MVVM NEDIR?

HAZIRLAYAN: ÖYKÜ PARLAK

IÇERIK

- PROBLEM TANIMI
- DESIGN PATTERN KAVRAMI
- MVVM KAVRAMI
- KAYNAKÇA

PROBLEM TANIMI

Yeni bir projeye başlayınca, bazı **kurallar** dahilinde ilerlemeye dikkat etmeliyiz. Bu kurallar projenin daha **anlaşılabilir**, **yeniden kullanılabilir** ve **sürdürülebilir** olmasını sağlar.

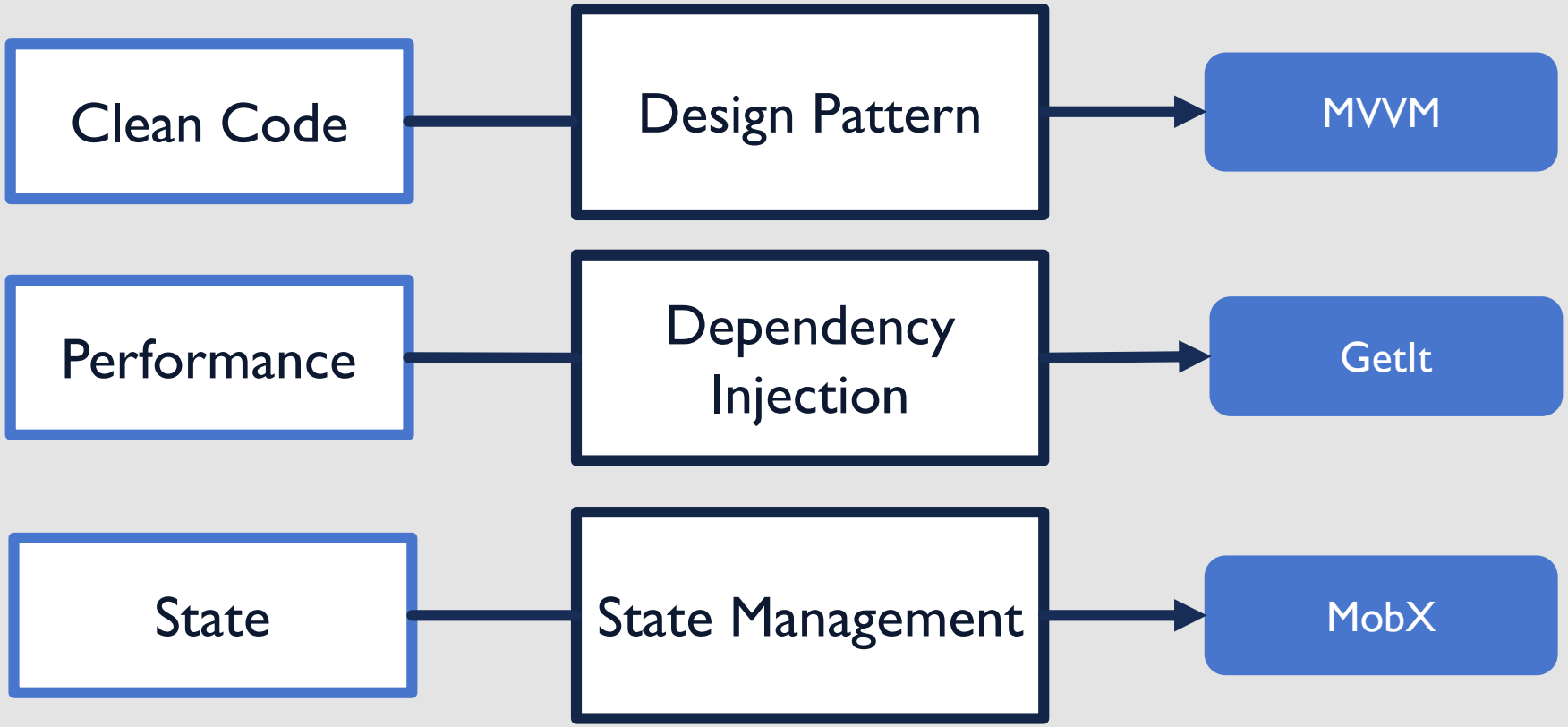
İlk aşamada **design pattern** devreye giriyor. Birçok alternatif pattern arasından seçilen birini kullanmak hem üsttekileri kapsar hem de kod yazmamız için bize **kolaylık** sağlar.

