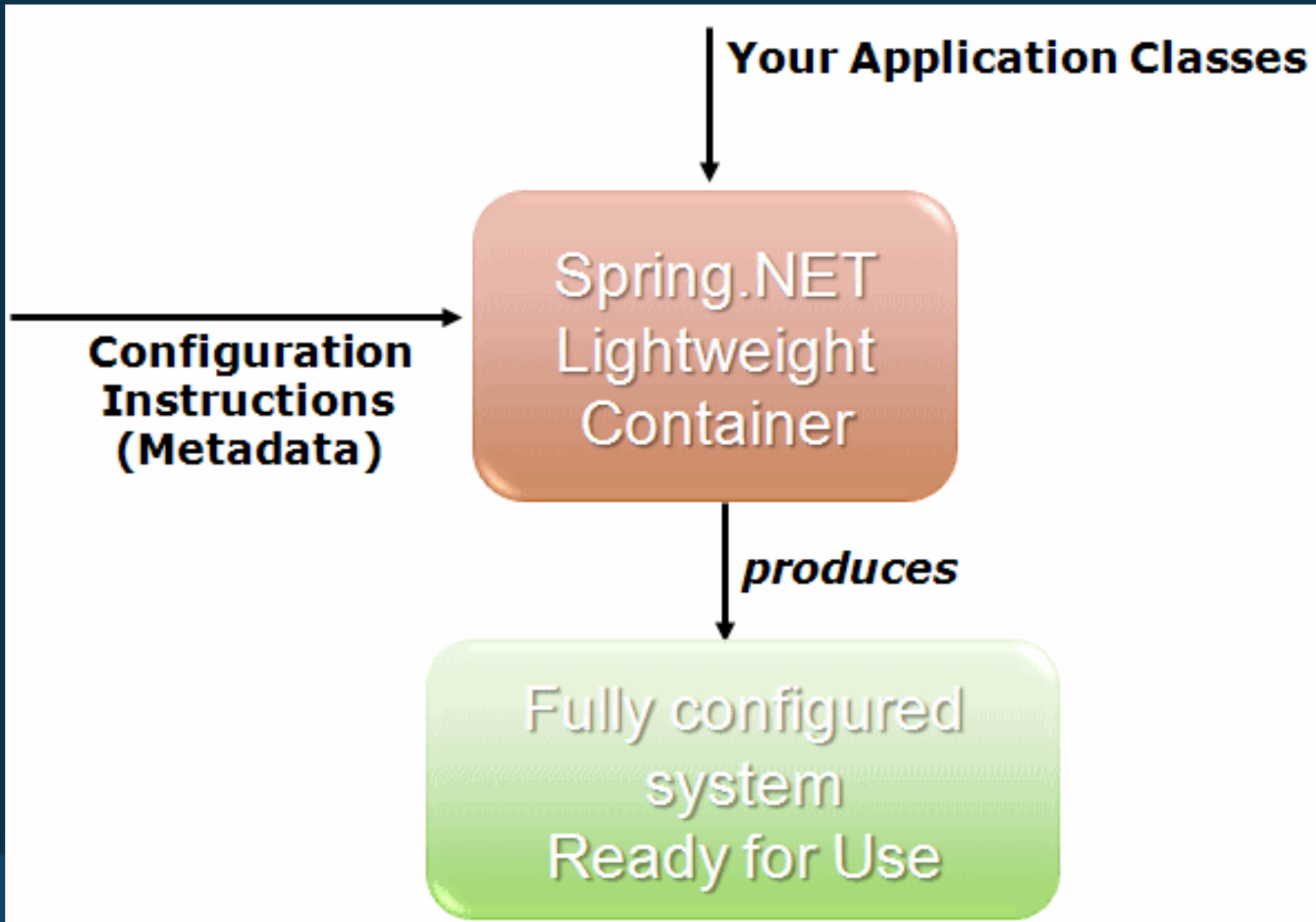
Spring .NET

легкий IoC контейнер

Модули Spring

1. Spring.Core - ядро Spring, DI фреймворк, система для создания пула объектов, фреймворк для валидации, формат логинга.
2. Spring.Aop - поддержка АОП, библиотека аспектов.
3. Spring.Data - поддержка DAO, декларативное управление транзакциями
4. Spring.Data.NHibernate - поддержка NHibernate.
5. Spring.Services - поддержка сервисов .NET
6. Spring.Web - web фреймворк от Spring
7. Spring.Web.Extencions - расширения Spring.Web дополнительные контролы
8. Spring.Scheduling.Quartz - поддержка Sheduling
9. Spring.Messaging - поддержка MOM

IoC Spring container



ApplicationContext - что такое КОНТЕКСТ.

