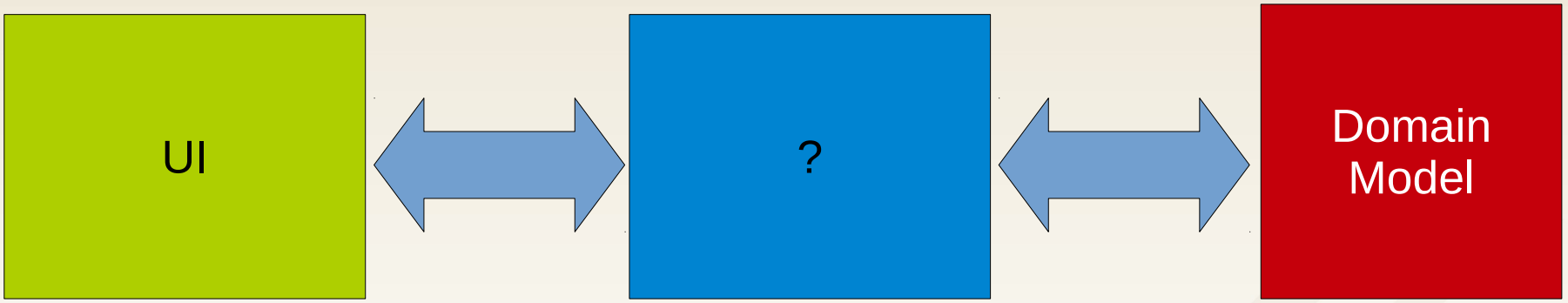
JAVA Eğitimleri

Vaadin UI – Model Binding

harezmi bilişim çözümleri

UI – Model Binding



- Vaadin, UI bileşenleri ile domain model'i birbirine bağlamak için kapsamlı bir **data binding model**i sunar
- Bu model **3 temel bileşen**den oluşur:
  - Property, Item, Container

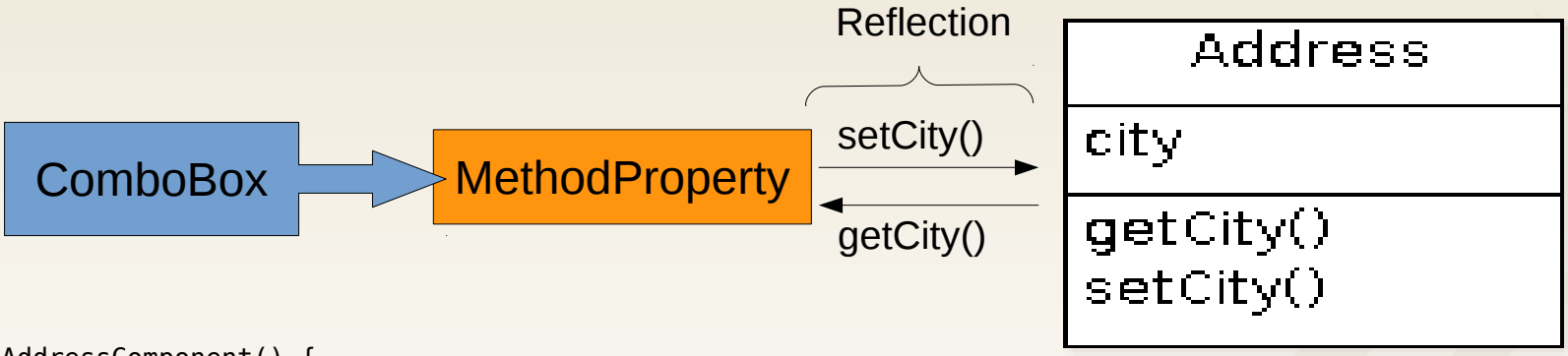
UI – Model Binding

- Property
  - Bir domain nesnesinin attribute'larını ekran bileşenlerine bind etmek için kullanılır
- Item
  - Bir domain nesnesini wrap eder, bu domain nesnesinin attribute'larının bind edildiği Property'leri gruplar
- Container
  - Domain nesnelerini wrap eden Item nesnelerini gruplar

Property

- Property'ler **atomik veri alanları** olarak düşünülebilirler
- Nesnenin **bir alan**ına karşılık gelirler
- Property'ler üzerine herhangi bir veri **setValue()** ile yazılıp **getValue()** ile okunabilir

