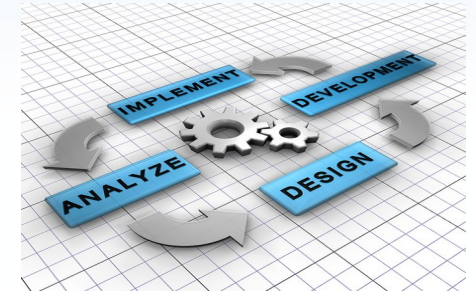
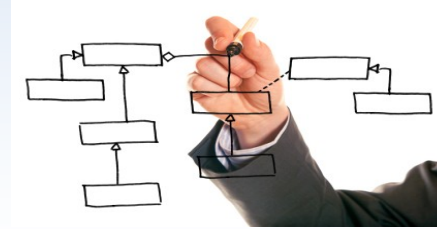


MVP, Observer ve Mediator Örüntüleri ile Yeniden Kullanılabilir Uygulama Bileşenleri Geliştirme

Kenan Sevindik Kimdir?

- 1999 ODTÜ **Bilgisayar Müh.** mezunu
- 15 yılın üzerinde **kurumsal uygulama geliştirme deneyimi** var
- Pek çok projenin geliştirilmesinde görev aldı
- Object Oriented ve Aspect Oriented Programlama, Tasarım Örüntüleri gibi konularda,
- Spring, Spring Security, Hibernate, Vaadin gibi **kurumsal Java teknolojilerinde** bilgi birikimi ve deneyime sahip

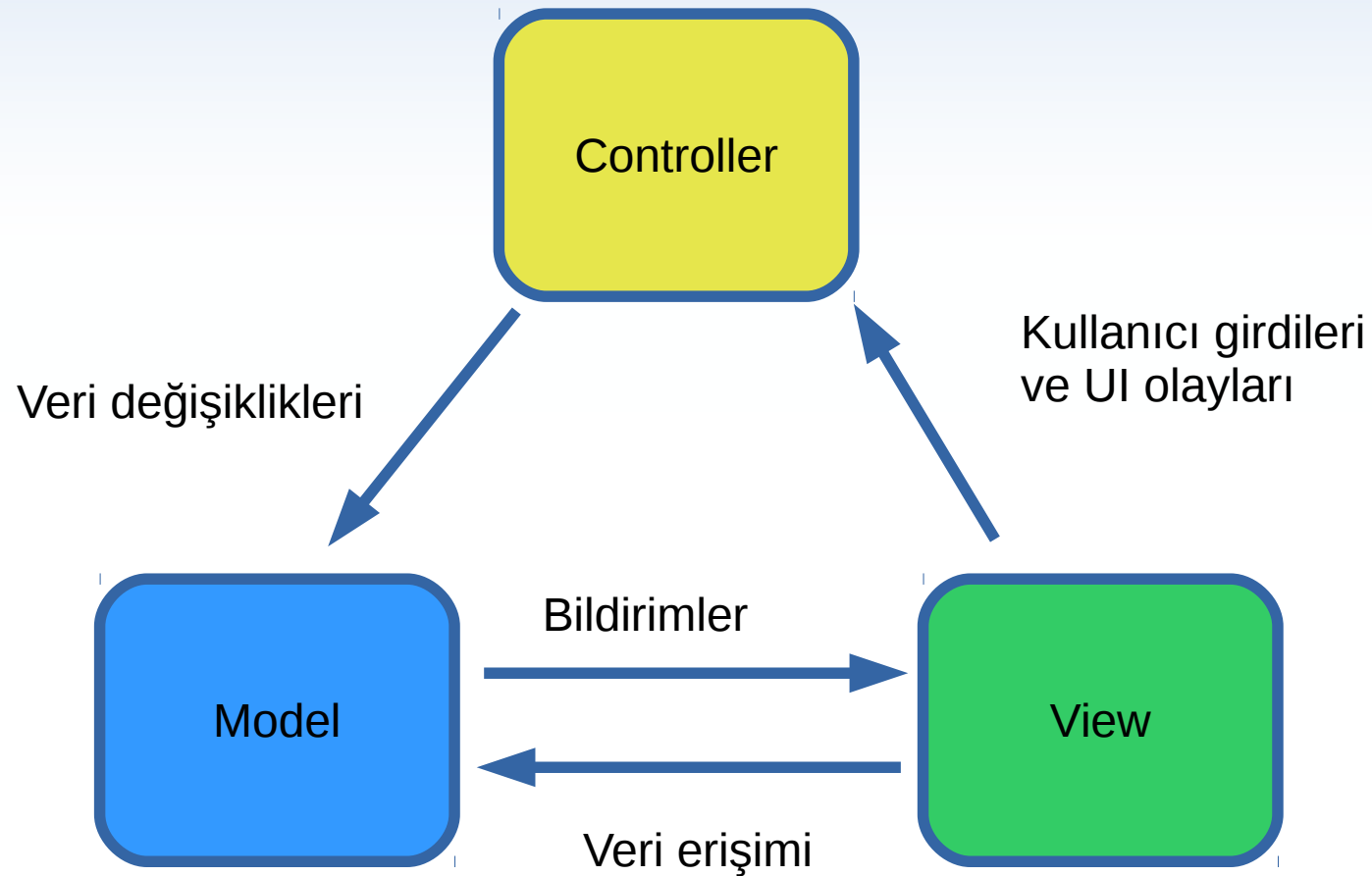


Kenan Sevindik Kimdir?

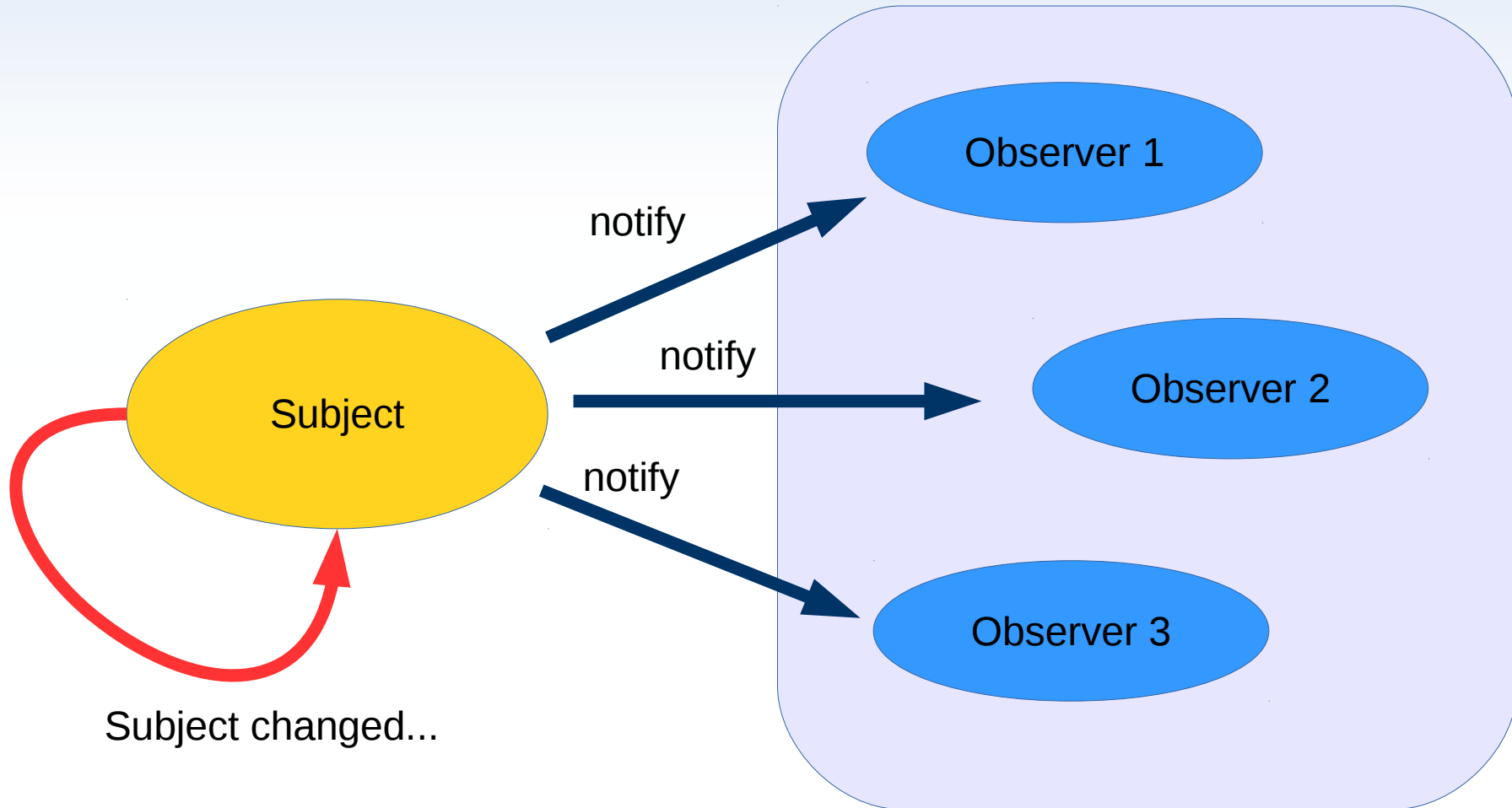
- **Beginning Spring** kitabının yazarlarından
- 2011 yılında **Harezmi Bilişim Çözümleri**ni kurdu
 - Kurumsal uygulama geliştirme faaliyetleri yürütüyor
 - Danışmanlık ve koçluk hizmetleri sunuyor
 - Kurumsal Java Eğitimleri adı altında eğitimler düzenliyor



