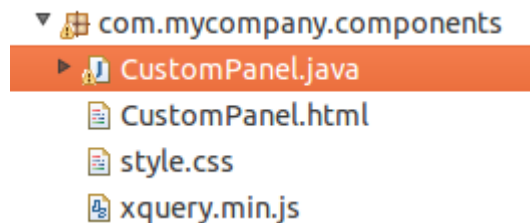# Resources

# Resources

**Grundlagen**

- Resources: CSS, JS, PDF, Images, …

- OO-Repräsentation: ResourceReference

- Statische Resourcen

  - Package Resources

  - HeaderContributors

  - Resource Bundles

  - NIO Resources

- Dynamische Resourcen

  - Custom Resources

  - JAX-RS-Annotationen

- Mounting an URL möglich

- Shared Resources

# Resources

**Package Resources**

- Resourcen in gleichem Package wie HTML + Java

- Distribution mit Komponenten (→ Abgeschlossenheit)

- PackageResourceReference

- Images: Responsive Design

- URL

  - *<Applikation>*/wicket/resource/*<Klasse>*/*<Name>*-<ver-*<id>*>[.*Dateiendung*]

  - ./wicket/resource/net.gfu.CustomPanel/calendar-ver-1297887542000.jpg

```
▼ 🎛 com.mycompany.components
    ▶ 📄 CustomPanel.java
      📄 CustomPanel.html
      📄 style.css
      📄 xquery.min.js
```

## Package Resources

- HTML

```html
<html>

<head></head>

<body>

<wicket:panel>
    Package resource image: <img wicket:id="packageResPicture"/>
</wicket:panel>

</body>

</html>
```

# Resources

## Package Resources

- Java

```java
public class CustomPanel extends Panel {

    public CustomPanel(String id) {
        super(id);

        Image image = new Image("packageResPicture",
                new PackageResourceReference(getClass(), "large.jpg"),
                new PackageResourceReference(getClass(), "medium.jpg"),
                new PackageResourceReference(getClass(), "small.jpg"));
        image.setXValues("1024w", "640w", "320w");
        image.setSizes("(min-width: 36em) 33.3vw", "100vw");

        this.add(image);
    }
}
```

# Resources

**Header Contributors**

- Interface *IHeaderContributor*
- Einfügen von Header Items in umgebende Page
  - CSSHeaderItem
  - JavaScriptHeaderItem
  - MetaDataHeaderItem
  - OnDomReadyHeaderItem
  - PriorityHeaderItem
  - …
- Jede Komponente ist selbst HeaderContributor!

# Resources

## Header Contributors

- Java

```java
public class MyComponent extends Component{

    @Override
    public void renderHead(IHeaderResponse response) {
        PackageResourceReference cssFile =
                new PackageResourceReference(this.getClass(), "style.css");
        CssHeaderItem cssItem = CssHeaderItem.forReference(cssFile);

        response.render(cssItem);
    }
}
```

**Resource Bundles**

- <u>Zusammenfassung</u> von Ressourcen (vgl. Grunt)

- Nicht zu verwechseln mit Java Resource Bundles!

- Registrierung in Application-Klasse

- Applikationsweite Verwendung

## Resource Bundles

- Java

```java
@Override
public void init() {
    super.init();

    getResourceBundles().addJavaScriptBundle(WicketApplication.class,
            "jqueryUiJs",
            jqueryJsReference,
            jqueryUiJsReference);

    getResourceBundles().addCssBundle(WicketApplication.class,
            "jqueryUiCss",
            jqueryCssReference,
            jqueryUiCssReference);

}
```

# Resources

## NIO Resources

- Zugriff auf Resourcen mittels Java NIO (seit Wicket 7.2.0)
- Laden von Daten aus dem Dateisystem

- HTML

```html
<video wicket:id="video"/>
```

- Java

```java
URI uri =
URI.create("jar:file:///videosFolder/videos.zip!/folderInZip/MyVideo.mp4");

Path path = FileSystemResourceReference.getPath(uri);
FileSystemResourceReference ref = new FileSystemResourceReference("myvideo", path);
Video video = new Video("video", ref);
add(video);
```

# Resources

**Custom Resources**

- Einbindung dynamisch zu erstellender Inhalte
- Erzeugung bei Bedarf durch WriteCallback
- Direkter Zugriff auf ResourceResponse

# Resources

```java
public class RSSProducerResource extends AbstractResource {

    @Override
    protected ResourceResponse newResourceResponse(Attributes attributes) {
        ResourceResponse resourceResponse = new ResourceResponse();
        resourceResponse.setContentType("text/xml");
        resourceResponse.setTextEncoding("utf-8");

        resourceResponse.setWriteCallback(new WriteCallback()
        {
            @Override
            public void writeData(Attributes attributes) throws IOException
            {
                OutputStream outputStream = attributes.getResponse().getOutputStream();
                Writer writer = new OutputStreamWriter(outputStream);
                SyndFeedOutput output = new SyndFeedOutput();
                try {
                    output.output(getFeed(), writer);
                } catch (FeedException e) {
                    throw new WicketRuntimeException("Problems writing feed to response...");
                }
            }
        });

        return resourceResponse;
    }
    // Methode getFeed()…
}
```

## Shared Resources

- Applikationsweit verfügbare Ressourcen

- Registrierung in Application-Klasse

- Einbindung nach Bedarf

# Resources

## Shared Resources

- Application-Klasse

```java
@Override
public void init(){
        RSSProducerResource rssResource = new RSSProducerResource();
        // …
        getSharedResources().add("globalRSSProducer", rssResource);
}
```

- Einbindende Komponente

```java
add(new ResourceLink("globalRssLink", new
SharedResourceReference("globalRSSProducer")));
```