MethodProperty




```
public AddressComponent() {

      ...

      ComboBox cmb = new ComboBox("City", petClinicService.getCities());
      cmb.setPropertyDataSource(
          new MethodProperty<String>(this.address,"city"));

      TextField telephone = new TextField("Telephone");
      telephone.setPropertyDataSource(new
                        MethodProperty<String>(this.address,"telephone"));

      OptionGroup phoneType = new OptionGroup("Phone Type",
            Arrays.asList(PhoneType.values()));
      phoneType.setPropertyDataSource(new MethodProperty<PhoneType>(this.address,"phoneType"));

      ...
}
```

ObjectProperty

- Herhangi bir **nesneyi wrap eden** Property implemantasyonudur
- getValue() metodu **wrap edilen nesne**yi döner
- setValue(..) metodu ise **verilen nesne**yi property değeri olarak set eder
- setValue() sırasında herhangi bir **conversion gerçekleşmez**
- Genellikle Integer, Date, String gibi değerleri UI bileşenine bind etmek için kullanılır

Property Change

- Property içerisindeki veri güncellendiğinde **ValueChangeEvent** fırlatılır
- Bu event Property.**ValueChangeListener** tarafından yakalanır

Converter

- Field bileşenlerine **girilen değerleri** model nesnelerine **dönüştüren nesneler**dir
- Presentasyon tipi (genellikle String) ile model tipi arasında **karşılıklı dönüşüm** yaparlar
- Field düzeyinde **Converter set edilebilir**
- Vaadin standart tipler için **built-in Converter nesneleri** sunmaktadır
- İstenirse **Converter arayüzü** üzerinden **custom Converter** yazılabilir

Converter

```
ObjectProperty<Integer> property = new ObjectProperty<Integer>(42);

TextField field = new TextField("Name");

field.setPropertyDataSource(property);

field.setConverter(new StringToIntegerConverter());
```

- **PropertyDataSource** set edilirken de **ConverterFactory**'den bulunan model tipine uygun bir Converter implicit biçimde set edilmektedir

Built-in Converter Sınıfları

Converter	Gösterim	Model
StringToIntegerConverter	String	Integer
StringToDoubleConverter	String	Double
StringToNumberConverter	String	Number
StringToBooleanConverter	String	Boolean
StringToDateConverter	String	Date
DateToLongConverter	Date	Long

ConverterFactory

- **ConverterFactory** arayüzü üzerinden Integer, Date, Long gibi **standart tipler için** Converter nesneleri sunulur
- **VaadinSession** üzerinden yönetilir
- **DefaultConverterFactory** implemantasyonudur
- İstenirse custom Converter nesneleri de dönülecek biçimde **override edilebilir**

Validator

- **Field** bileşenlerine girilen değerler **Validator** nesneleri tarafından validate edilebilirler
- Eğer bileşen geçersiz bir değer içeriyor ise validasyon sonucu bileşenin hata indikatörü ve hata mesajı görüntülenecektir

```
TextField fieldFirstName = new TextField("First Name");

fieldFirstName.addValidator(
    new StringLengthValidator("Name must be of min length 3",3,null,false));

...

fieldFirstName.validate();
```

Otomatik Validasyon

- Eğer Field bileşeninin **validationVisible** property'si aktif ise (**default true**) bileşenin değeri değiştiği vakit validasyon otomatik olarak gerçekleştirilecektir
- Bileşen immediate modda ise validasyon **unfocus olunduğunda** yine otomatik olarak gerçekleşecektir
- Immediate aktif iken hem ilgili field'ın hem diğer field'ların değerleri validate edilir

