

# Dependency Injection

Simplified

Examples

+

real-world analogy

**swipe >>>**

# Dependency Injection

**What?**

**is a technique**

**For?**

**that allows us to remove  
hard-coded dependencies  
from our code**

**How?**

**by injecting the dependencies  
of a class through its  
constructor or methods, rather  
than hard-coding them.**

**Why?**

**can help you write  
more flexible and  
maintainable code.**

**Let's get this  
straight**



**The kitchen is the place where the chef does the main work he is in charge of**

**Chef leaves the kitchen (his sphere of interest)**



**chef going to the grocery store to buy the ingredients themselves**





**Chef stays in the kitchen to focus on preparing the meal**



**The ingredients are provided to the chef by someone else.**



## **Explanation**



- The chef has a recipe that specifies the ingredients that are needed to make the meal.
- Rather than the chef going to the grocery store to buy the ingredients themselves...
- ...the ingredients are provided to the chef by someone else.
- This allows the chef to focus on preparing the meal, rather than worrying about where to get the ingredients

In the same way

**dependency injection allows a class to focus on its core functionality, rather than worrying about where to get its dependencies.**

**Example =>**

# Without Dependency Injection



```
class EmailSender
{
    private SmtpClient smtpClient = new SmtpClient();

    public void SendEmail(string to, string subject, string body)
    {
        var message = new MailMessage("sender@example.com",
                                      to, subject, body);
        smtpClient.Send(message);
    }
}
```

## Problem:

The **EmailSender class is tightly coupled to the SmtpClient class.**

If we want to use a different **SmtpClient implementation**, we would have to modify the **EmailSender class**, which can be inflexible and error-prone.

# With Dependency Injection



```
class EmailSender
{
    private SmtpClient smtpClient;

    public EmailSender(SmtpClient smtpClient)
    {
        this.smtpClient = smtpClient;
    }

    public void SendEmail(string to, string subject, string body)
    {
        var message = new MailMessage("sender@example.com",
                                      to, subject, body);
        smtpClient.Send(message);
    }
}
```

## Solution:

**Allows the caller of the EmailSender class to provide the SmtpClient instance, rather than the EmailSender class creating it itself. This makes the code more flexible and easier to maintain, because we can use a different SmtpClient implementation without having to modify the EmailSender class.**

---

Stefan Djokic



**WANT MORE POSTS LIKE THIS?**

**CLICK ON THE NOTIFICATION  
BELL ON MY PROFILE** 