Mimarisel Bir Örüntü: MVC



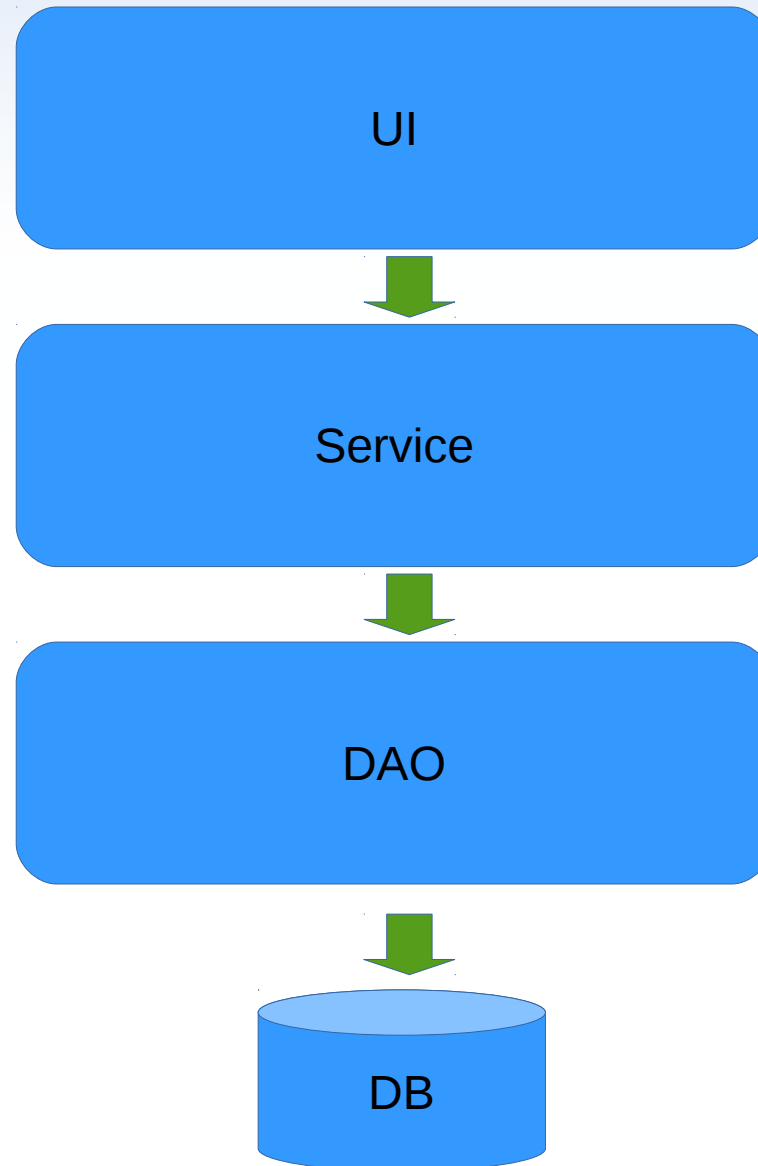
MVC & Observer



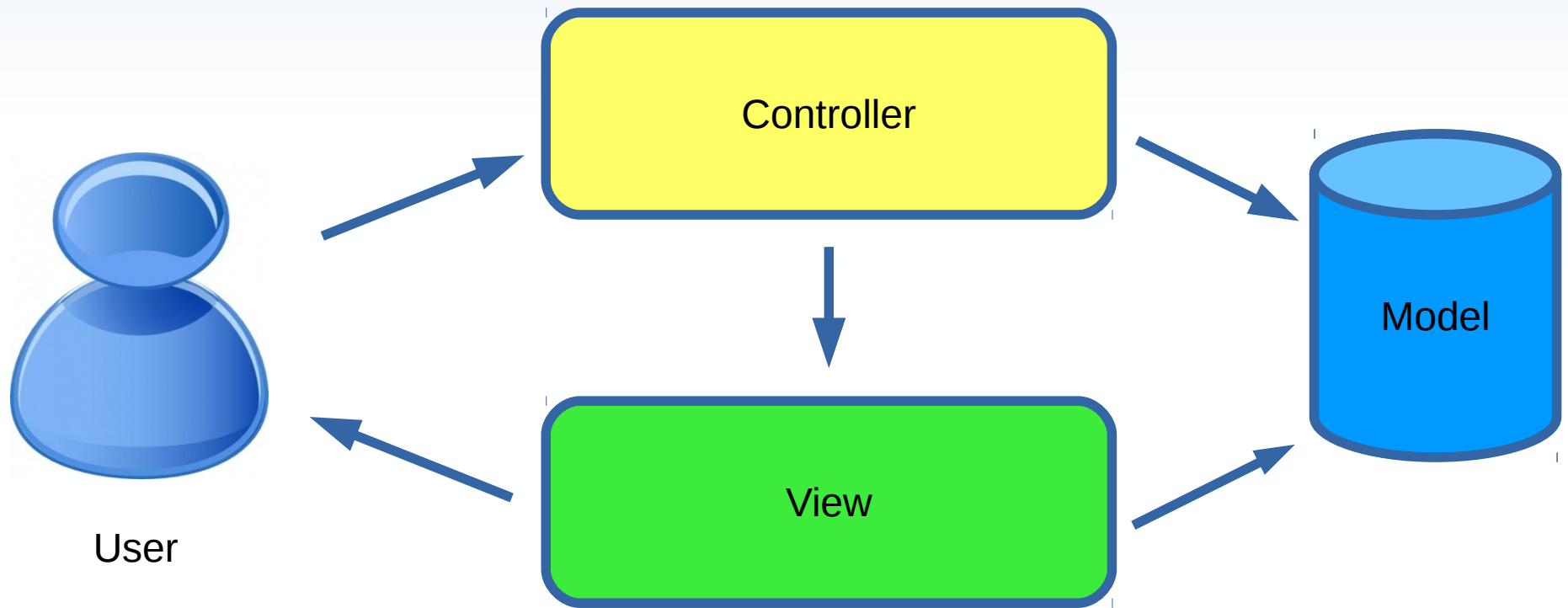
MVC'nin Temel İşlevi

“Seperation of Concern”

Kurumsal Uygulamalarda Katmanlı Mimari

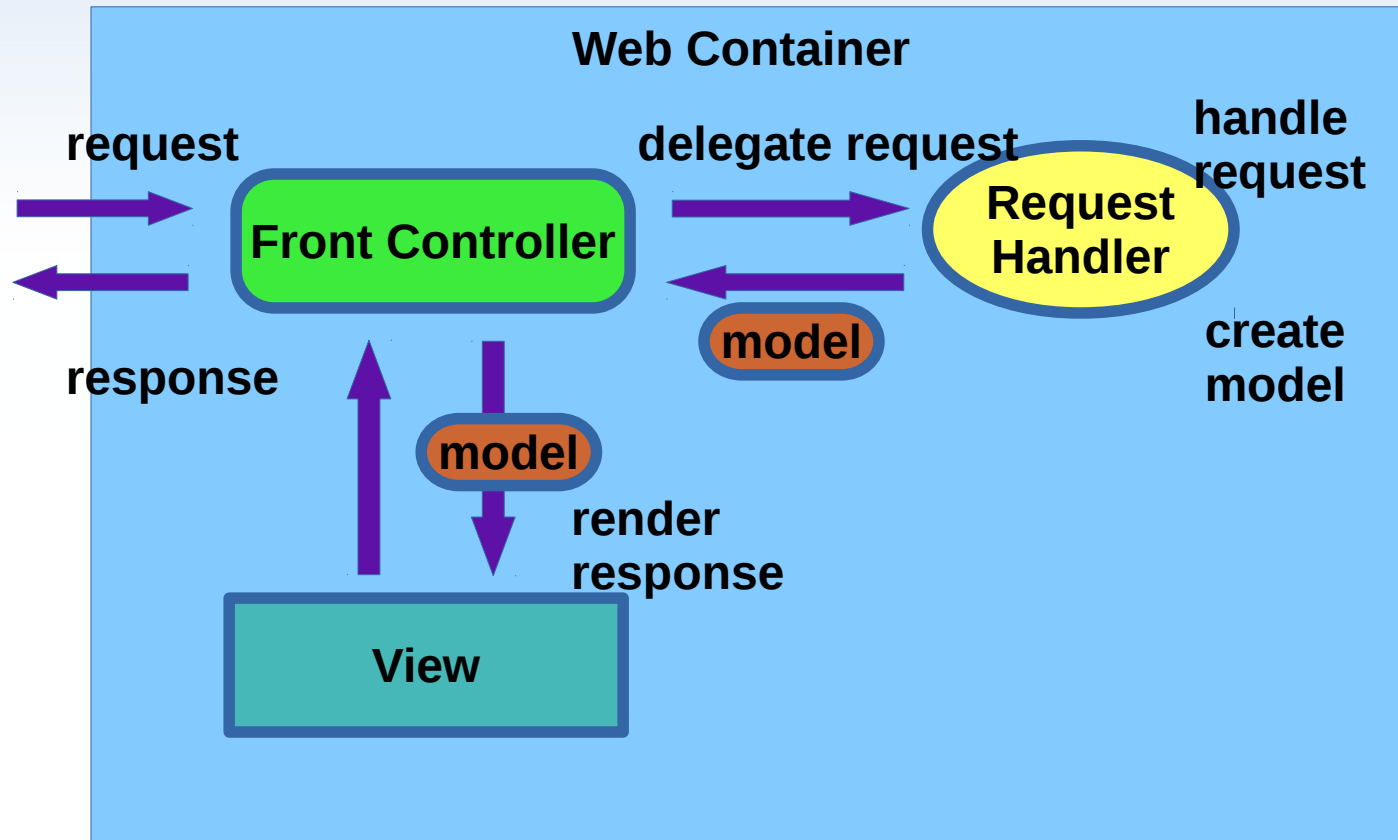


Kurumsal Java Dünyasında MVC Yorumlaması



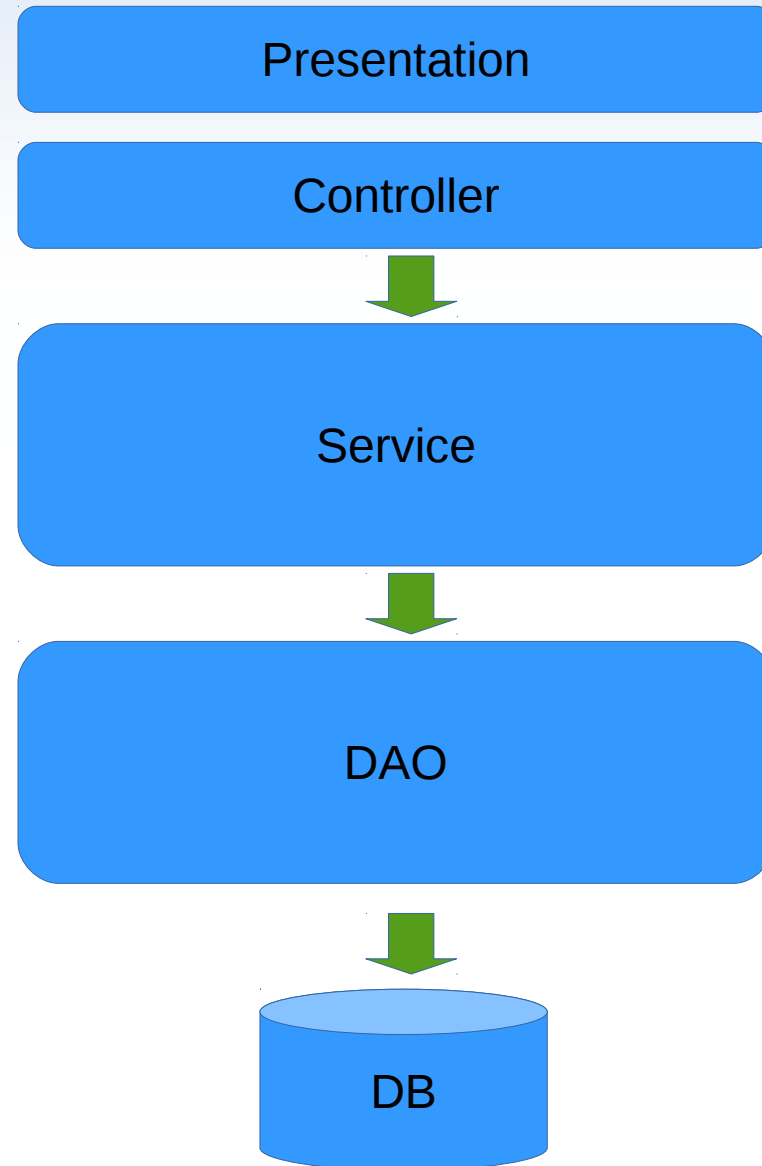
Web MVC veya MVC2 olarak adlandırılmıştır