IObjectFactory - фабрика конфигурации объектов.

ApplicationContext наследует *IObjectFactory*

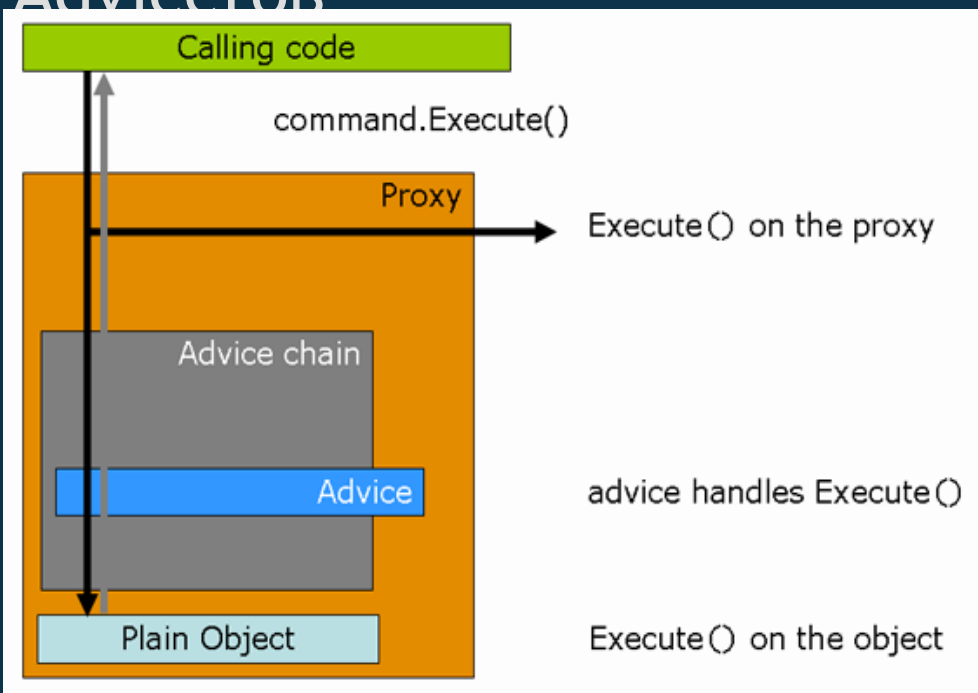
ApplicationContext предоставляет доступ к:

1. Фабрике объектов связанной с контекстом
2. Ресурсам связанным с контекстом с поддержкой интернационализации и глобализации.
3. Регистрации поставщиков и обработчиков слабо-связанных событий.

Существует реестр контекстов *ContextRegistry*

АОП в Spring

Система работы Adviserov



Proxy создается автоматически во время выполнения с помощью ProxyFactory. Для Advice можно указать специфические параметры для метода к которому он будет применяться (имя, сигнатура...). Есть разные виды Advice - вызываемые до или после метода, полностью обертывающие метод, вызываемые после возбуждения методом исключения.

AutoProxy

Возможность Spring прозрачно создавать прокси для объектов удовлетворяющим определенным условиям.

```
<object type="Spring.Aop.Framework.  
AutoProxy.  
ObjectNameAutoProxyCreator, Spring  
.Aop">  
  <property name="ObjectNames">  
    <list>  
      <value>English* </value>  
      <value>PortugeseSpeaker </val  
ue>  
    </list>  
  </property>  
  <property name="InterceptorNames"  
>  
    <list>  
      <value>debugInterceptor </val  
ue>  
    </list>  
  </property>  
</object>
```

NHibernate в Spring

```
ICollection employeeList = null;
ISession session = HibernateUtil.SessionFactory.
GetCurrentSession();
session.BeginTransaction();
try
{
    employeeList = session.CreateQuery(
        "from Employee" + " WHERE employeeCode='H'"
        + "ORDER BY name").List();
    session.Transaction.Commit();
}
catch (HibernateException e)
{
    session.Transaction.Rollback();
    throw e;
}
return employeeList;
```

```
return SessionFactory.GetCurrentSession().
CreateQuery( "from Employee" + " WHERE
employeeCode='H' + "ORDER BY name").List();
```