Ayrıca bu problemlere karşılık gelen çözümler birbirinden farklı olabilir veya hiçbir kütüphane kullanılmadan da üstesinden gelinebilir.

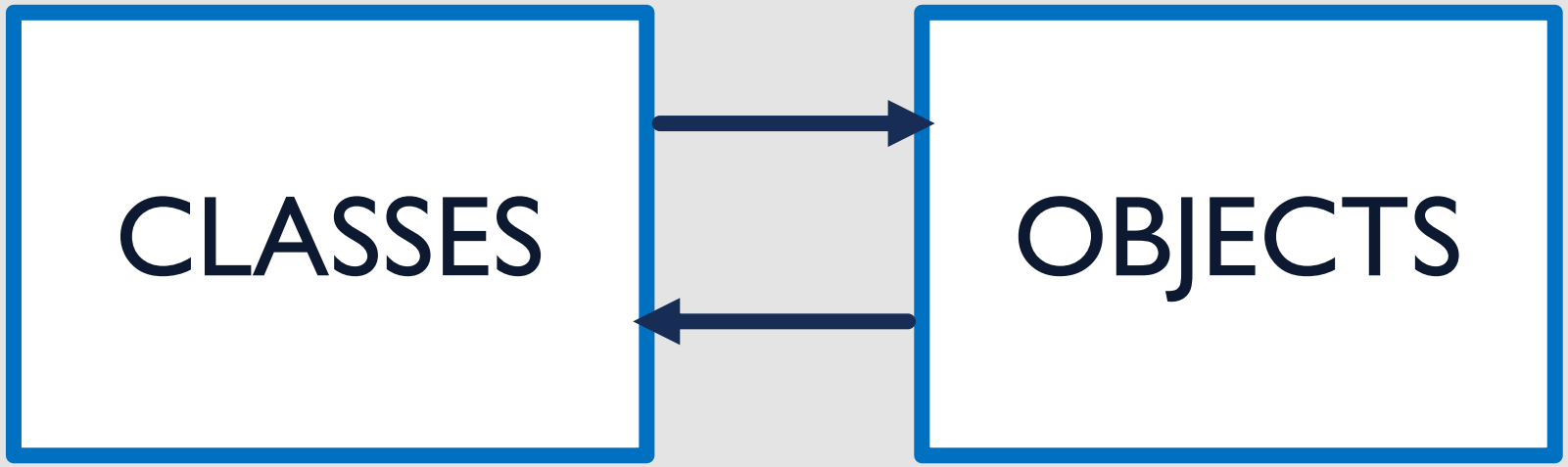
SORUNUN AÇIKLANMASI

- Belirli Kılavuz

- Comprehensible
- Reusable
- Sustainable



DESIGN PATTERN NEDIR?