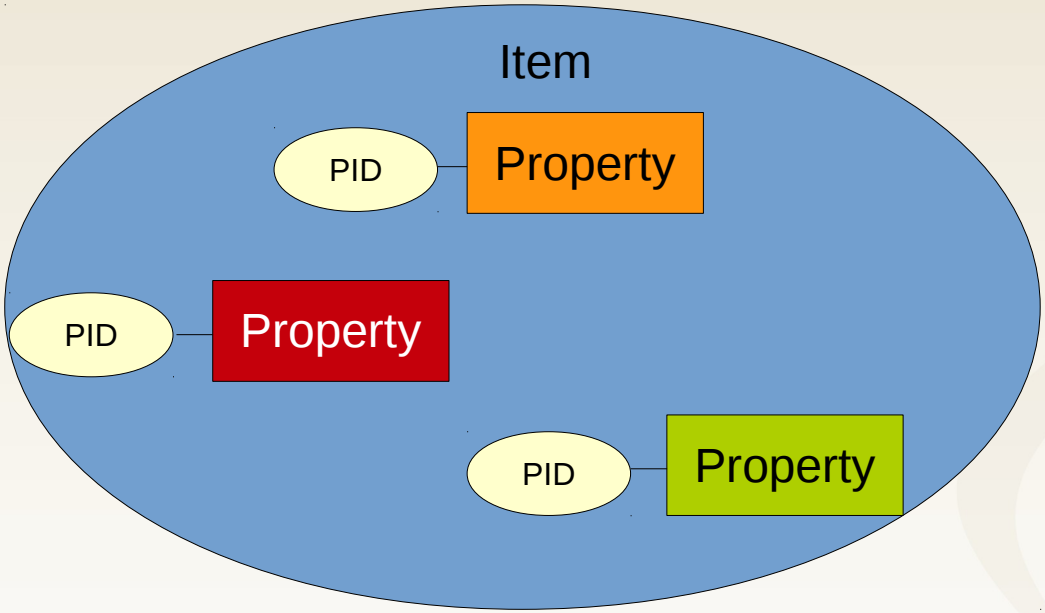
Explicit Validasyon

- Validator nesneleri ayrıca Field bileşeninin **validate()** ve **commit()** metotları çağrılarak da çalıştırılabilirler

Built-in Validator Sınıfları

- NullValidator
- StringLengthValidator
- IntegerRangeValidator
- DoubleRangeValidator
- DateRangeValidator
- EmailValidator
- RegexpValidator
- CompositeValidator
- BeanValidator

Item

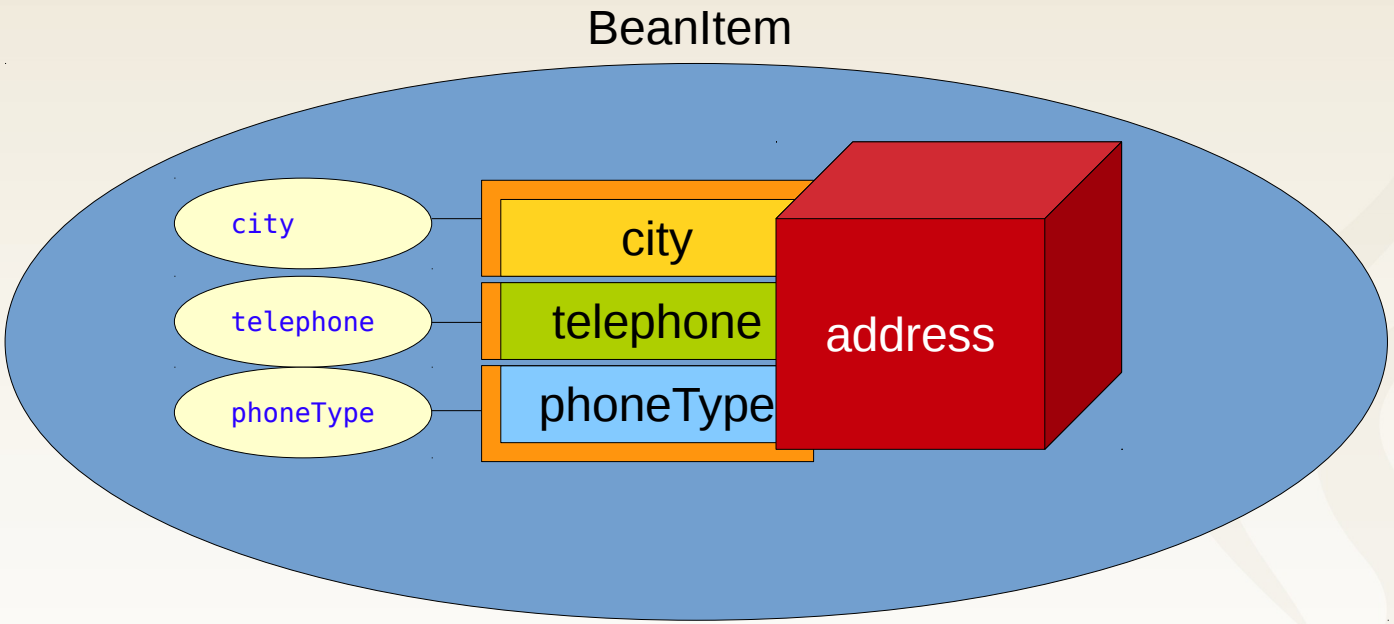


- Item'lar **bir grup Property**'nin bir araya gelmesiyle oluşurlar
- Item'lar nesneye yönelik programlamada **nesnelere karşılık** gelir

