Avoiding Nulls with the Maybe Type

Vladimir Khorikov
PROGRAMMER

@vkhorikov www.enterprisecraftsmanship.com

The Billion-dollar Mistake

```
string someString = null;
Customer customer = null;
Employee employee = null;
```

“I call it my billion-dollar mistake. It has caused a billion dollars of pain and damage in the last forty years.”

Tony Hoare

The Billion-dollar Mistake

```
public class Organization
{
    public Employee GetEmployee(string name)
    {
        /* ... */
    }
}
```

```
public class OrganizationRepository
{
    public Organization GetById(int id)
    {
        /* ... */
    }
}
```

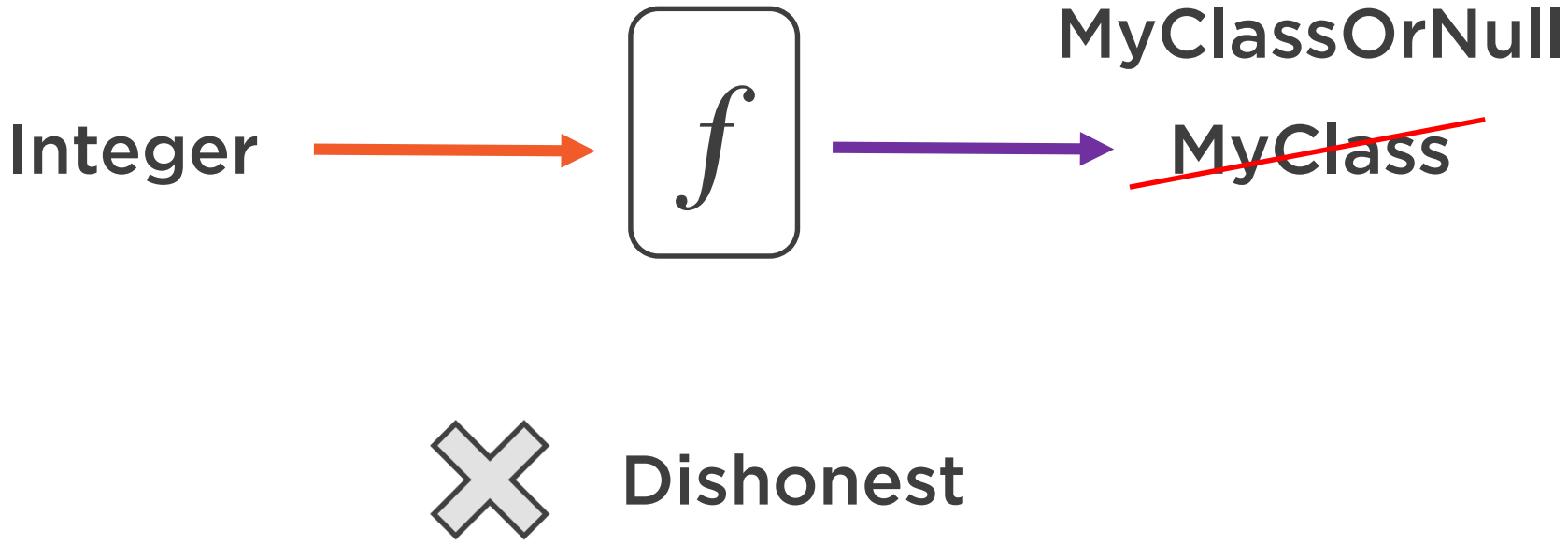
The Billion-dollar Mistake

```
public class MyClass
{
}
```

```
public class MyClassOrNull
{
    // either null
    public readonly Null Null;

    // or actually a MyClass instance
    public readonly MyClass MyClass;
}
```

The Billion-dollar Mistake



The Billion-dollar Mistake

```
Organization organization = _repository.GetById(id);
Console.WriteLine(organization.Name);
```

The Billion-dollar Mistake

```
Organization organization = _repository.GetById(id);

/* Other code */

Console.WriteLine(organization.Name);
```

Non-nullability on the Language Level

```
Organization! organization = _repository.GetById(id);
Console.WriteLine(organization.Name);
```

```
public class OrganizationRepository
{
    public Organization! GetById(int id)
    {
        /* ... */
    }
}
```

Non-nullability on the Language Level

```
Organization nullable = GetOrganization(id);
Organization! nonNullable = nullable;    // Error
```

```
Organization! nonNullable = GetOrganization(id);
Organization nullable = nonNullable;    // Ok
```

```
Organization organization = null; // Compiler error
Organization? organization = null; // Ok
```

http://bit.ly/1TW4ofH **http://bit.ly/1VTxlIi**

Mitigating the Billion-dollar Mistake

`Maybe<T>`

Mitigating the Billion-dollar Mistake

```
public class OrganizationRepository
{
        Maybe<Organization>
    public Organization GetById(int id)
    {
        /* ... */
    }
}
```