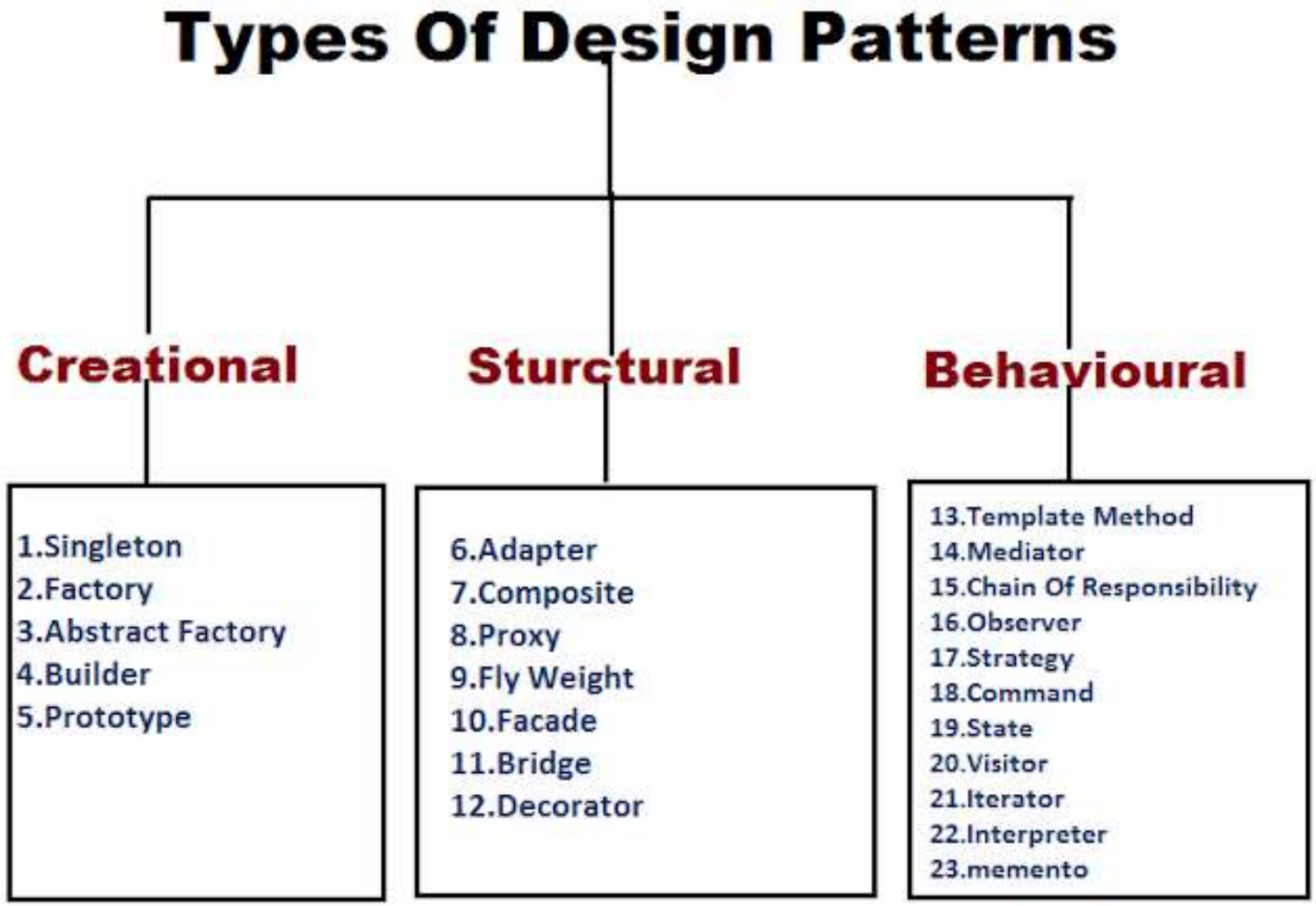


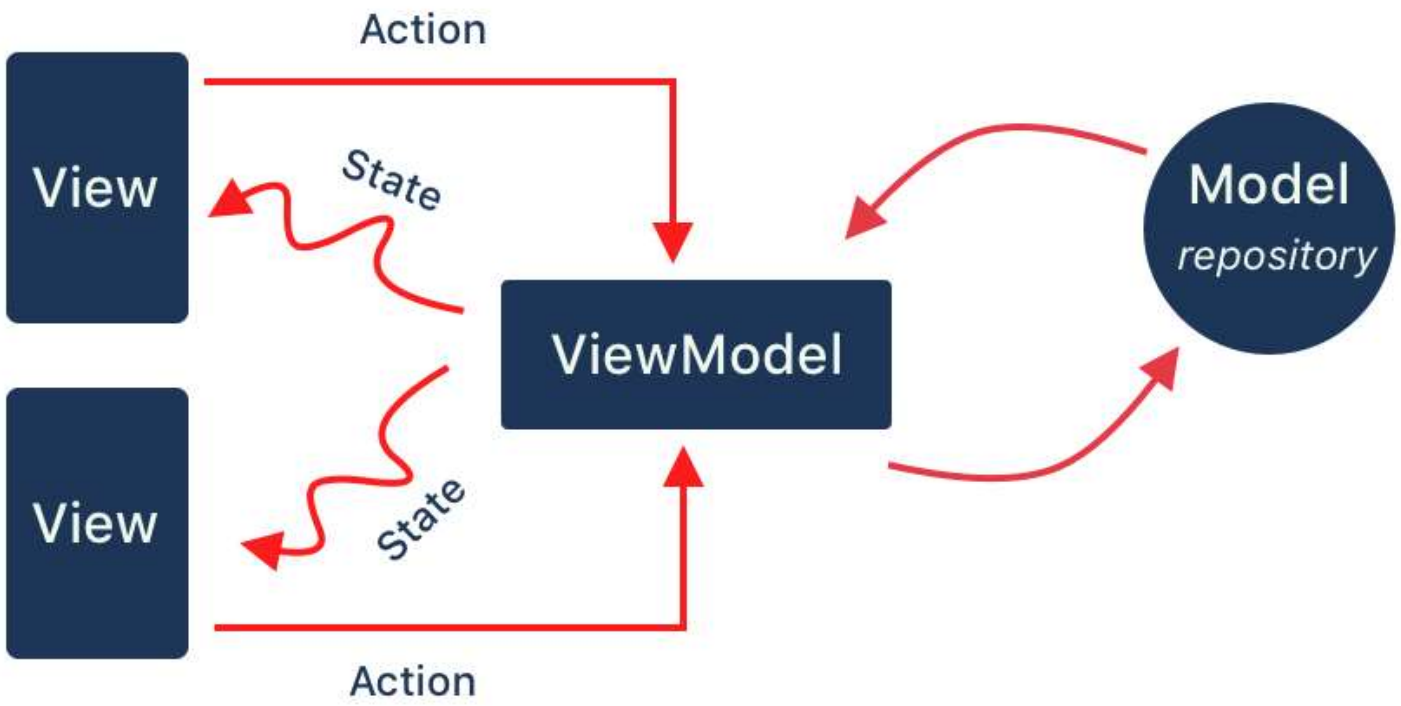
FLEXIBLE

REUSABLE

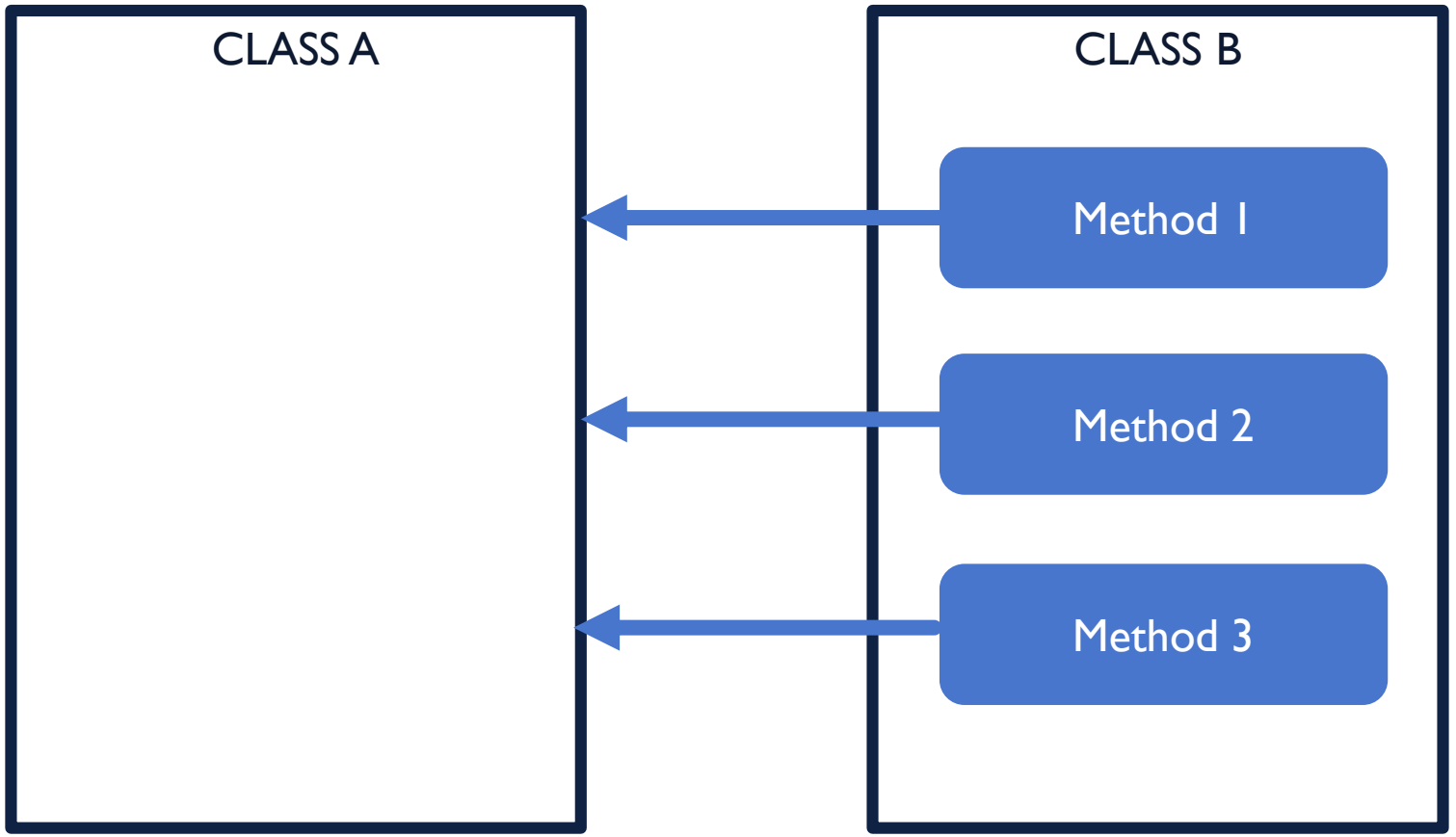
MAİNTAİNED

- CREATIONAL (YARATIŞSAL): Kalıtım ile sınıf soyutlaştırma veya yetkilendirerek nesne oluşturma yapılır.
- STRUCTURAL (YAPISAL): Yapıyı büyütmek ve yeni fonksiyonlar sağlamak için kullanılır.
- BEHAVİOURAL (DAVRANIŞSAL): Nesnelerin kendi aralarındaki iletişimini belirlemeyi ve yönetmeyi temel alır.




Action
View
State
Model
repository
ViewModel
View
State
Action

MVVM NEDIR?