Item

- Aynı zamanda veri tabanında bir kayıt ya da ekrandaki bir form gibi de düşünülebilir
- Item içerisindeki property'lere **PID** üzerinden **getItemProperty()** ile erişilir

Bean Item




```
public AddressComponent() {

      BeanItem<Address> item = new BeanItem<Address>(address);

      Property cityProperty = item.getItemProperty("city");

      ...
}
```

Address bean'inin tüm alanları property olarak sunulur

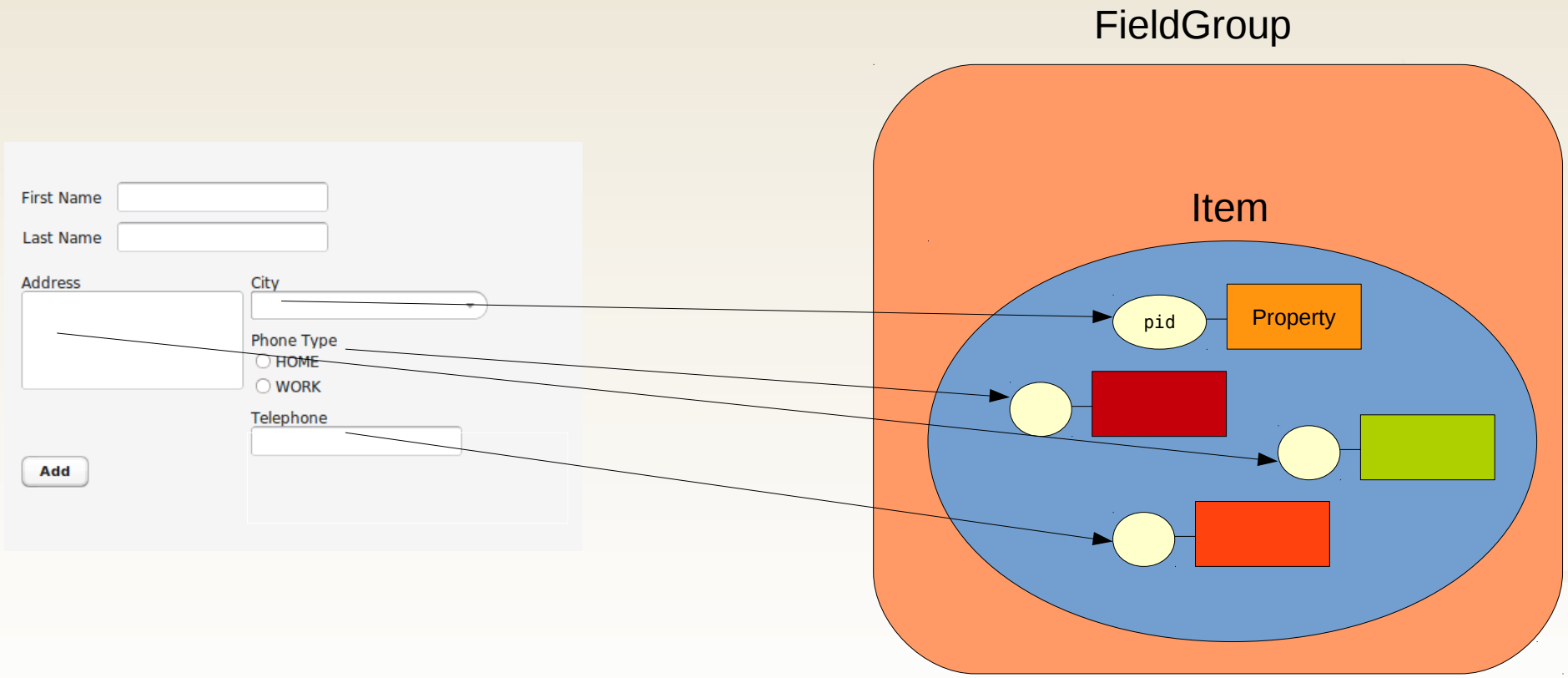
Item Change

- Item içerisinde yer alan property grubunda bir değişiklik olursa **PropertySetChangeEvent** event'i fırlatılır
- Bu event **PropertySetChangeListener** ile yakalanabilir

