Agenda

- **Tag 1 – Einführung**
  - Installation
  - Erste Anwendung
  - Architektur
- **Tag 2 – Entwicklung**
  - Models
  - Darstellung
  - Formulare
  - Ajax
- **Tag 3 – Fortgeschrittene Themen**
  - Tests
  - Security & Deployment
  - Lokalisierung & Internationalisierung
  - Performance
  - Best Practices

MOTIVATION

Historie

- Timeline:
  - **2004**: Gründung
  - **2005**: Wicket 1.0, 1.1
  - **2006**: Wicket 1.2
  - **2007**: → Apache Foundation
  - **2008**: Wicket 1.3, „Wicket in Action“
  - **2009**: Wicket 1.4
  - **2011**: Wicket 1.5
  - **2012**: Wicket 6 (aka 1.6)
  - **2014**: Wicket 7
  - **2018:** Wicket 8
  - **2020**: Wicket 9
- Aktuell:
  - Wicket 9.x 9.12.0 current, supported
  - Wicket 8.x 8.14.0 supported
  - Wicket 7.x 7.18.0 security fixes only, upgrade to 8.x or 9.x
  - Wicket 6.x 6.30.0 discontinued, no longer maintained and will no longer receive any updates
  - https://wicket.apache.org/start/wicket-9.x.html#migrate

Community

- Weiterentwicklung
- Dokumentation und Tutorials
- Beispiele
- Vorgefertigte Komponenten
- Bug Tracker

http://wicket.apache.org/community
http://wicket.apache.org/help
http://wicket.apache.org/learn

Warum Wicket?

anderScore trust in competence

GFU Cyrus AG

"*Wicket bridges the impedance mismatch between the stateless HTTP and stateful server-side programming in Java.*"

M.Dashorst, E.Hillenius, *Wicket in Action*

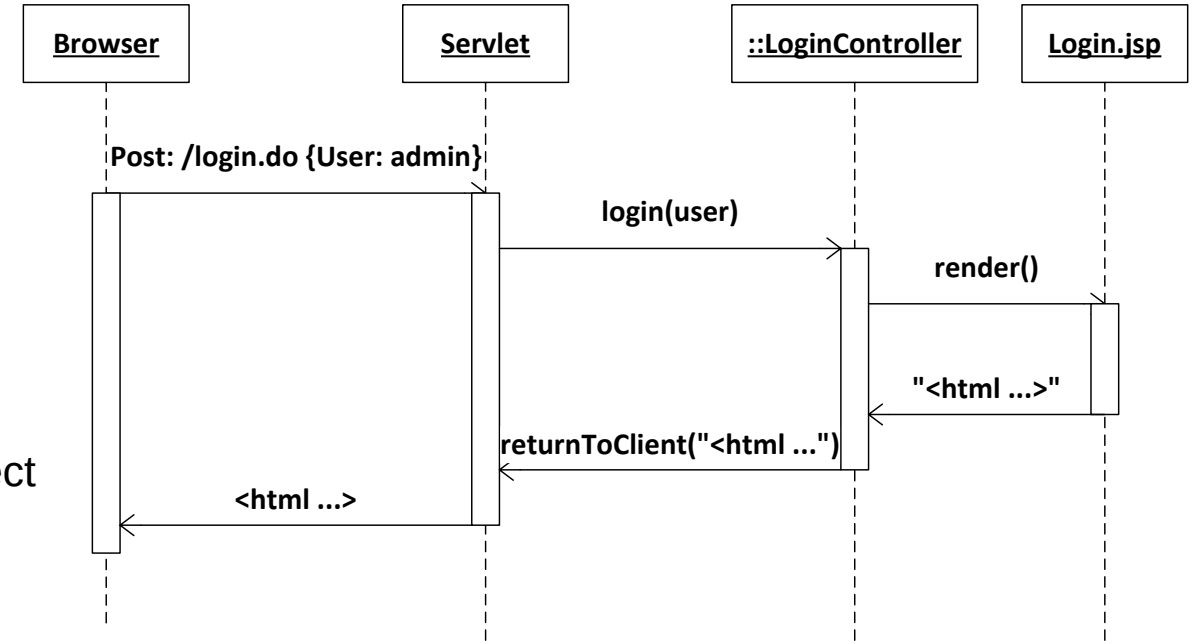
Java Virtual Machine (JVM): Classes, Objects, Members

vs.