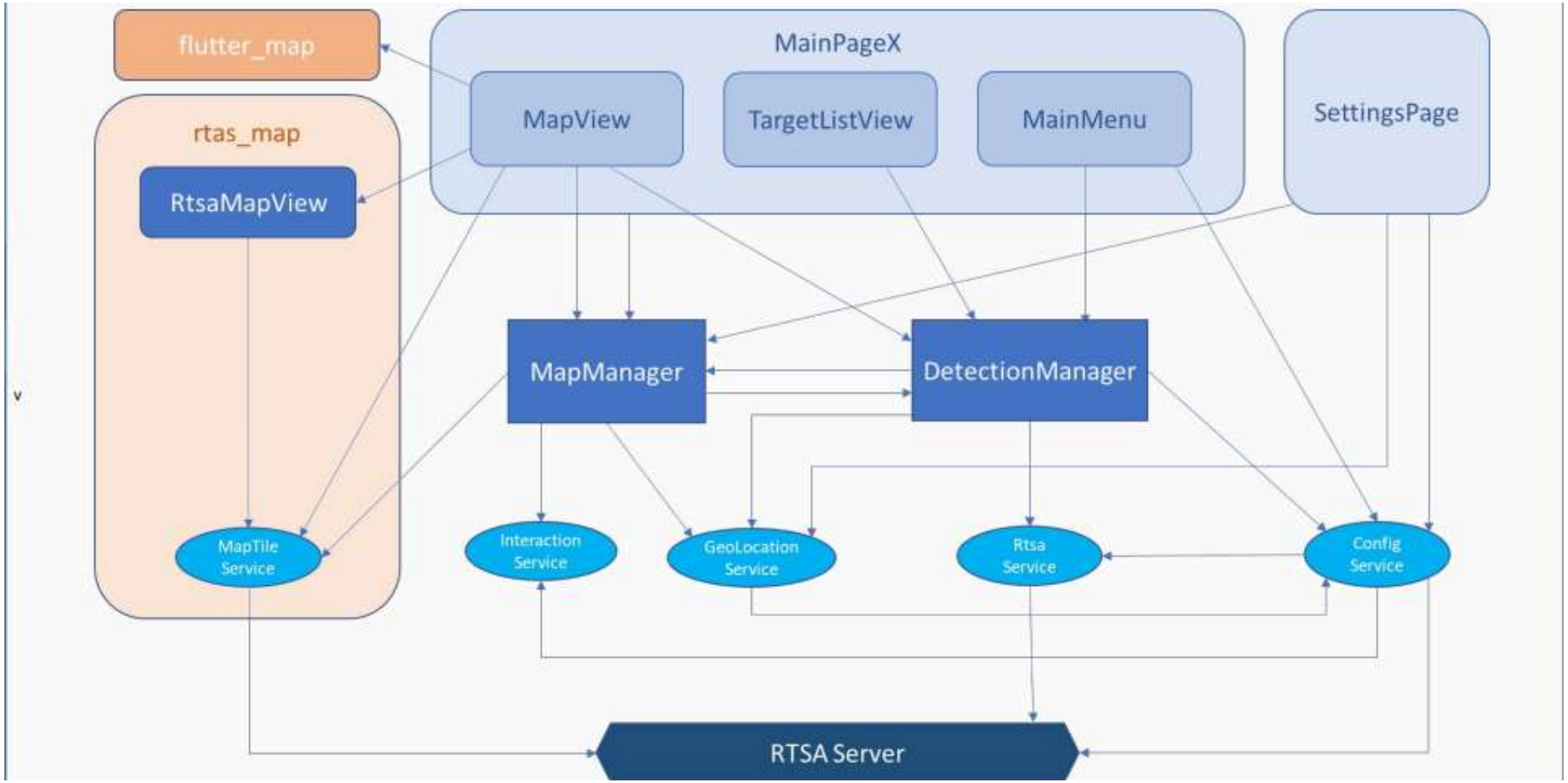


DEPENDENY INJECTION NEDIR?

EN TEMEL YAKLAŞIM..

```
class MyClass {
    // Creating service object
    MyService service;
    //Importing this class to constructor with
    MyClass(this.service);
}
```