FieldGroup

- UI field bileşenleri ile veri arasındaki **bind ve validasyon işlemleri**ni kolaylaştırır
- Vaadin 7 öncesindeki Form elemanına benzer, ancak FieldGroup bir **UI bileşeni değil**dir
- Layout bileşenlerine eklenemez
- UI **field bileşenlerinin ayrıca oluşturulmaları** ve layout'a eklenmeleri gerekir

FieldGroup


First Name
Last Name
Address
City
Phone Type
HOME
WORK
Telephone
Add
FieldGroup
Item
pid
Property

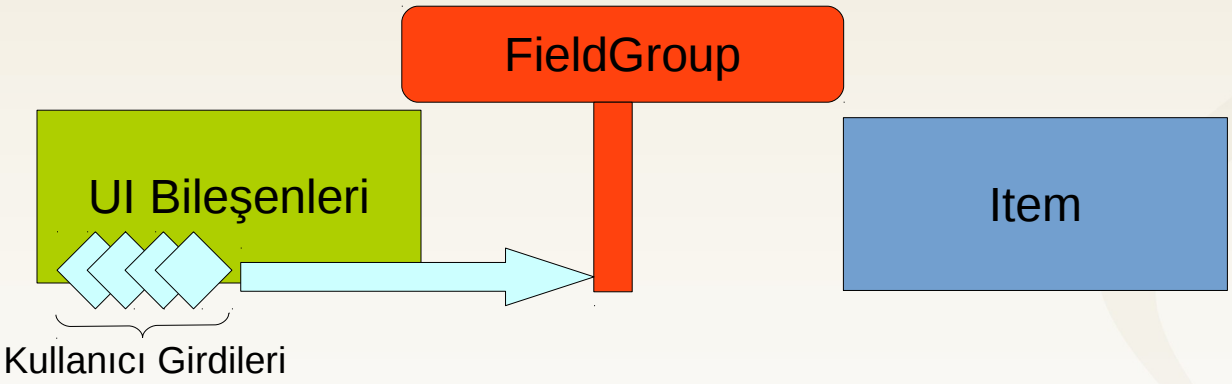
FieldGroup ve Bind İşlemi

```
private Address address;

@PropertyId("address")
private TextArea addressField;

@PropertyId("phone")
private TextField telephone;

private ComboBox city;
private OptionGroup phoneType;

public AddressComponent() {

    addressField = new TextArea("Address");
    city = new ComboBox("City", petClinicService.getCities());
    telephone = new TextField("Telephone");
    phoneType = new OptionGroup("Phone Type", Arrays.asList(PhoneType.values()));

    fieldGroup = new FieldGroup(new BeanItem<Address>(address));
    fieldGroup.bindMemberFields(this);

    FormLayout layout = new FormLayout();
    layout.addComponent(addressField);
    ...
    setCompositionRoot(layout);
}
```

