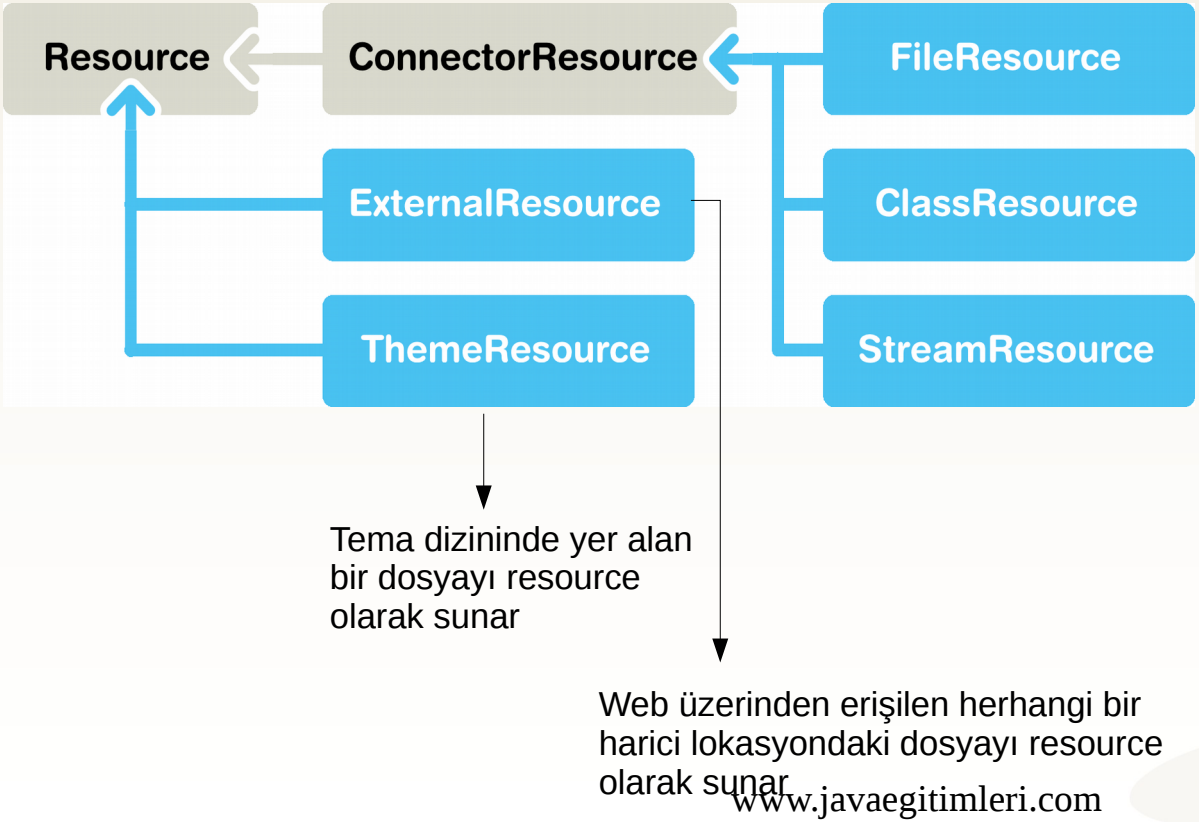
# Resource Yönetimi

# Resource Yönetimi

- Uygulamadaki imaj, video, indirilebilir dosya gibi herhangi bir içeriğe **resource** denir



# FileResource ile Image ve Link Kullanımı

```
// Projenin deploy edildiği dizinin absolute path'i elde edilir
String basepath = VaadinService.getCurrent().getBaseDirectory()
                        .getAbsolutePath();


// Absolute path kullanılarak resim FileResource olarak yüklenir
FileResource resource = new FileResource(new File(basepath +
                            "/WEB-INF/images/image.png"));


// Resim vaadin uygulaması içerisinde Image nesnesi ile gösterilebilir
Image image = new Image("Image from file", resource);


// Ya da resource olarak bir link ile tarayıcı üzerinden erişime açılabilir
Link link = new Link("Link to the image file", resource);
```

# ClassResource ve ThemeResource Kullanımı

```
//resim projenin classpath'inden yüklenir
ClassResource resource1 = new ClassResource("images/image.png");


//resim /VAADIN/petclinic/images/image.png lokasyonundan yüklenir
//petclinic burada uygulamanin aktif temasıdır
ThemeResource resource2 = new ThemeResource("images/image.png");
```