Web MVC ve Front Controller

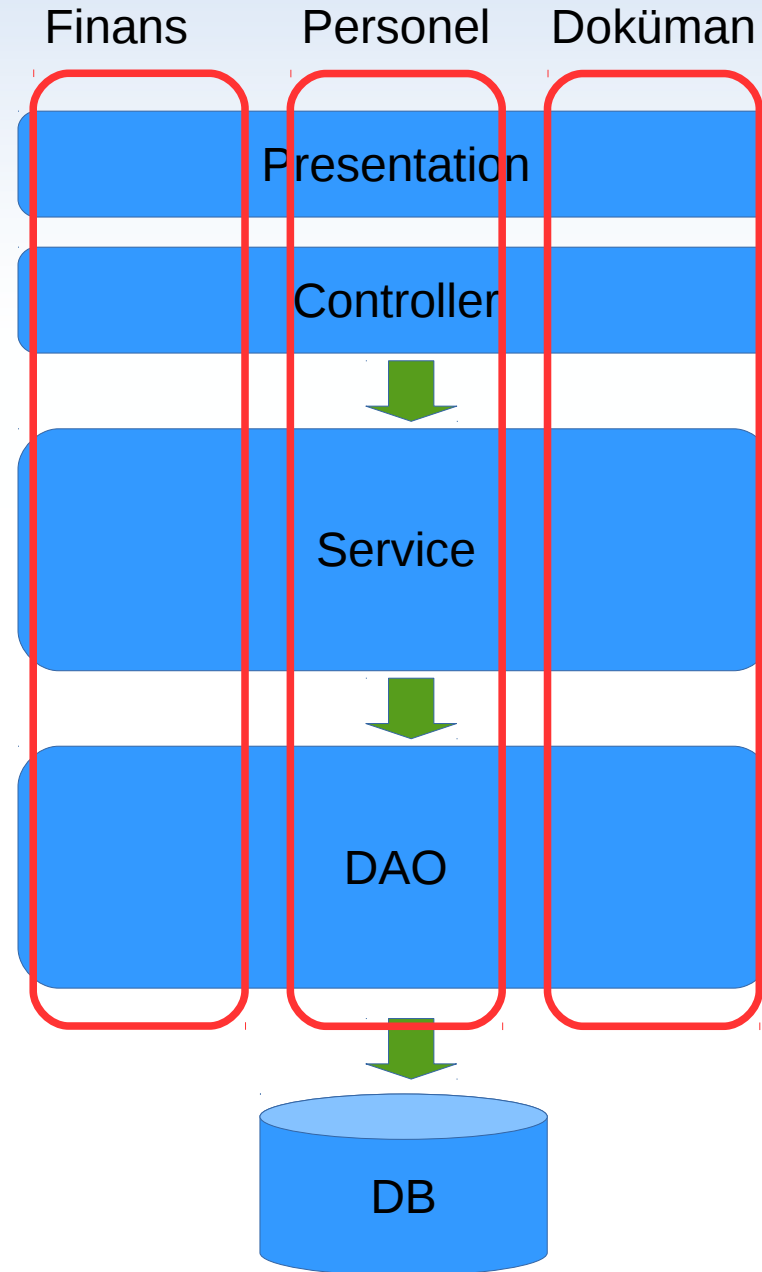


Struts, Spring MVC, JSF gibi pek çok Web Framework'üne temel teşkil etmiştir

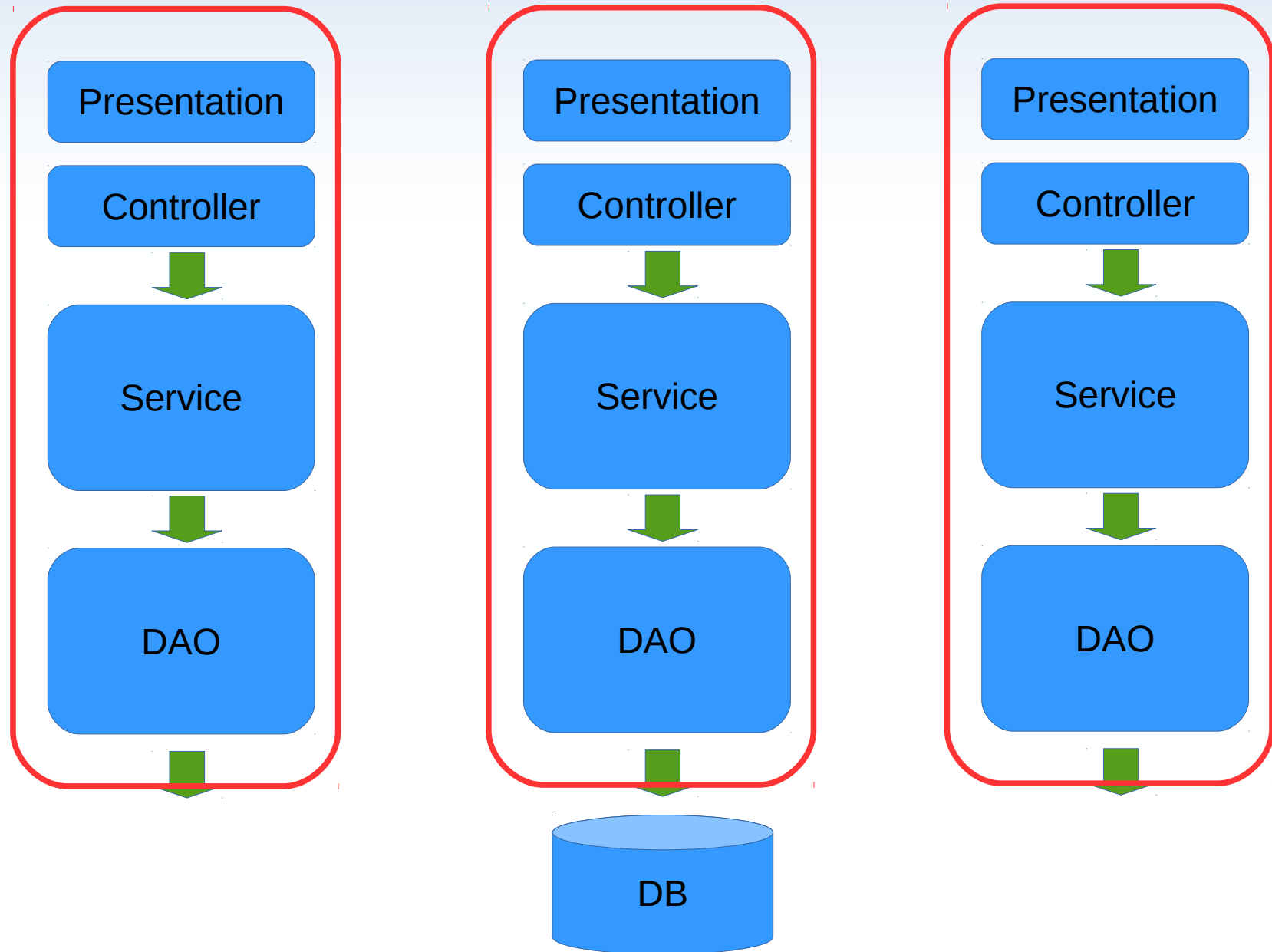
Kurumsal Uygulamalarda Katmanlı Mimari



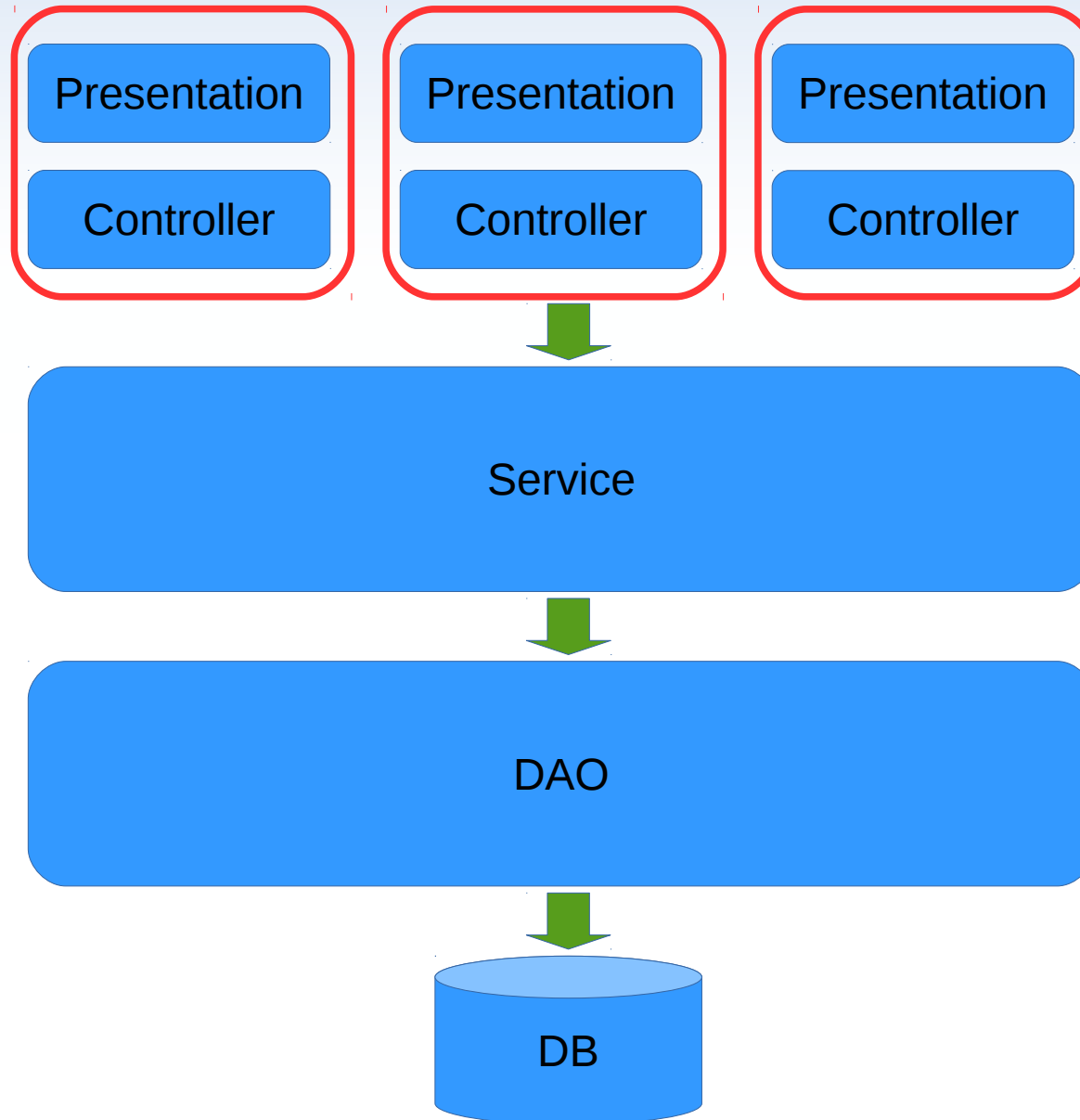
Katmanlı Mimari ve Modülerlik



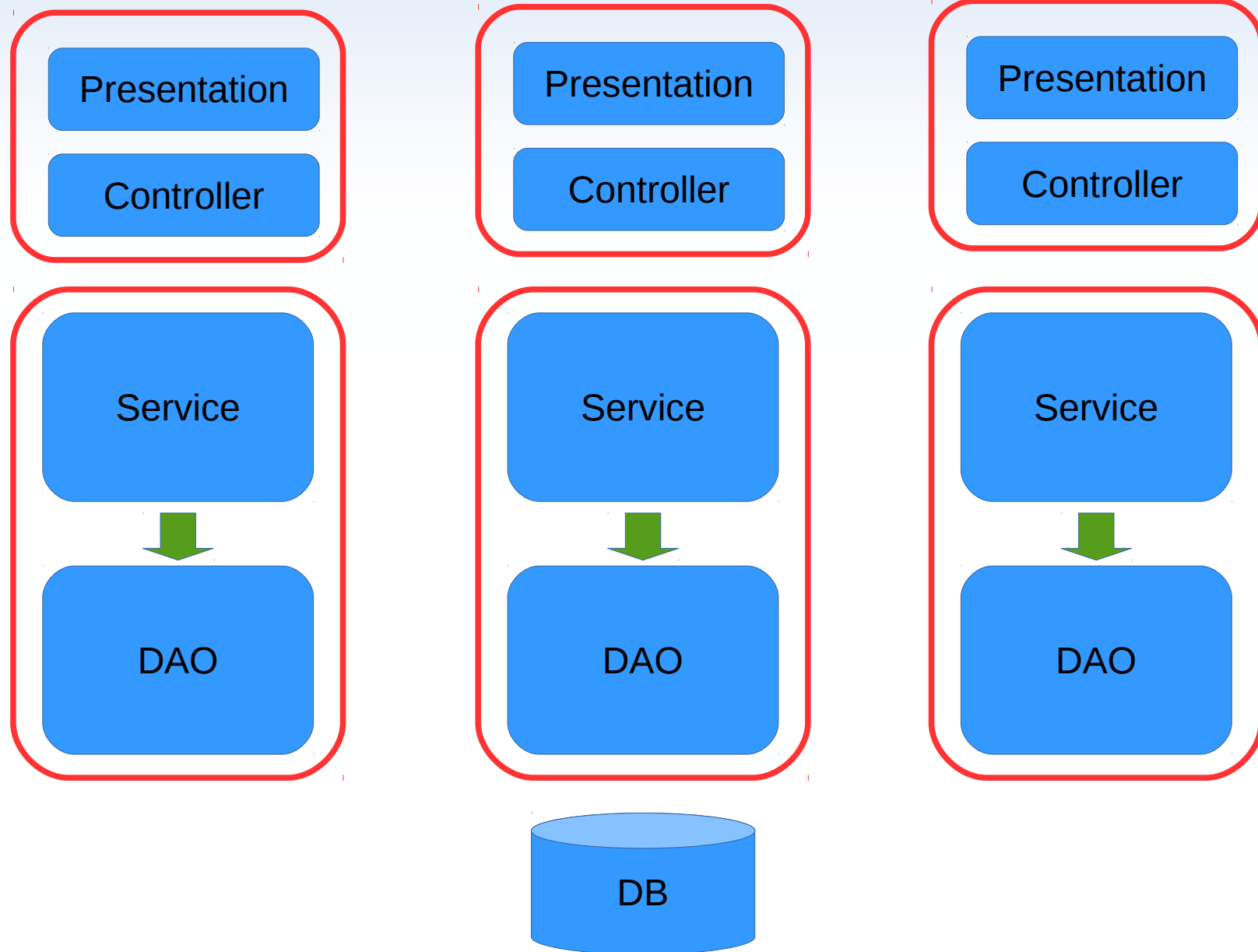
Katmanlı Mimari ve Modülerlik



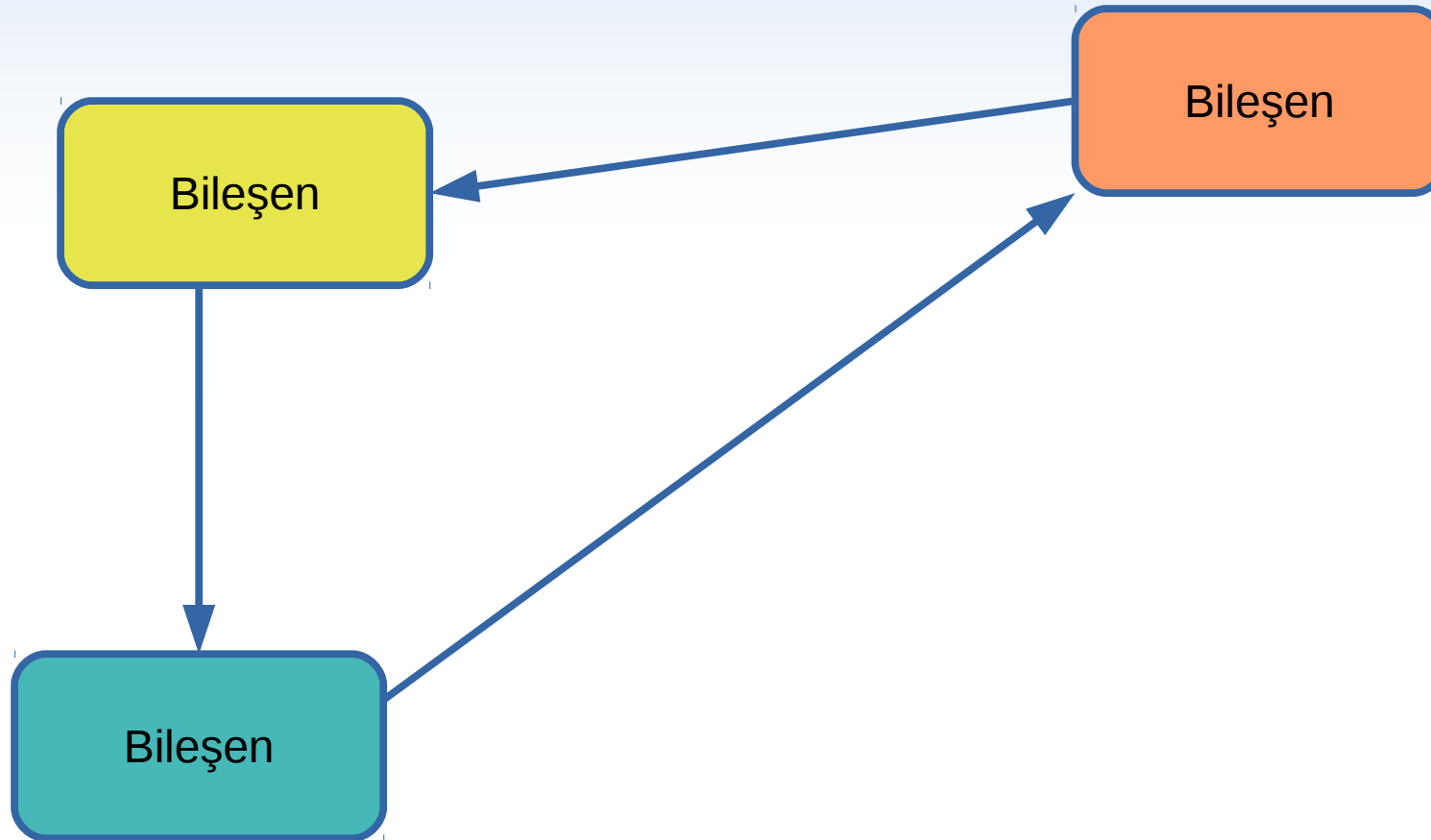
Katmanlı Mimari ve Modülerlik



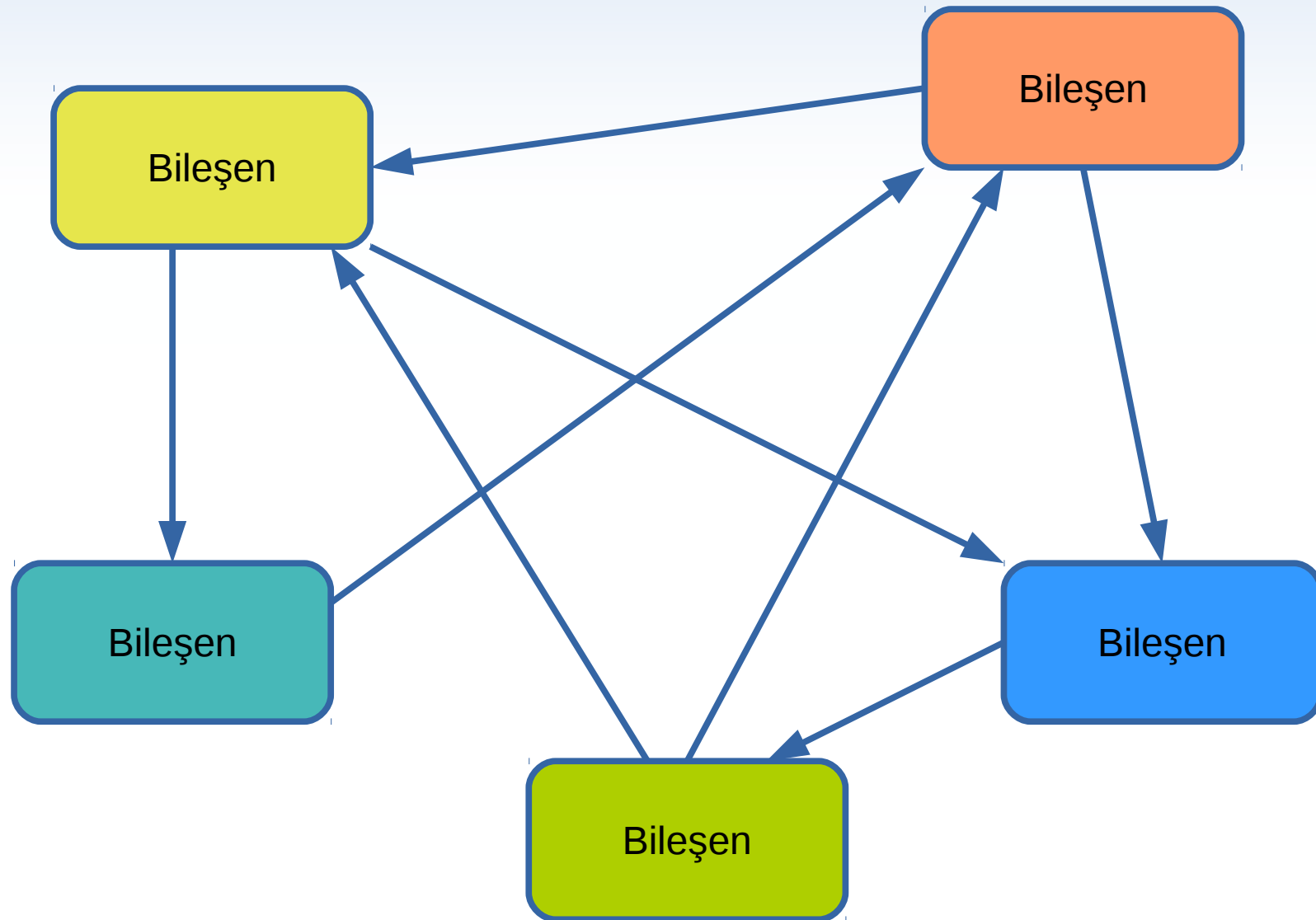
Katmanlı Mimari ve Modülerlik



Bileşenler Arasındaki Etkileşim



Bileşenler Arasındaki Etkileşim



MVC'nin Problemleri

- MVC örüntüsü, mimarisel olarak sistemi **işlevsel açıdan modülerize** etmeye yardımcı olmaktadır
- Ancak kullanıcı etkileşimlerinin fonksiyonel davranışa **nasıl dönüştürüleceği** ile ilgili **net bir yol gösterememektedir**
- Bileşenler arasındaki **iletişimi düzenleyememektedir** ve bileşenlerin birbirleri ile aralarındaki **bağımlılıkları da tam olarak ortadan kaldıramamaktadır**