Member Fields

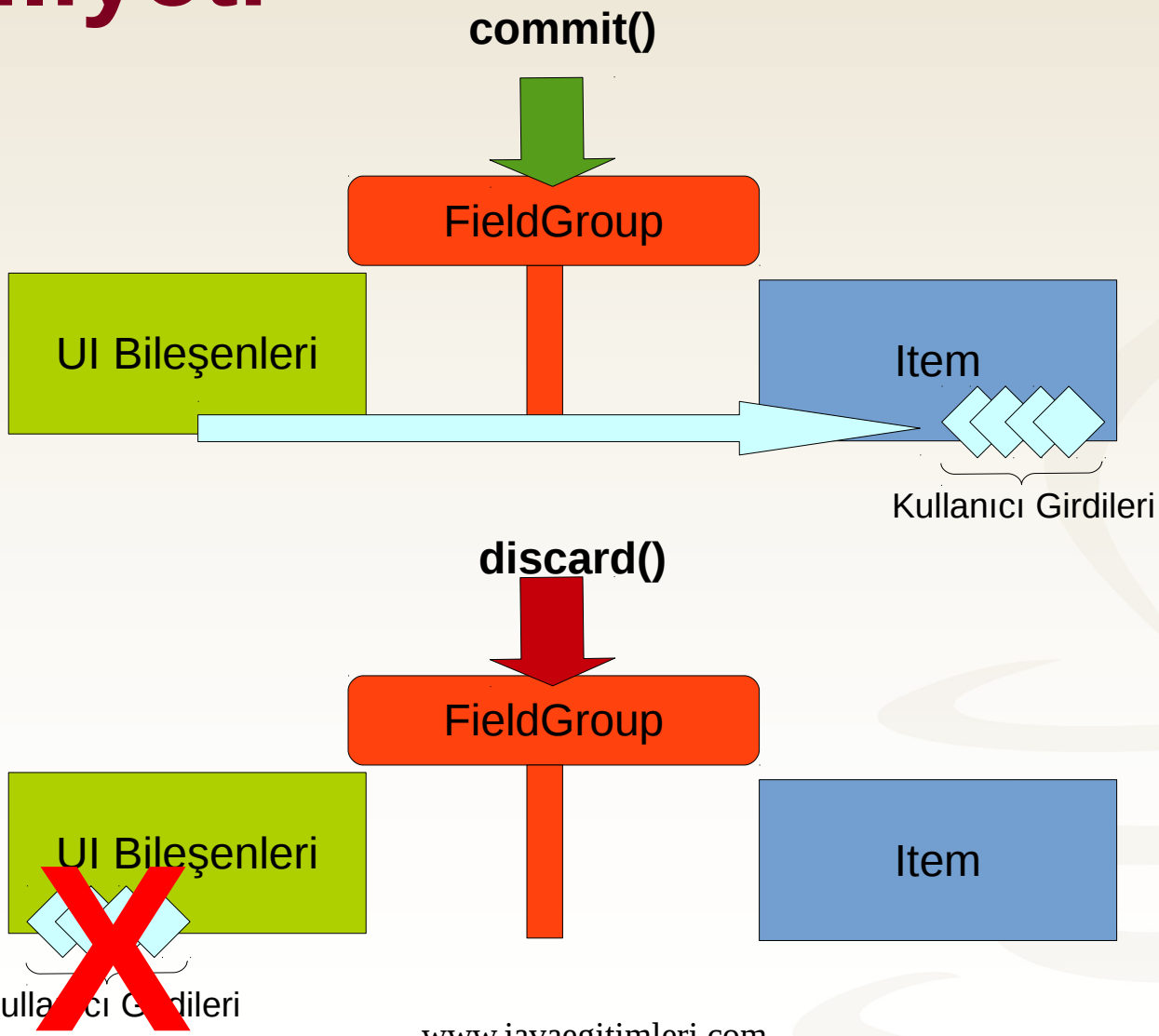
Property ID'ler ayrıca belirtilebilir.
Belirtilmezse değişken adı kabul edilir.

FieldGroup Buffer Kabiliyeti



UI bileşenlerinde toplanan kullanıcı girdileri **commit** ile arka taraftaki Item'a buradan da domain nesnesine topluca yansıtılabilir, yada **rollback** ile bu girdiler toptan gözardı edilebilir