Mitigating the Billion-dollar Mistake

```
public class OrganizationRepository
{
    public Maybe<Organization> GetById(int id)
    {
        /* ... */
    }
}

public class Organization
{
    public Employee GetEmployee(string name)
    {
        /* ... */
    }
}
```

Mitigating the Billion-dollar Mistake

```
Maybe<T>  =  Nullable<T>
```

Mitigating the Billion-dollar Mistake

```
public class OrganizationRepository
{
    public Maybe<Organization> GetById(int id)
    {
        /* ... */
    }
}
```

Honest

```
public static int? Divide(int x, int y)
{
    if (y == 0)
        return null;

    return x / y;
}
```

Honest

Mitigating the Billion-dollar Mistake

```
Maybe<Organization> nullable = GetOrganization(id);
Organization nonNullable = nullable;    // Error
```

```
Organization? nullable = GetOrganization(id);
Organization nonNullable = nullable;    // Error
```

Mitigating the Billion-dollar Mistake

```
Organization nonNullable = GetOrganization(id);
Maybe<Organization> nullable = nonNullable;    // Ok
```

```
Organization nonNullable = GetOrganization(id);
Organization? nullable = nonNullable;    // Ok
```

Enforcing the Use of the Maybe Type

```
ProcessOrganization(null);


private void ProcessOrganization(Organization organization)
{
    // Method body
}
```

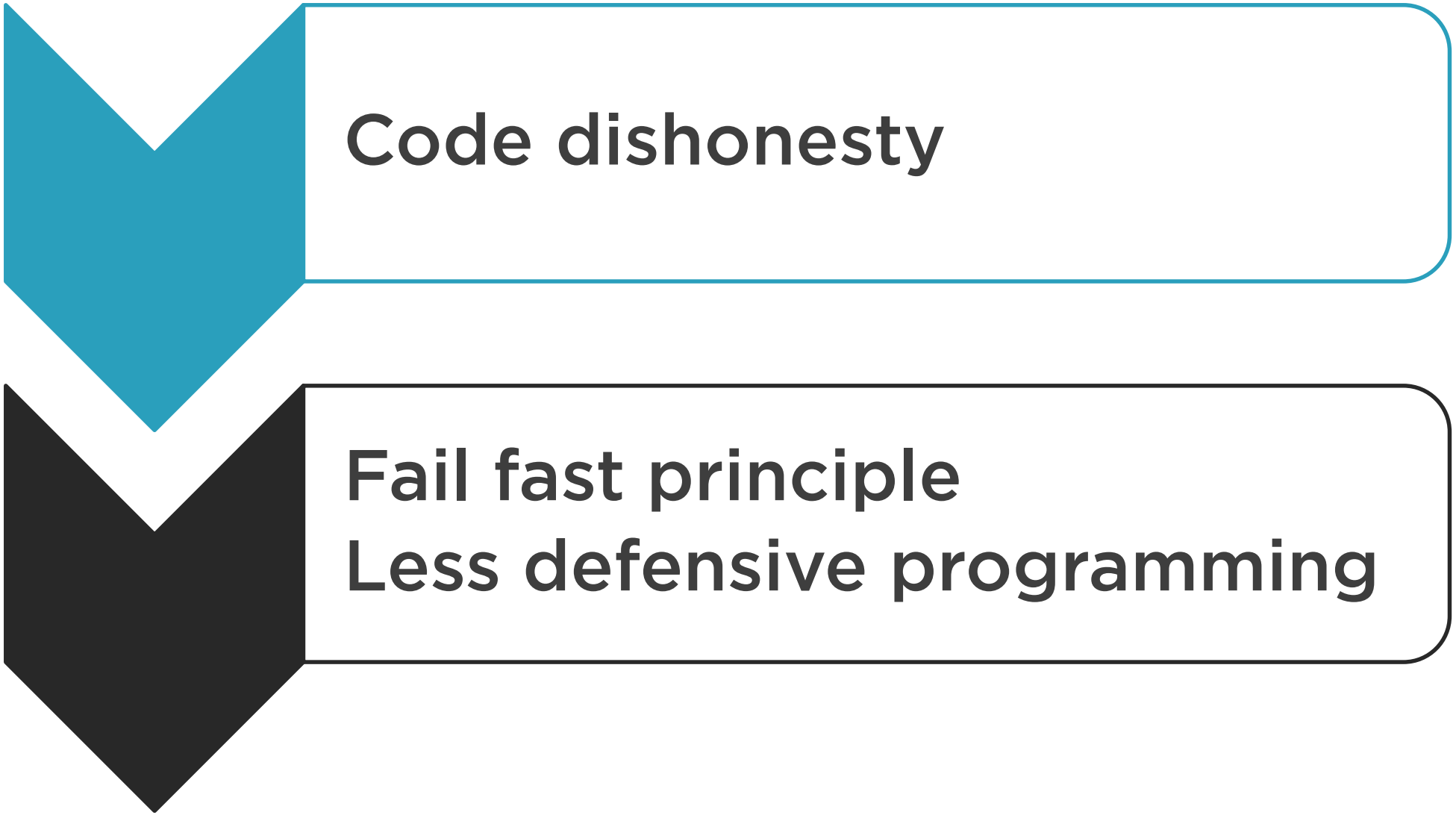
Enforcing the Use of the Maybe Type

```
ProcessOrganization(null);


private void ProcessOrganization(Organization organization)
{
    if (organization == null)
        throw new ArgumentNullException(nameof(organization));

    // Method body
}
```