TEMEL YAKLAŞIMIN EKSIK KISMI



GET IT KÜTÜPHANESI

Factory

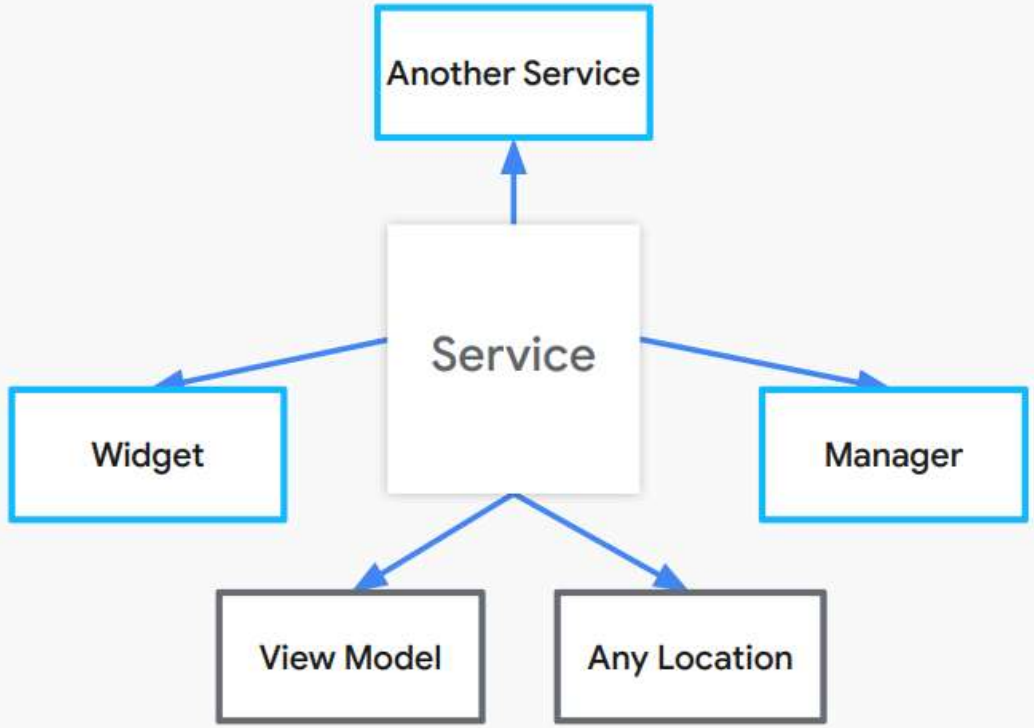
- Bir servisi her istediğimizde tekrardan oluşmasını istediğimiz durumda kullanıyoruz.

Singleton

- Sınıfınızdan sadece **bir adet nesne** üretilmesini sağlayan ve ikinci kez daha **nesne** üretilmesini engelleyen bir tasarım desenidir

Lazy Singleton

- Bu yaklaşımda ise **biz istediğimizde** hafızada bir instance oluşturulur**.** Bu sayede **uygulama ayağa kaldırılırken** hafızada **gereksiz yer kaplamaz**.



GET IT NASIL KULLANILIR?

```dart
import 'package:get_it/get_it.dart'

// We create an instance object from it.
GetIt locator = GetIt.instance;

// We define a method in which we register our services.
void setupLocator() {
    locator.registerFactory(() => ThemeService());
    locator.registerSingleton(() => DatabaseService());
    // We use this
    locator.registerLazySingleton(() => ViewModel());
}
```

```dart
void main() {
    setupLocator();
    runApp(MyApp());
}
```

```dart
final ViewModel viewModel = locator<ViewModel>();
```

STATE MANAGEMENT NEDIR?