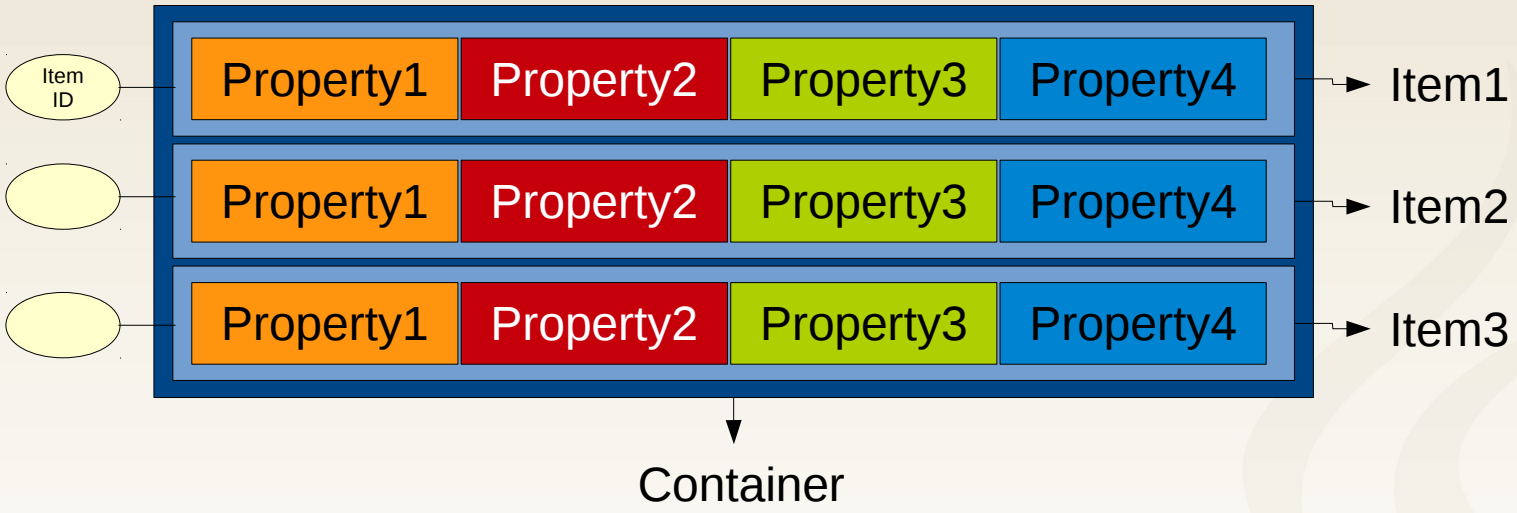
FieldGroup Buffer Kabiliyeti



FieldGroup ve Validasyon

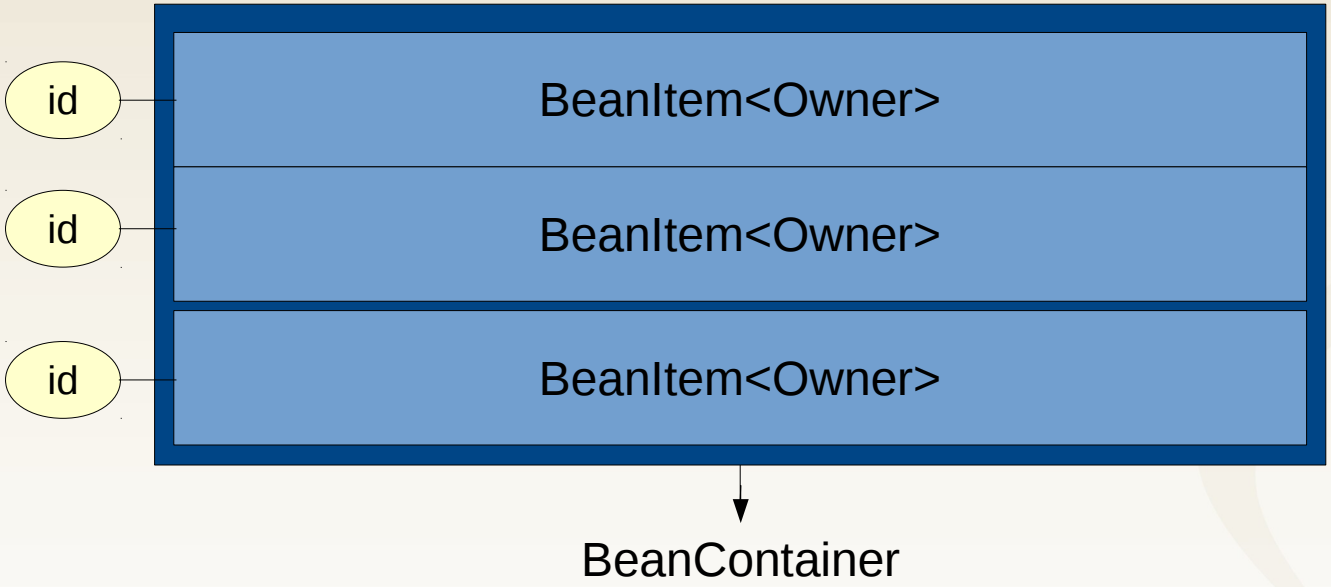
- FieldGroup.**commit()** metodunun çağrılması FieldGroup tarafından yönetilen **Field bileşenlerinin validasyonu**nu tetikler
- Ayrıca **BeanFieldGroup** sınıfı **JSR-303** Bean Validation API'si ile çalışabilmektedir
- BeanFieldGroup tarafından yönetilen Field'ların validasyonu **BeanValidator** kullanılarak otomatik olarak gerçekleştirir
- Classpath'de JSR-303 kütüphanesi olmalıdır

Container



- Container, **item'lardan oluşan** bir gruptur
- Collection içerisindeki item'lara **IID** (item identifier) üzerinden erişilebilir
- Genellikle Table, Tree, ComboBox, ListSelect gibi bileşenlerde kullanılırlar