Hyper Text Transfer Protocol (HTTP): Stateless, Request Response

Model 2 Frameworks

- Klass. MVC Ansätze:
  - Spring MVC
  - Apache Struts
  - Ruby on Rails
  - Grails



- Vorgehen:
  - Routing:
    URL ↔ Controller Object
  - HTTP-Request
    → Action-Methode
    → View
- Idee:
  - Abstraktion `login(user)` statt `doPost(request)`
  - Viele Helper / Taglibs: Links, Formulare, etc.

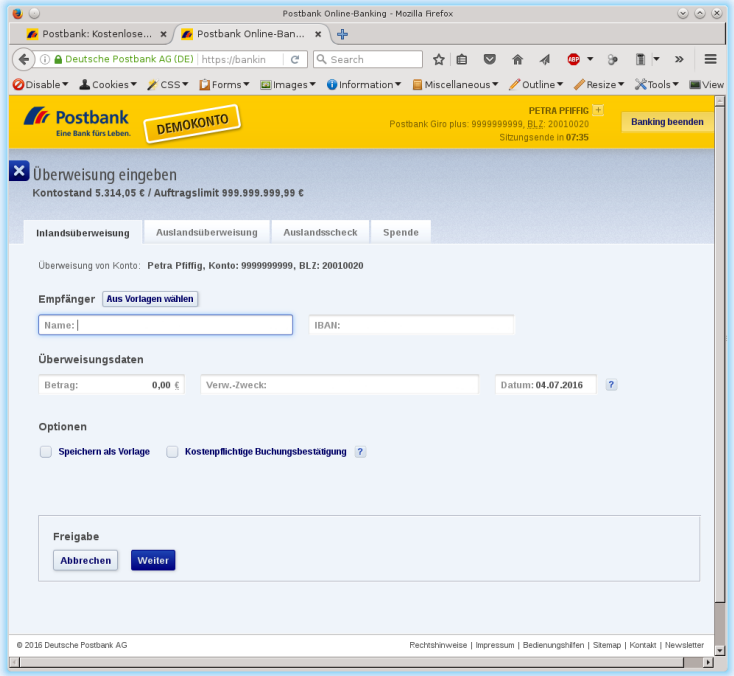
- **Passt das zu den Anforderungen?**

Ziel: Komplexe Webanwendung

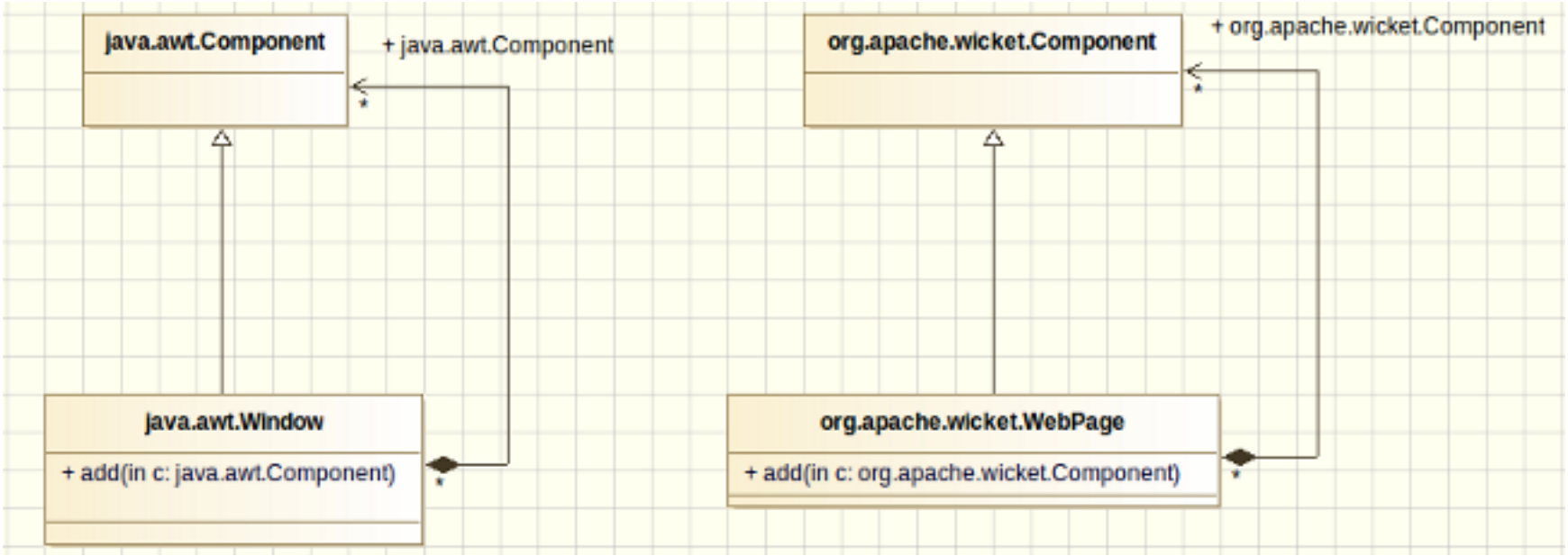
- User Interfaces modellieren
  Model-View-{ Controller, Presenter, ViewModel }