Recap: Mitigating the Billion-dollar Mistake



Limitations

Decide which assemblies should be weaved

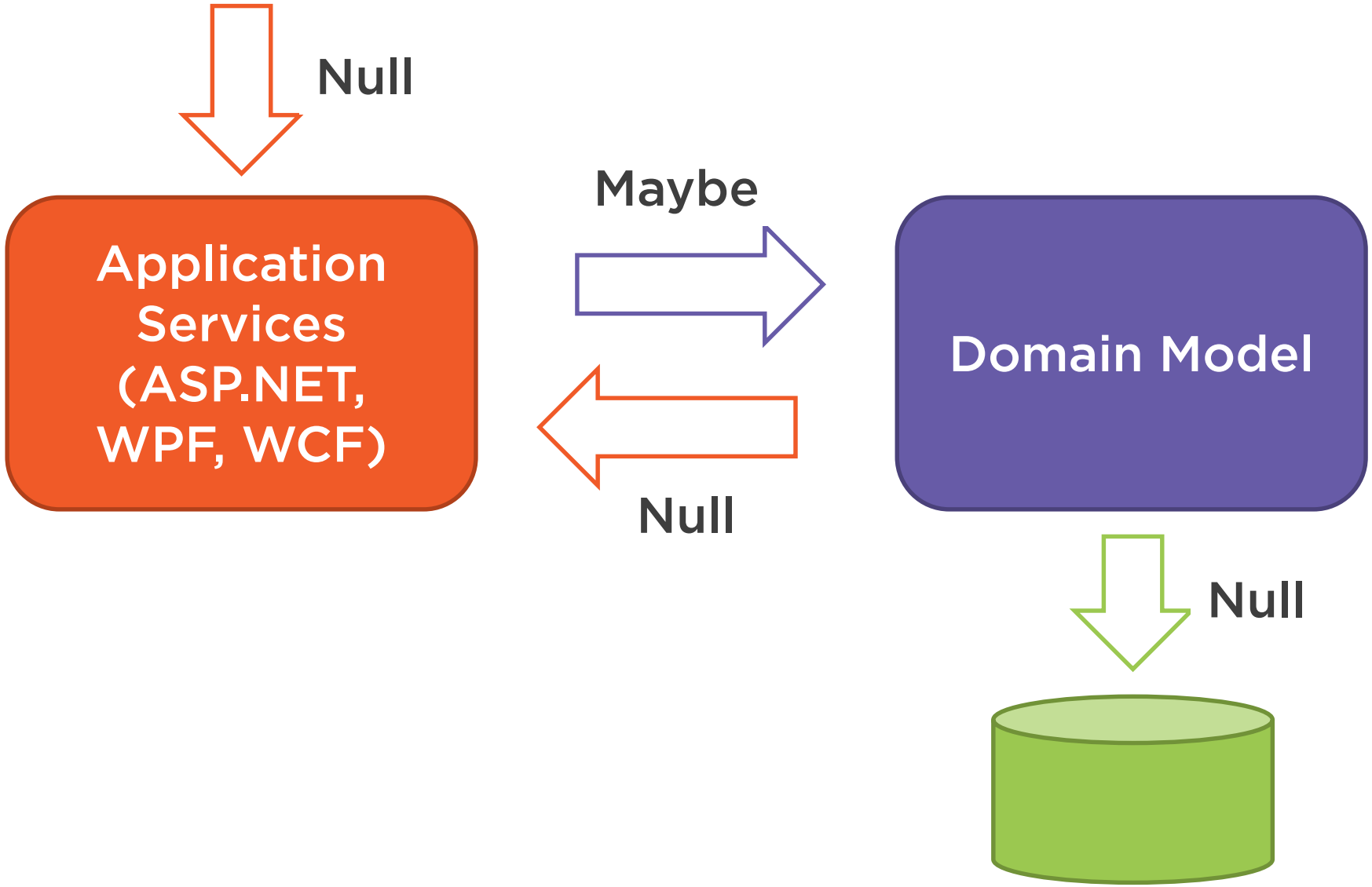
Limitations

WPF 

ASP.NET 

Domain Logic 

Guidelines


Null
Application Services (ASP.NET, WPF, WCF)
Maybe
Null
Domain Model
Null

Summary

Nulls references

- Make your code dishonest
- Contradict the fail fast principle

The Maybe type helps tackle the dishonesty problem

Fody.NullGuard

- Helps with the fail fast principle
- Reduces the effort for defensive programming

Convert nulls into Maybe when they enter the domain model

Convert them back to nulls when they leave the domain model

In the Next Module

Handling Failures and Input Errors in a Functional Way