Çözüm: MVP + Mediator

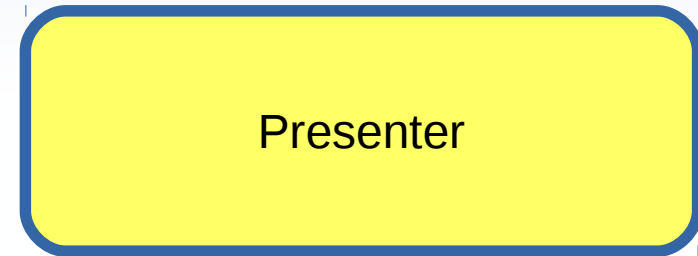
- MVC'nin bir varyasyonu olan **MVP**, kullanıcı arayüzünün gösterimi ve fonksiyonel davranışların birbirlerinden bağımsız biçimde ele alınabilmesini sağlamaktadır
- **Mediator** ise bileşenler arasındaki iletişimi düzenleyip, bağımlılıkları ortadan kaldırmaktadır

Model View Presenter



View

UI event'leri uygulamaya özel business event'lere dönüştürülür



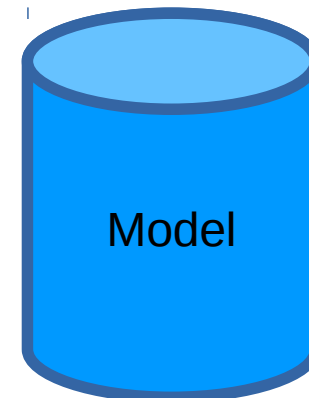
Presenter

UI üzerindeki değişiklikler Presenter tarafından yansıtılır



Presenter Model üzerinde değişiklik yapabilir

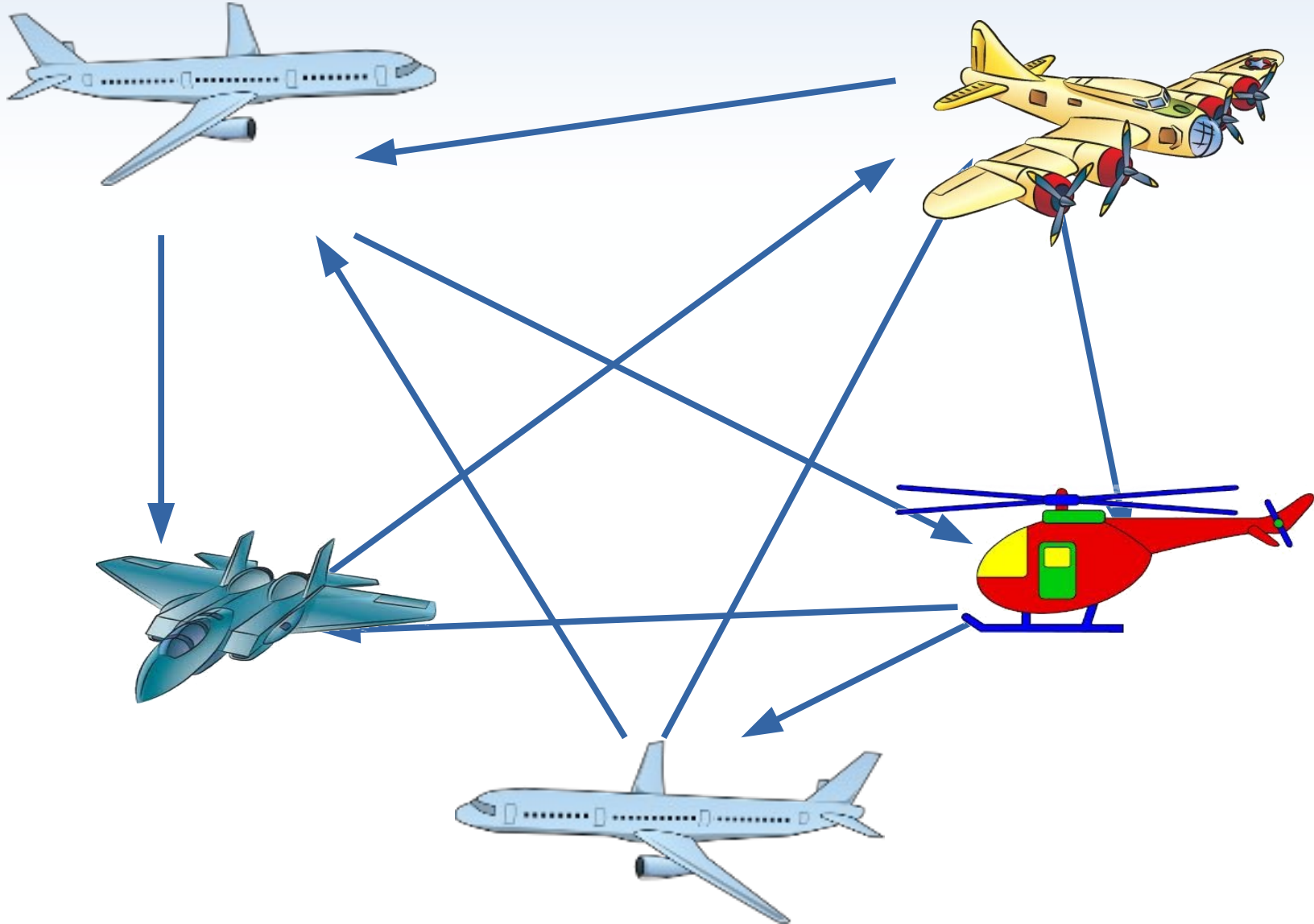
Model verisine erişebilir



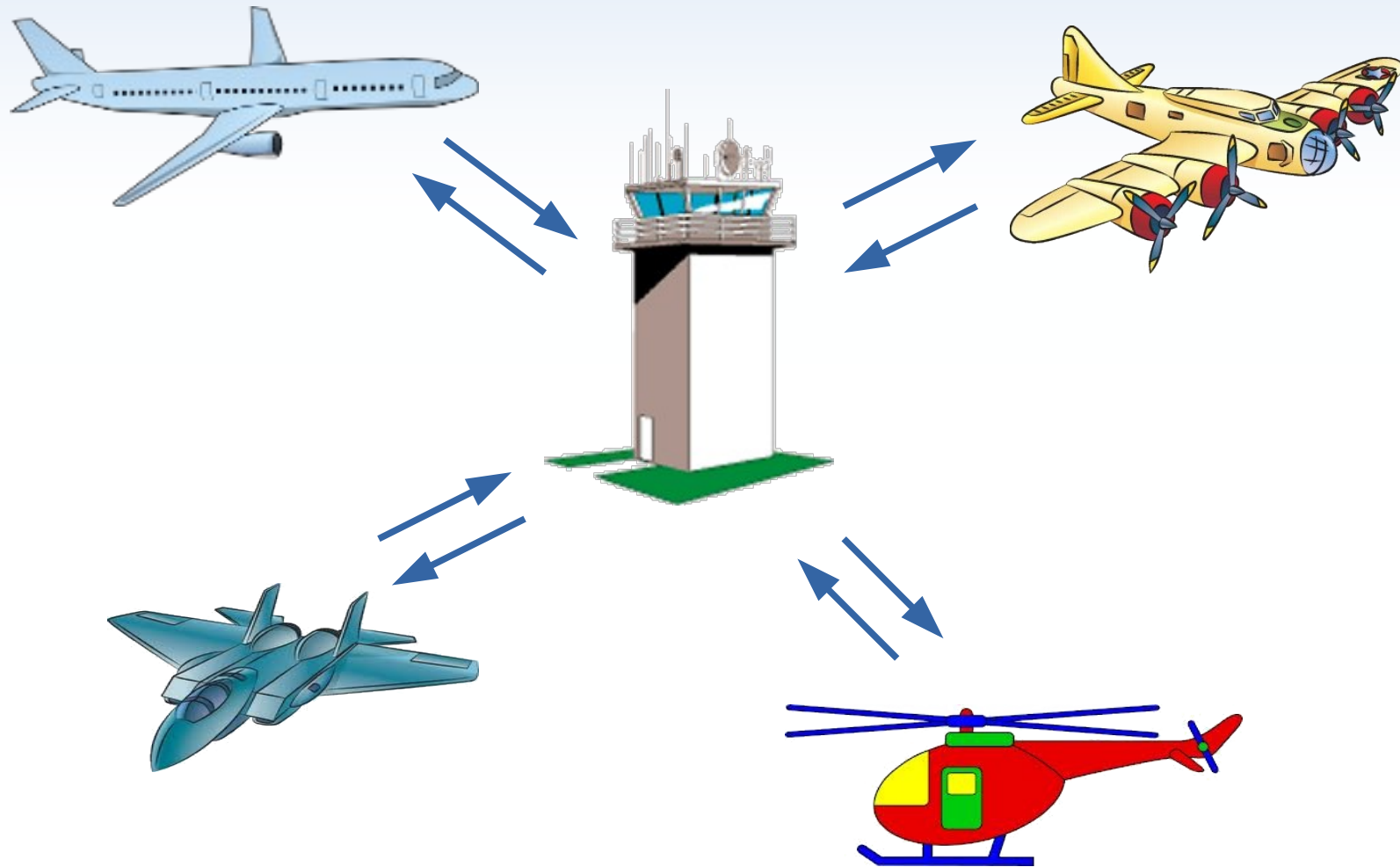
Model

Model üzerindeki değişiklikler Event'ler ile Presenter'a iletilir

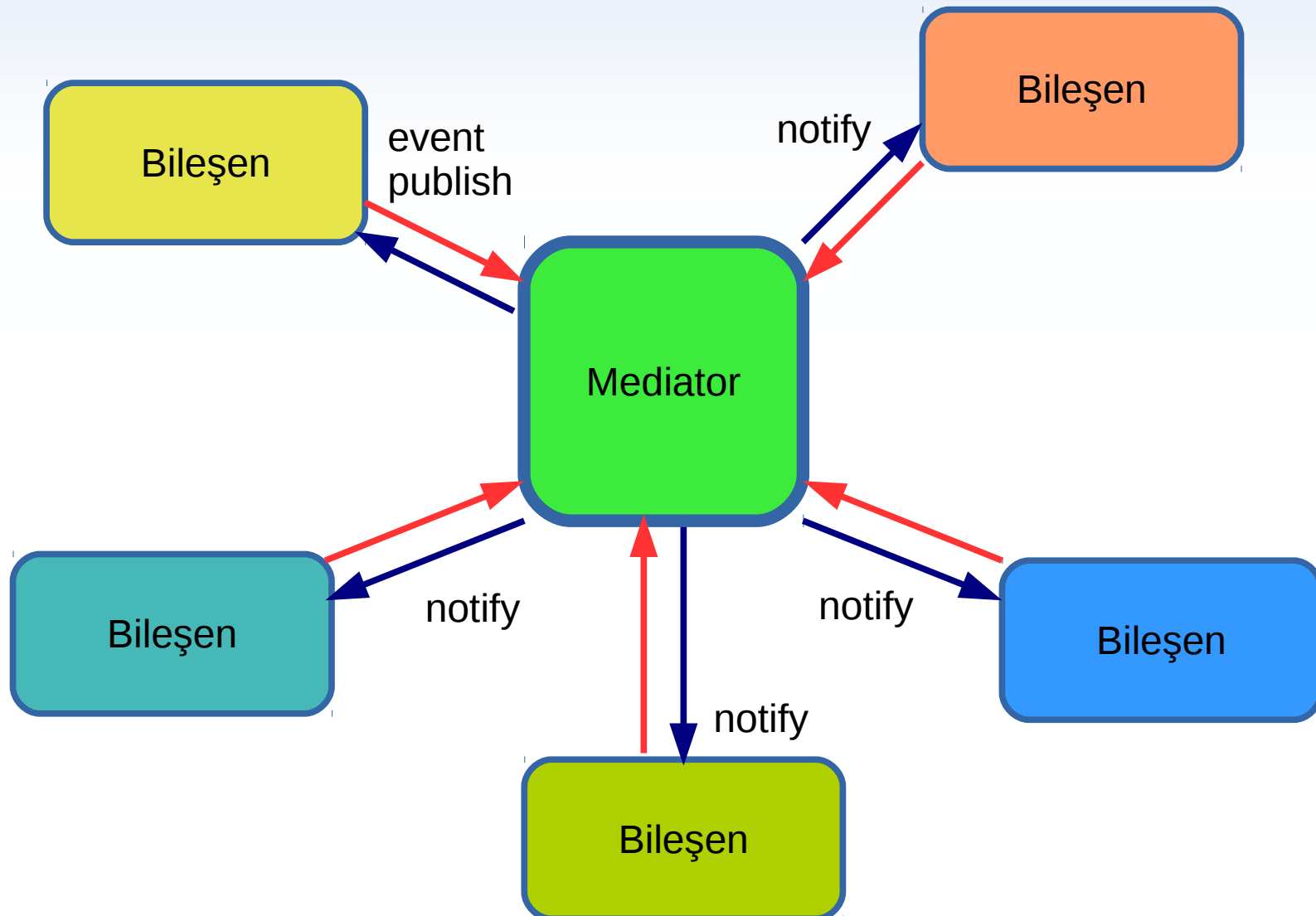
Bileşenler Arasındaki Etkileşim Kaosu



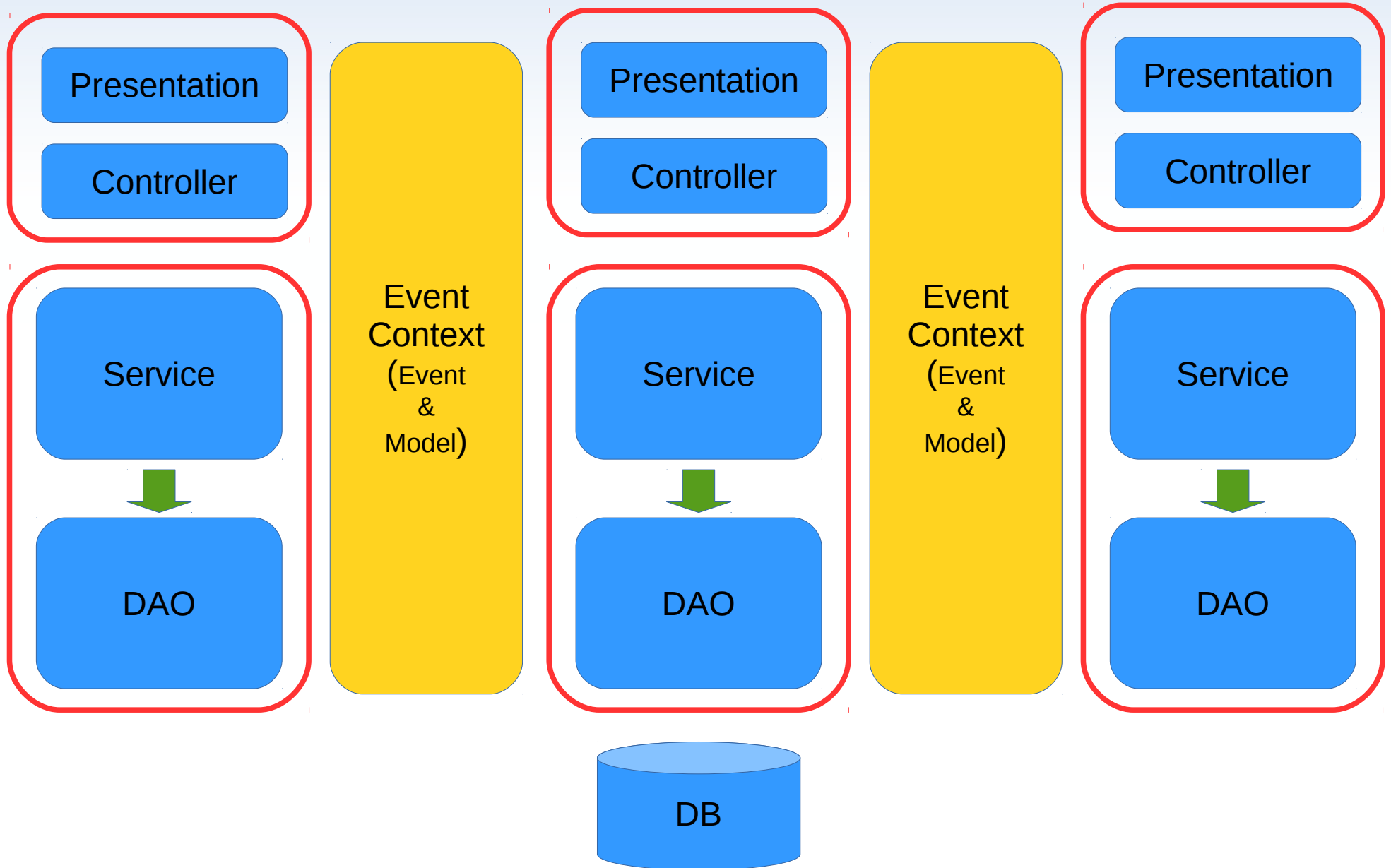
Mediator



Mediator Sonrası Bileşenler Arasındaki Etkileşim