- Analog zu: AWT, Swing, Java FX

```
public class Index extends WebPage {

    private Label linkLabel;
    private Link<Void> link;

    public Index(){
        int linkClicks = 0;

        linkLabel = new Label("link-label", Model.of(linkClicks));

        link = new Link<Void>("link-link"){
            @Override
            public void onClick() {
                linkClicks++;
                linkLabel.setDefaultModelObject(linkClicks);
            }
        };

        add(linkLabel);
        add(link);
    }
}
```



Wicket vs. AWT



Zustand der UI

- Problem: Zustand der UI verfolgen
  - HTTP ist Stateless
  - Idee: Zustand in URL codieren?

```
/tsp/web?lefttab=mngworkspaces&ctab=groups&ltab=members&rtab=comps…
```

- Wicket: Zustand der UI in Session ablegen
  - Komponentenbaum serialisieren
  - URLs referenzieren gespeicherte Seiten
  - Vorsicht: Anzahl der Seiten begrenzen

- **Offen**: Aussehen der UI beschreiben.

UI Design

- Wicket Philosophie:
  Just Java, just HTML
- Layout in HTML (+ CSS)
  Wicket-IDs referenzieren Komponenten
- Damit: Keine *eigene* Beschreibungssprache
- Round-Trip:
  Designer ↔ Entwickler möglich

```
<div class="col-md-12">
   <h2>Echo-Formular</h2>
   <form wicket:id="form">
      <div class="form-group">
         <input wicket:id="echo-input" type="text"/>
      </div>
      <button type="submit">Submit</button>
   </form>
   <pre wicket:id="echo-message"></pre>
</div>
```

```
public EchoFormPage() {
   Model<String> inputModel = new Model<>();
   TextField<String> textField = new TextField<>("echo-input", inputModel);

   form = new Form<>("form");
   form.add(textField);

   message = new Label("echo-message", inputModel);

   add(form);
   add(message);
}
```

Zusammenfassung

- Klassische MVC-Frameworks
  - Routing: Controller, View
  - UI-Komponenten werden nicht modelliert
  - Wenig Hilfe bei komplexen User Interfaces
- Wicket
  - Komponenten-orientiert
  - Models und Zustand der Komponenten in Session serialisiert
  - Java-API beschreibt die Komponenten
  - HTML beschreibt das Aussehen
  - XML-Attribute stellen Bezug her
- Vorsicht: Session evtl. groß
  - Speicherverbrauch
  - Cluster-Overhead
  - Wichtig: Größe der Session beschränken!

Wicket in Beispielen

1. Hello Wicket
2. Link Click Counting
3. Echo Server

Die erste Wicket Anwendung

HELLO WICKET

Installation

Voraussetzungen

- JDK 8 oder neuer
- Servlet API 3.1
- SLF4J

- https://jdk.java.net/8/

Installation

anderScore trust in competence

GFU Cyrus AG

Manuelle Installation

- Einbindung von JARs

http://www.apache.org/dyn/closer.cgi/wicket/8.14.0/apache-wicket-8.14.0.zip

oder

- Maven

```
<dependency>
    <groupId>org.apache.wicket</groupId>
    <artifactId>wicket-core</artifactId>
    <version>8.14.0</version>
</dependency>
```

Installation

anderScore trust in competence

GFU Cyrus AG

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
    <display-name>Hello Wicket</display-name>
    <filter>
        <filter-name>WicketApplication</filter-name>
        <filter-class>org.apache.wicket.protocol.http.WicketFilter
        </filter-class>
        <init-param>
            <param-name>applicationClassName</param-name>
            <param-value>nrw.it.hellowicket.WicketApplication</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>WicketApplication</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
</web-app>
```

Installation

… oder WebFilter