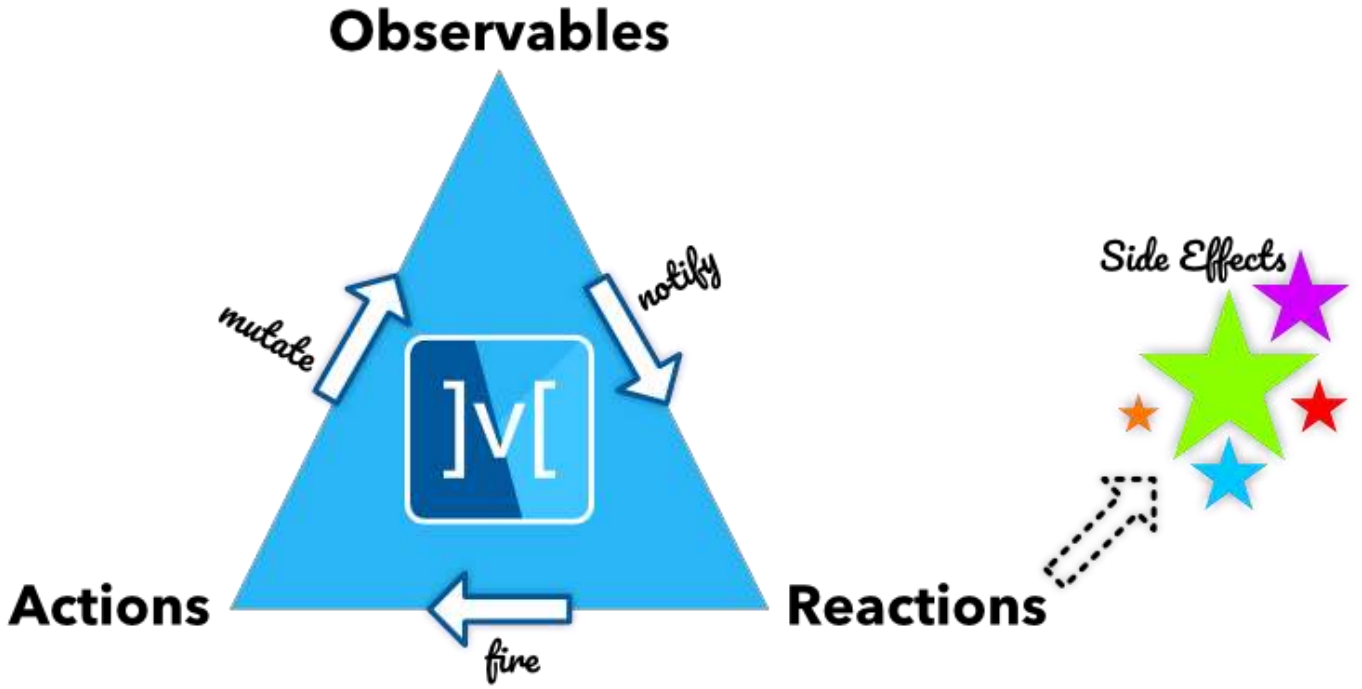
- Buna neden ihtiyacımız var?

Program akışını tüm bileşenler arasında senkronize etmek için

setState()--->?

MOBX KÜTÜPHANESİ

- Kullanıcının tepkisi, gözlemlenebilir verileri değiştiren ve arayüze bir bildirimle yanıt veren eylemi başlatır.



MOBX IÇIN 3 TEMEL ANA BILEŞEN

- Observables

Gözlemlemek istediğimiz değişken için kullandığımız yapı.

- Computed Observables

En basit haliyle gözlemlemek istediğimiz değişkenlerdeki değişimleri algılayıp ona göre değişen yapı.

- Actions

İçerisinde observableslarımızı tanımladığımız methodlarımız

```
@observable
int value = 0;
```

```
@computed
bool get isValueEven => value.isEven;
```

```
@action
void increment() {
    value++;
}
```

MOBX'İN KENDİNE AİT WİDGET'I OBSERVER WIDGET

- Observer widget bizim kendi içinde gözlemlenebilir değerlerin widgetlarını sarmaladığımız widget.
- Sadece observable değerlerin olduğu widget'ı sarmalıyız.

```
Observer(builder: (_) {
        return Text('${counter.value}',);
})
```

MOBX'İN ÇALIŞMA SÜRECİ

```dart
import 'package:mobx/mobx.dart'
part 'view_model.g.dart'

class ViewModel = _ViewModelBase with _$ViewModel;

abstract class _ViewModelBase with Store {

}
```

```dart
// for first run
flutter pub run build_runner watch
// if you are getting an error
flutter pub run build_runner watch -delete-conflicting-outputs
```