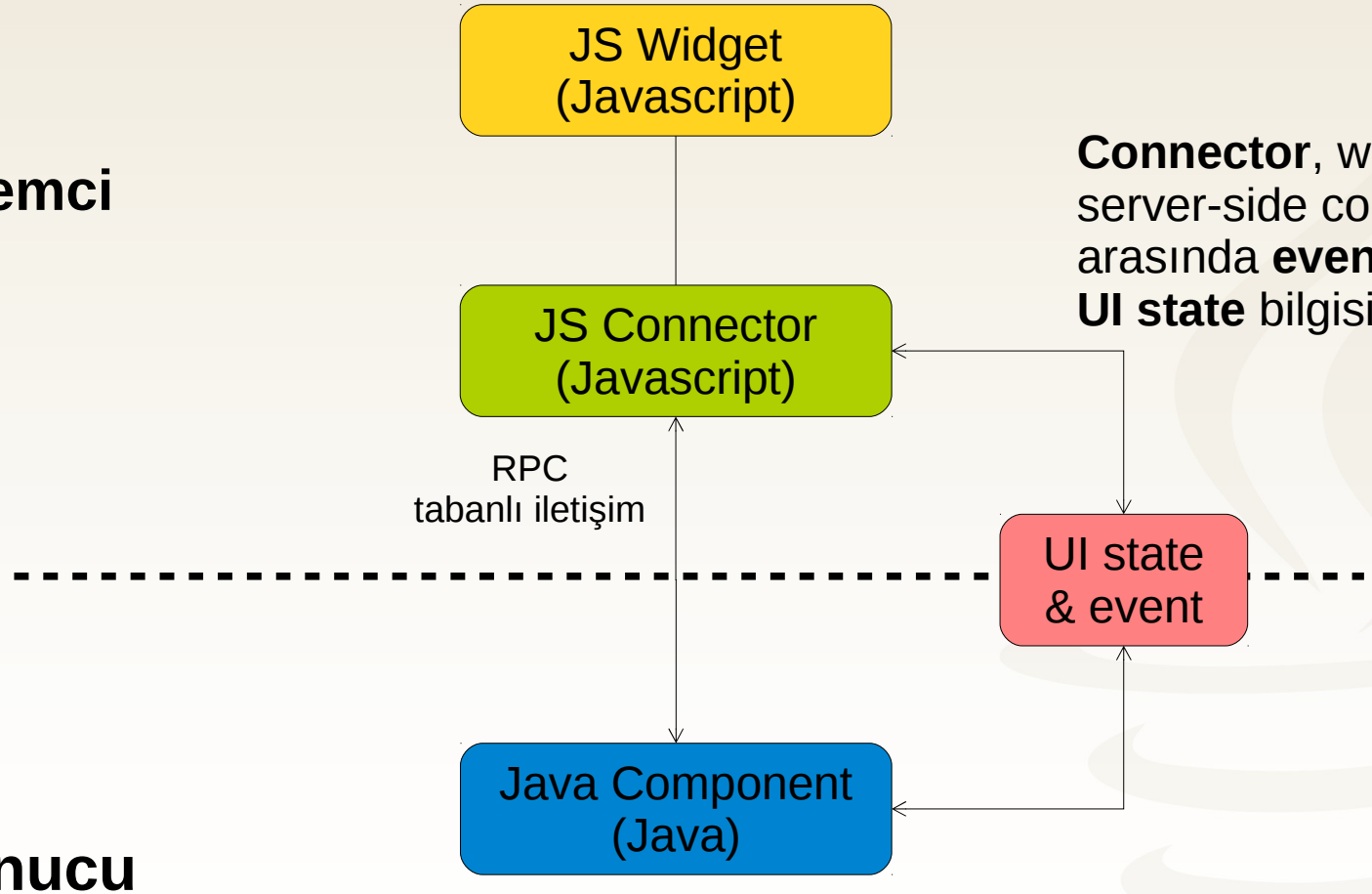
JAVA Eğitimleri

Javascript ile Bileşen Geliştirme

harezmi bilişim çözümleri

JavaScript ile Component Geliştirme



Connector, widget ile server-side component arasında **event** ve **UI state** bilgisini yönetir

JS Component State

```
public class MyComponentState extends JavaScriptComponentState {

    private String text;

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }
}
```

State bilgisini tutmaya yarar.
JavaScriptComponentState sınıfından türer.

Server Side Bileşenin Oluşturulması

```
package com.javaegitimleri.example;
```

Javascript widget'ın çalışması için gerekli JS dosyalarını yüklemek için kullanılır. Dosyalar **projenin (src/main/resources) classpath**'inde MyComponent ile aynı pakette yer almalıdır.

```
@JavaScript({"mywidget.js","mycomponent-connector.js"})
public class MyComponent extends AbstractJavaScriptComponent {

    public MyComponent() {
        getState().setText("hello world from js component!");
    }

    @Override
    protected MyComponentState getState() {
        return (MyComponentState)super.getState();
    }
}
```

Javascript Widget'ın Yazılması

mywidget.js

element, JS'de UI bileşeninin DOM içeriğine karşılık gelir

```
MyWidget = function(element) {

    element.style.border = "thin solid black";
    element.style.display = "inline-block";

    this.setMessage = function(message) {
        element.innerHTML = message;
    };
};
```

JS Widget Connector'ün Yazılması

mycomponent-connector.js

Server side bileşenin FQN'ine karşılık gelen isimde bir JS fonksiyon JS widget connector olarak tanımlanır

```
com_javaegitimleri_example_MyComponent = function() {

    var widget = new MyWidget(this.getElement());

    this.onStateChange = function() {
      widget.setMessage(this.getState().text);
    }

}
```

UI bileşenin DOM element'ini elde etmeyi sağlar

Component'in shared state nesnesini döner. Property'lere private bile olsa isimleri ile erişilebilir

Sunucu tarafındaki state değişikliklerini ele almayı sağlar

Click Event'in Ele Alınması

Fonksiyon dışında Connector wrapper'ın connector değişkenine assign edilmesi önemlidir, çünkü bind edilen fonksiyon içerisinde **this** artık **Element'e refer edecek**tir

```
com_javaegitimleri_petclinic_view_MyComponent = function() {

    var connector = this;

    this.getElement().onclick = function() {

        connector.onClick();

    }

    ...

}
```

Element'in **onclick event**'ine bind edilen fonksiyon içerisinde **connector.onClick()** şeklinde MyComponent'e eklenen "**onClick**" fonksiyonu istenirse input argüman eklenerek invoke edilebilir

Click Event'in Ele Alınması

```
@JavaScript({"mywidget.js","mycomponent-connector.js"})
public class MyComponent extends AbstractJavaScriptComponent {

    public MyComponent() {
        addFunction("onClick", new JavaScriptFunction() {

            @Override
            public void call(JsonArray arguments) {
                Notification.show("widget clicked !");
            }
        });
    }
    ...
}
```

AbstractJavaScriptComponent.callFunction() metodu ise JS Connector içerisinde tanımlanan bir fonksiyonu sunucu tarafından invoke etmeyi sağlar.

İletişim

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