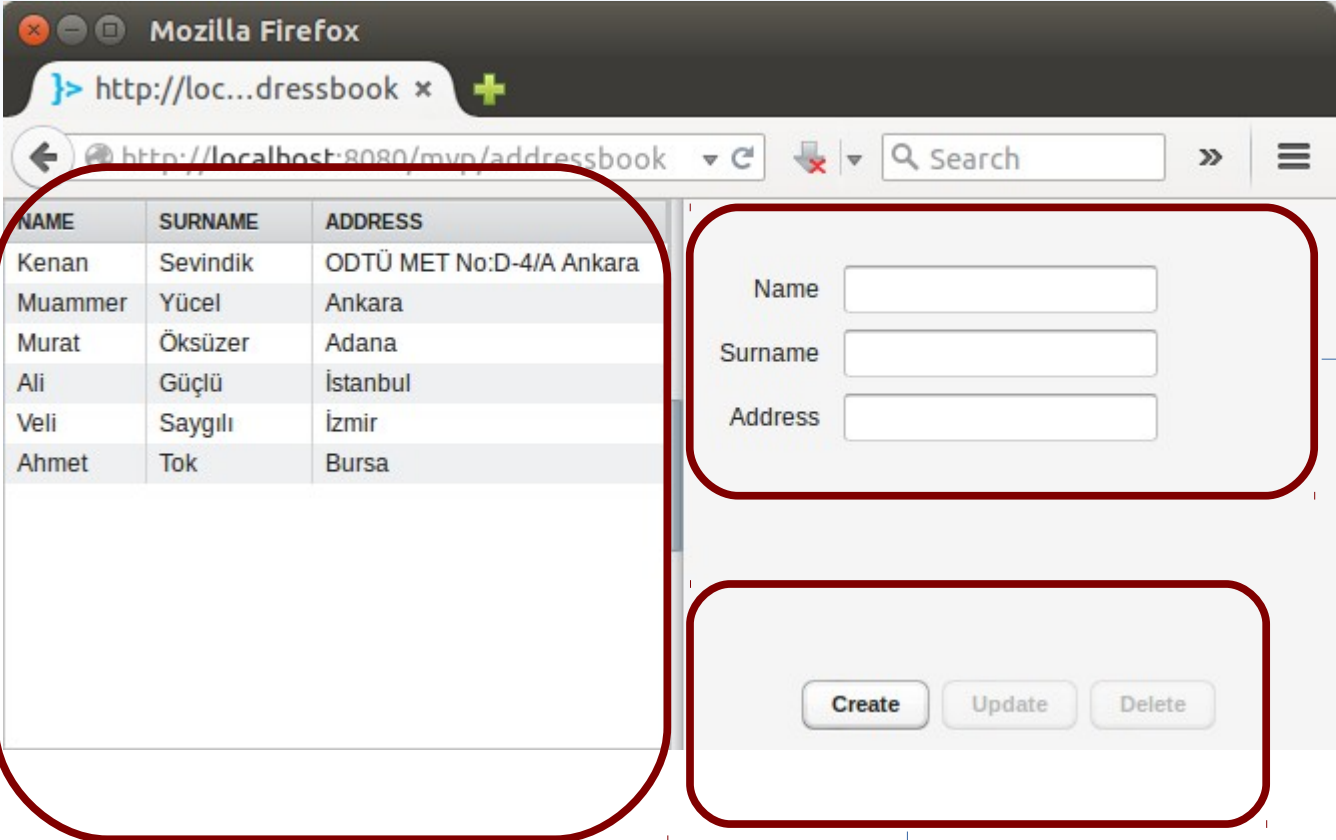


Mediator Sonrası Bileşenler Arasındaki Etkileşim



Örnek:

Adres Bilgileri Yönetim Ekranı



Address List View

NAME	SURNAME	ADDRESS
Kenan	Sevindik	ODTÜ MET No:D-4/A Ankara
Muammer	Yücel	Ankara
Murat	Öksüzer	Adana
Ali	Güçlü	İstanbul
Veli	Saygılı	İzmir
Ahmet	Tok	Bursa

Address Detail View

Create Update Delete

Address ToolBar View

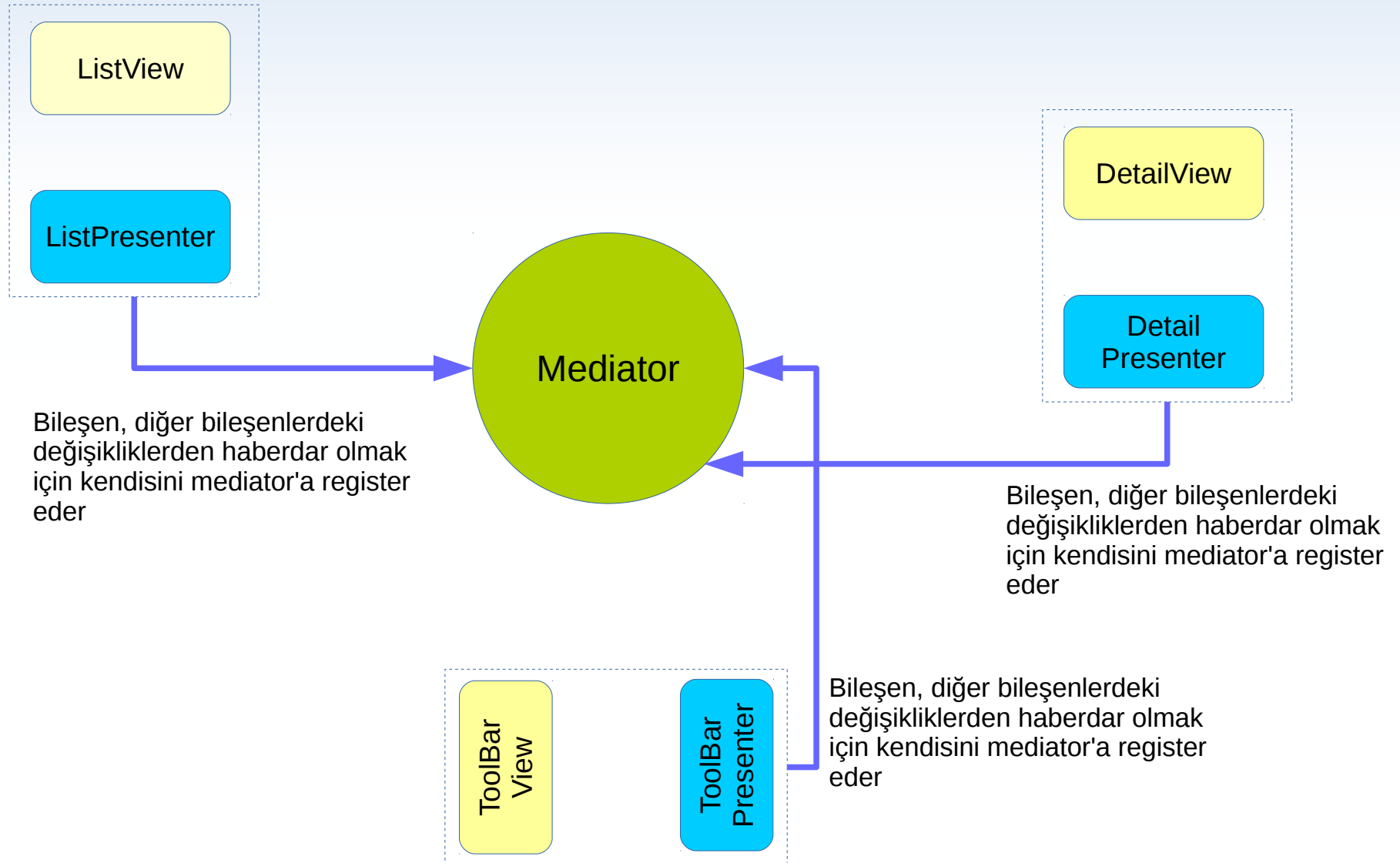
Mediator

```
public class Mediator {  
  
    private Collection<Presenter> listeners = new  
        ArrayList<Presenter>();  
  
    public void addListener(Presenter listener) {  
        listeners.add(listener);  
    }  
  
    public void removeListener(Presenter listener) {  
        listeners.remove(listener);  
    }  
  
    public void publish(BusinessEvent event) {  
        for(Presenter listener:listeners) {  
            listener.handle(event);  
        }  
    }  
}
```