BeanContainer




```
Table ownerTable = new Table("Owners");

BeanContainer<Integer, Owner> ownerContainer =
        new BeanContainer<Integer, Owner>(Owner.class);

ownerContainer.setBeanIdProperty("id");

ownerContainer.addBean(owner);

ownerTable.setContainerDataSource(ownerContainer);
```

Tabloya veri ekler

Owners

ADDRESS	FIRSTNAME	ID	LASTNAME	PETS
ODTÜ Ankara	Kenan	3	Sevindik	[]

BeanContainer ve Nested Property'ler

address property'sini çıkarıp yerine nested property'leri ekliyoruz

```
BeanContainer<Integer, Owner> ownerContainer =
            new BeanContainer<Integer, Owner>(Owner.class);

ownerContainer.setBeanIdProperty("id");

ownerContainer.removeContainerProperty("address");

ownerContainer.addNestedContainerProperty("address.address");
ownerContainer.addNestedContainerProperty("address.city");
ownerContainer.addNestedContainerProperty("address.telephone");
ownerContainer.addNestedContainerProperty("address.phoneType");
```

Owners

FIRSTNAME	ID	LASTNAME	PETS	ADDRESS.ADDRESS	ADDRESS.CITY	ADDRESS.TELEPHONE	ADDRESS.PHONETYPE
Kenan	1	Sevindik	[]	ODTÜ	Ankara	1234567	WORK

Tablo'da Veri Gösterme: 1. yol

```
Table ownerTable = new Table("Owners");

ownerTable.setColumnHeader("firstName", "Name");
ownerTable.setColumnHeader("lastName", "Surname");
ownerTable.setColumnHeader("address.address", "Address");
ownerTable.setColumnHeader("address.city", "City");
ownerTable.setColumnHeader("address.telephone", "Phone Number");
ownerTable.setColumnHeader("address.phoneType", "Phone Type");

ownerTable.setContainerDataSource(ownerContainer);
```

Container'dan remove edilmeyen bütün property'ler sütun olarak görüntülenir.

Owners

NAME	ID	SURNAME	ADDRESS	CITY	PHONE NUMBER	PHONE TYPE
Kenan	4	Sevindik	ODTÜ	Ankara	123456	WORK
Ali	5	Veli	ODTÜ	Ankara	123456	WORK

Tablo'da Veri Gösterme: 2. yol

```
Table ownerTable = new Table();

ownerTable.setContainerDataSource(ownerContainer);

ownerTable.setVisibleColumns("firstName", "lastName",
"address.address", "address.city", "address.telephone",
"address.phoneType");


ownerTable.setColumnHeaders("Name", "Surname", "Address", "City",
"Phone Number", "Phone Type");
```

Tablo'dan Veri Seçme: 1. yol

```
table.setSelectable(true);

table.addItemClickListener(new ItemClickListener() {

    @Override
    public void itemClick(ItemClickEvent event) {
        Item item = event.getItem();
        Long itemId = (Long) event.getItemId();
        ...
    }
});
```

Event üzerinden seçilen **Item** nesnesine veya **itemId** değerine erişilebilir

Tablo'dan Veri Seçme: 2. yol

```
table.setImmediate(true);
```

ValueChangeEvent'in satır seçilir seçilmez yakalanabilmesi için gereklidir

```
table.addValueChangeListener(new ValueChangeListener() {

    @Override
    public void valueChange(ValueChangeEvent event) {
        Long itemId = (Long) table.getValue();
        ...
    }
});
```

ValueChangeEvent meydana geldiği vakit Table üzerinden seçilen satırın **itemId** değerine de erişilebilir

BeanItemContainer

```
Table ownerTable = new Table("Owners");

BeanContainer<Integer, Owner> ownerContainer =
        new BeanContainer<Integer, Owner>(Owner.class);

ownerContainer.setBeanIdProperty("id");

ownerContainer.addBean(owner);

ownerTable.setContainerDataSource(ownerContainer);
```

```
Table ownerTable = new Table("Owners");

BeanItemContainer<Owner> ownerContainer =
        new BeanItemContainer<Owner>(Owner.class);

ownerContainer.addBean(owner);

ownerTable.setContainerDataSource(ownerContainer);
```

BeanItemContainer item id olarak bean'in kendisini kullanır.

Bu yüzden bean sınıfındaki **equals()** ve **hashCode()** metodları önemlidir.

BeanContainer'daki gibi identifier property set etmeye gerek yoktur.

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com