# Dinamik Image Üretme ve Yükleme

İlk yükleme

```
//Dinamik resim bir isim ile
StreamResource kullanılarak
üretilir
StreamResource imageResource =
    new StreamResource(imageSource,
"initial-filename.png");


//Tarayıcının resmi cache'lememesi
belirtilir
imageResource.setCacheTime(0);


//Resim Image bileşeni olarak UI'a
eklenerek gösterilir
//Resmin caption'ı null verilerek
disable edilebilir
Image image = new Image(null,
imageResource);
```

Yeniden yükleme

```
//Image nesnesini markAsDirty
ile işaretlemek tek başına
yeterli değildir
image.markAsDirty();


//Resme zaman damgası içerecek
şekilde yeni bir isim de atamak
gerekir
SimpleDateFormat df = new
SimpleDateFormat("yyyyMMddHHmmss
SSS");
String filename = "myfilename-"
+ df.format(new Date()) +
".png";


imageResource.setFilename(filena
me);
```

# Flash Bileşeni

```
Flash flash = new Flash(null, new ThemeResource("img/myvideo.swf"));

layout.addComponent(flash);
```

- Adobe flash bileşenleri Vaadin UI içerisinde görüntülenebilir
- **Flash.setParameter()** ile flash bileşenin özellikleri set edilebilir

# BrowserFrame Bileşeni

```
BrowserFrame browser = new BrowserFrame("Browser",
    new ExternalResource("http://demo.vaadin.com/sampler/"));

browser.setWidth("600px");
browser.setHeight("400px");

layout.addComponent(browser);
```

- HTML **<iframe>** elemanı içerisine herhangi bir HTML content yerleştirmeyi sağlar
- Resource harici bir lokasyon dan **ExternalResource** ile de yüklenebilir
- BrowserFrame'e **fixed** veya **relatif size** verilmesi önemlidir

# Embedded Bileşeni

- **Embedded** bileşen ile SVG, applet, pdf gibi herhangi bir tür resource Vaadin UI içerisine yerleştirilebilir
- Resource'un **MIME** tipi genellikle dosya ismi uzantısından tespit edilebilir
- İhtiyaç duyulursa ayrıca set edilebilir

```
Embedded object = new Embedded("My SVG", new ThemeResource("img/image.svg"));

object.setMimeType("image/svg+xml");

layout.addComponent(object);
```

# Request Handlers

- **Request parametreleri**ni işleyerek dinamik olarak herhangi bir formatta içerik oluşturmak içindir
- **VaadinSession** ile ilişkilidirler
- Böylece kullanıcıya ait veriye kolaylıkla **RequestHandler** içerisinden erişilebilir
- RequestHandler **arayüzü** implement edilerek yazılırlar
- Aynı anda **birden fazla** RequestHandler devrede olabilir

# RequestHandler Kullanımı

```
RequestHandler requestHandler = new RequestHandler() {

    @Override
    public boolean handleRequest(VaadinSession session,
            VaadinRequest request, VaadinResponse response) throws
IOException {
        if ("/showMessage".equals(request.getPathInfo())) {
                response.setContentType("text/plain");
                response.getWriter().format(
                        "Message is: %s\n",
                session.getAttribute("message"));

                return true;
        } else
                return false;
    }
};

VaadinSession.getCurrent().addRequestHandler(requestHandler);
```

Response üretildi ise return değeri **true** olmalıdır! Bu request handling işleminin sona ermesini sağlar

# RequestHandler Kullanımı

```
TextField message = new TextField("Message");

message.addValueChangeListener(
            new ValueChangeListener() {
    @Override
    public void valueChange(ValueChangeEvent event) {
        VaadinSession.getCurrent().setAttribute(
            "message",event.getProperty().getValue());
    }
});
```

TextField'dan alınan mesaj **VaadinSession** üzerinde **message** isimli bir **attribute**'a set edilir

```
String servletPath = VaadinServlet.getCurrent()
        .getServletContext().getContextPath() + "/app";

Link openLink = new Link("Click to Show Message",
    new ExternalResource(servletPath +
"/showMessage"),
    "_blank", 500, 350, BorderStyle.DEFAULT);

layout.addComponents(message, openLink);
```

Link'e tıklandığında yeni bir pencerenin açılması sağlanarak, **/showMessage URI**'ına **request** gönderilir

# İletişim

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