Presenter

```
public interface Presenter {  
  
    public void handle(BusinessEvent event);  
  
}
```


Adım 1: Mediator Registration



Address List Presenter

```
public class AddressListPresenter implements Presenter {  
  
    private AddressListView view;  
    private Mediator mediator;  
  
    public AddressListPresenter(AddressListView view,  
        Mediator mediator) {  
        this.view = view;  
        this.mediator = mediator;  
        mediator.addListener(this);  
    }  
  
    @Override  
    public void handle(BusinessEvent event) {  
        ...  
    }  
}
```

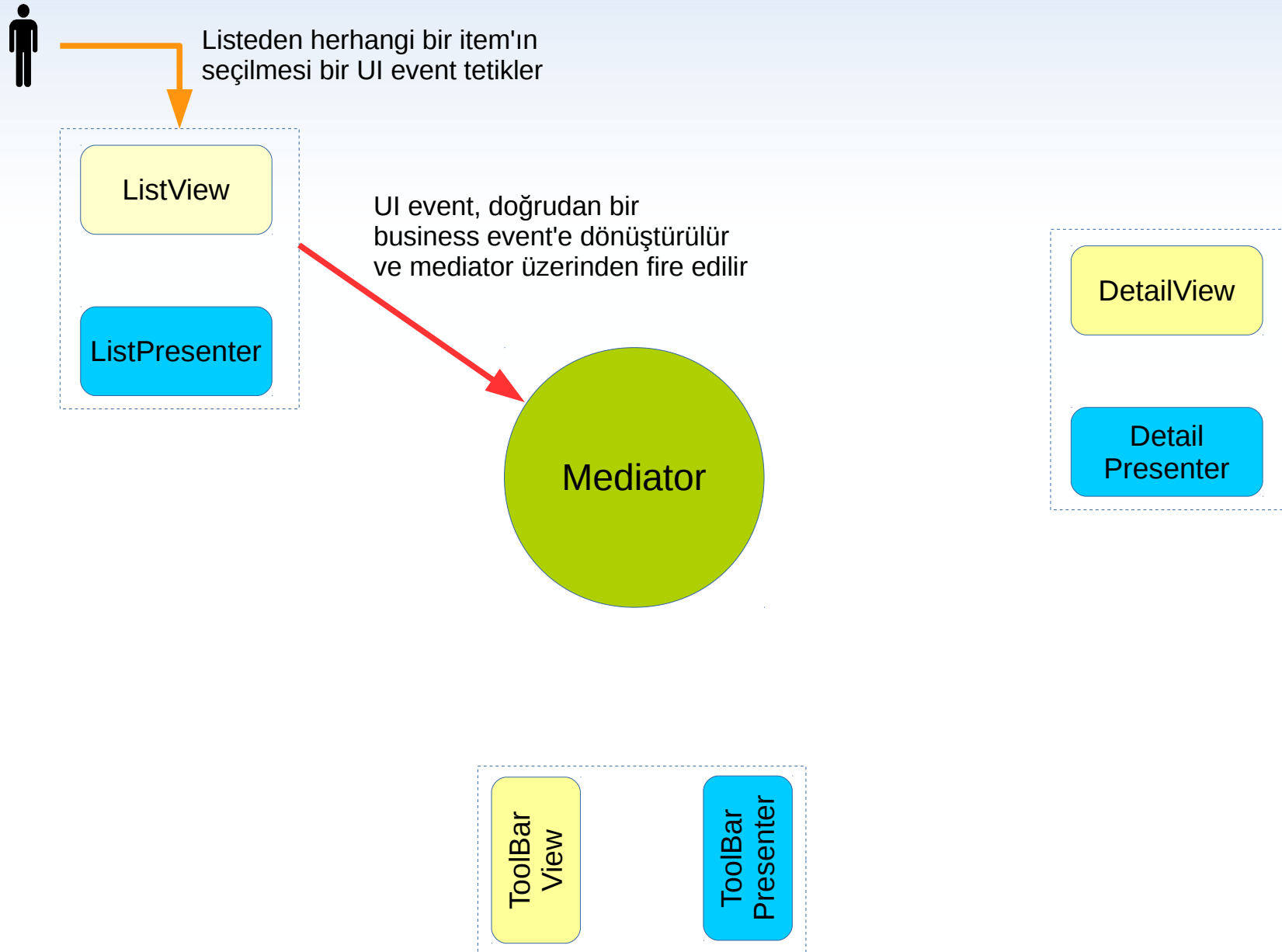
Address Detail Presenter

```
public class AddressDetailPresenter implements Presenter {  
  
    private AddressDetailView view;  
    private Mediator mediator;  
  
    public AddressDetailPresenter(AddressDetailView view,  
        Mediator mediator) {  
        this.view = view;  
        this.mediator = mediator;  
        mediator.addListener(this);  
    }  
  
    @Override  
    public void handle(BusinessEvent event) {  
        ...  
    }  
}
```

Address ToolBar Presenter

```
public class AddressToolBarPresenter implements Presenter {  
  
    private AddressToolBarView view;  
    private Mediator mediator;  
  
    public AddressToolBarPresenter(AddressToolBarView view,  
        Mediator mediator) {  
        this.view = view;  
        this.mediator = mediator;  
        mediator.addListener(this);  
    }  
  
    @Override  
    public void handle(BusinessEvent event) {  
        ...  
    }  
}
```

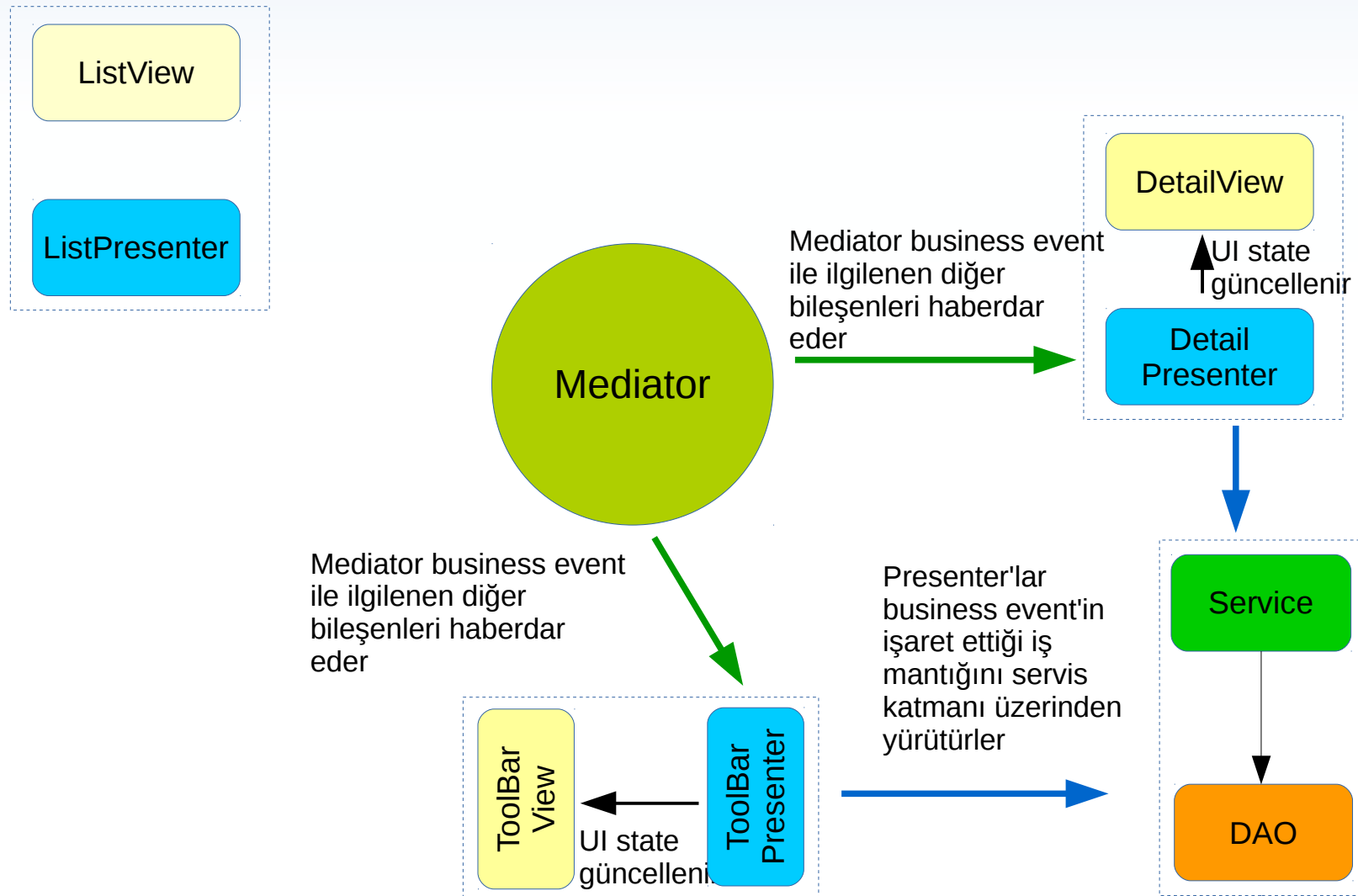
Adım 2: UI Interaction (Item Select)



Address List View

```
public class AddressListView implements ValueChangeListener {  
  
    public AddressListView(Mediator mediator) {  
        this.mediator = mediator;  
    }  
  
    @Override  
    public void valueChange(ValueChangeEvent event) {  
        Address address = (Address) table.getValue();  
  
        AddressSelectedEvent selectedEvent = new  
            AddressSelectedEvent(address);  
        mediator.publish(selectedEvent);  
    }  
  
    ...  
}
```


Adım 3: Event Notification (Address Selected)



Address Detail Presenter

```
public class AddressDetailPresenter implements Presenter {  
  
    @Override  
    public void handle(BusinessEvent event) {  
        if(event instanceof AddressSelectedEvent) {  
            AddressSelectedEvent selectedEvent =  
                (AddressSelectedEvent)event;  
            Address address =  
                selectedEvent.getSelectedAddress();  
            view.displayAddress(address);  
        }  
    }  
  
    ...  
}
```

Address ToolBar Presenter

```
public class AddressToolBarPresenter implements Presenter {  
  
    @Override  
    public void handle(BusinessEvent event) {  
        if(event instanceof AddressSelectedEvent) {  
            AddressSelectedEvent selectedEvent =  
                (AddressSelectedEvent)event;  
            Address address =  
                selectedEvent.getSelectedAddress();  
  
            view.switchToUpdateMode();  
            view.setAddress(address);  
  
        }  
    }  
  
    ...  
}
```

Address Selected

Mozilla Firefox

http://loc...dressbook x

http://localhost:8080/mvp/addressbook

NAME	SURNAME	ADDRESS
Kenan	Sevindik	ODTÜ MET No:D-4/A Ankara
Muammer	Yücel	Ankara
Murat	Öksüzer	Adana
Ali	Güçlü	İstanbul
Veli	Saygılı	İzmir
Ahmet	Tok	Bursa

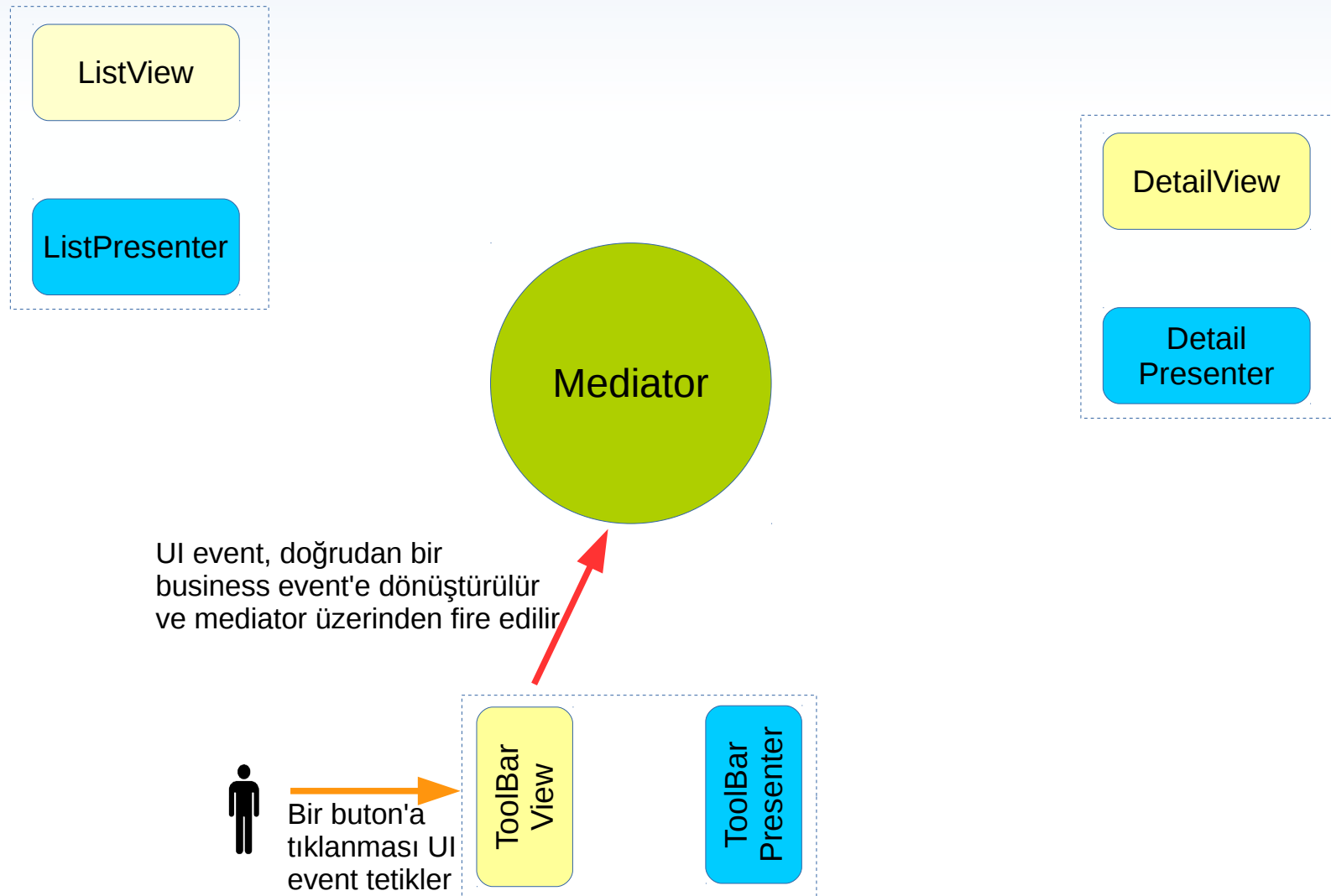
Name

Surname

Address

Create Update Delete

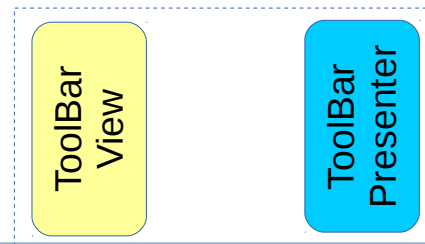
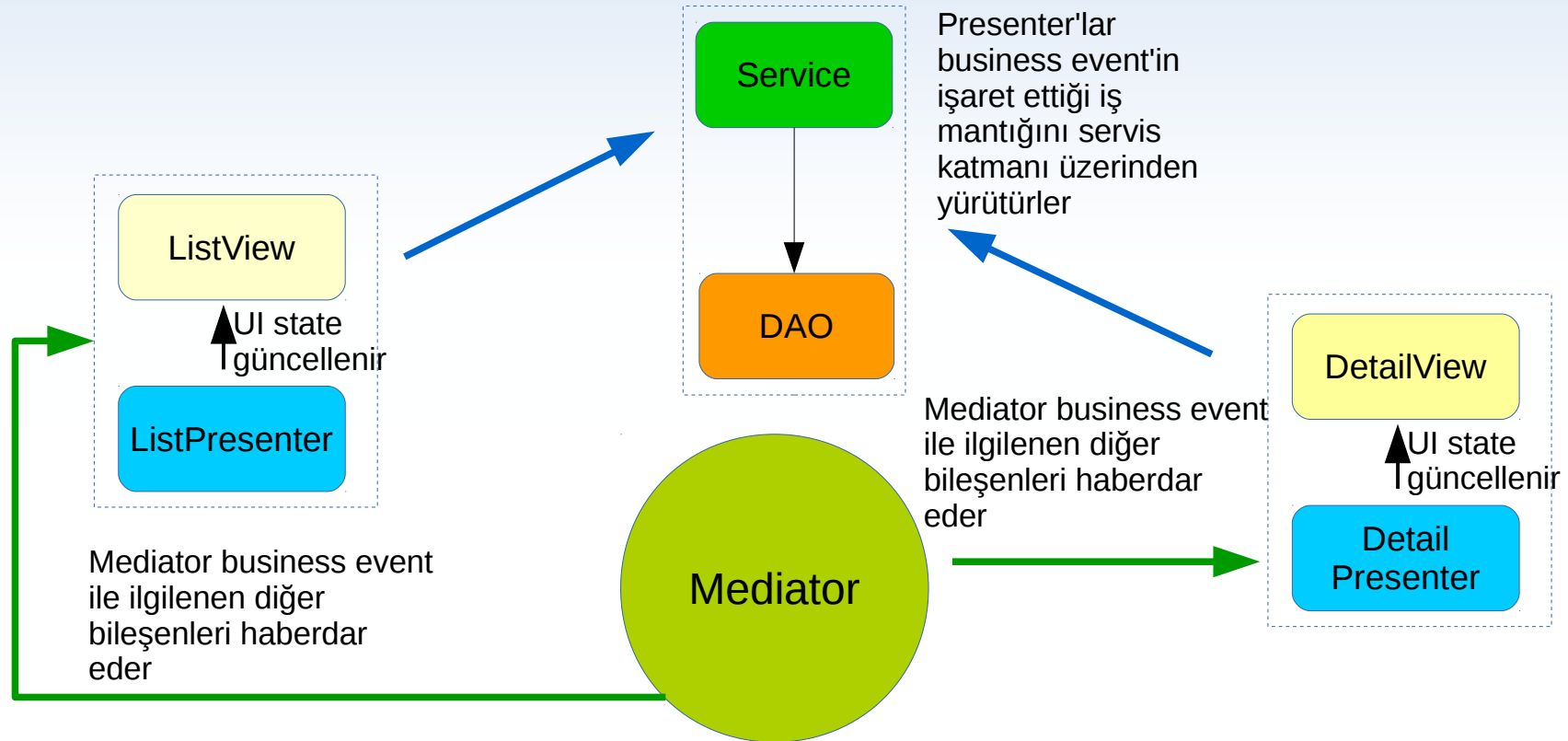
Adım 2: UI Interaction (Update Button Click)



Address ToolBar View

```
public class AddressToolBarView implements ClickListener {  
  
    public AddressToolBarView(Mediator mediator) {  
        this.mediator = mediator;  
    }  
  
    @Override  
    public void buttonClick(ClickEvent event) {  
        if(event.getButton() == updateButton) {  
            AddressUpdateEvent updateEvent =  
                new AddressUpdateEvent(address);  
            mediator.publish(updateEvent);  
        }  
    }  
  
    ...  
}
```

Adım 3:Event Notification (Address Update)



Address List Presenter

```
public class AddressListPresenter implements Presenter {  
  
    @Override  
    public void handle(BusinessEvent event) {  
  
        if(event instanceof AddressUpdateEvent) {  
            AddressUpdateEvent updateEvent =  
                (AddressUpdateEvent)event;  
  
            Address address = updateEvent.getAddress();  
  
            view.reloadAddress(address);  
  
        }  
    }  
  
    ...  
}
```


Address ToolBar Presenter

```
public class AddressToolBarPresenter implements Presenter {

    @Override
    public void handle(BusinessEvent event) {
        if(event instanceof AddressSelectedEvent) {
            AddressSelectedEvent selectedEvent =
                (AddressSelectedEvent)event;
            Address address =
                selectedEvent.getSelectedAddress();

            view.switchToUpdateMode();
            view.setAddress(address);

        } else if(event instanceof AddressUpdateEvent) {

            view.switchToSelectionMode();

        }
        ...
    }
}
```

Address Updated

Mozilla Firefox

http://loc...dressbook x

http://localhost:8080/mvp/addressbook

NAME	SURNAME	ADDRESS
Kenan	Sevindik	ODTÜ MET No:D-4/A Ankara
Muammer	Yücel	Ankara
Murat	Öksüzer	Adana
Veli	Saygılı	İzmir
Ahmet	Tok	Bursa
Ali	Tatlı	İstanbul

Name

Surname

Address

Create Update Delete

Soru & Cevap

İletişim

- Harezmi Bilişim Çözümleri A.Ş.
- <http://www.harezmi.com.tr>
- info@harezmi.com.tr