```
@WebFilter(value = "/*", initParams = {
    @WebInitParam(name = "applicationClassName",
        value = "com.mycompany.WicketApplication"),
    @WebInitParam(name="filterMappingUrlPattern",
       value="/*")
})
public class ProjectFilter extends WicketFilter {
}
```

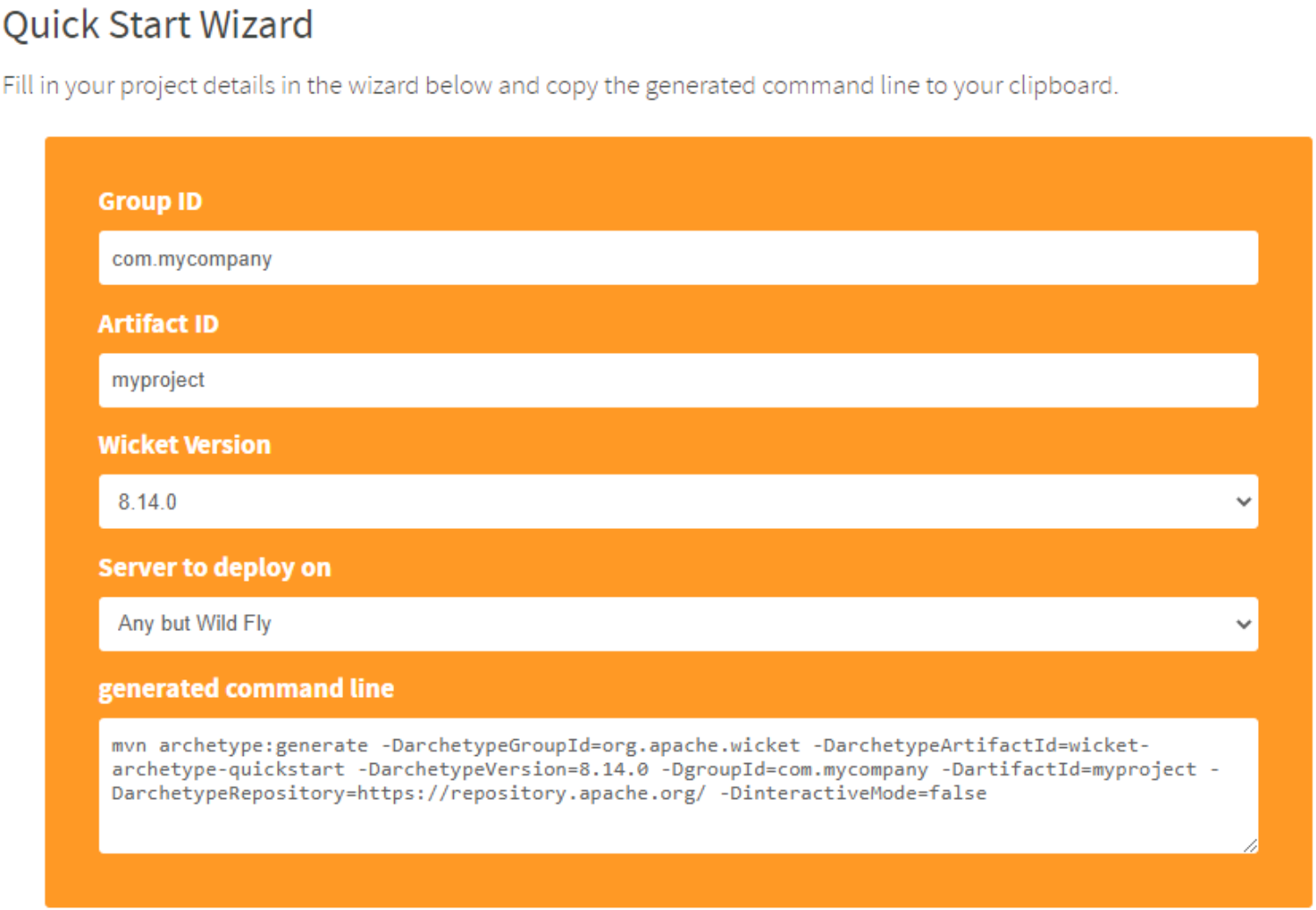
Installation

Empfehlung: Wicket Quick Start

- Generierung der Projektstruktur
  - Lauffähige Wicket-Applikation
  - Starter für Server Jetty
- Eigener Maven-Archetype
- Import in IDE Ihrer Wahl

- http://wicket.apache.org/start/quickstart.html

Installation



Installation

anderScore trust in competence

GFU Cyrus AG

Result of the Maven command

Executing the Maven command line will result the following directory structure:

```
.\myproject
    |   pom.xml
    |
    \---src
        +---main
        |   +---java
        |   |   \---com
        |   |       \---mycompany
        |   |               HomePage.html
        |   |               HomePage.java
        |   |               WicketApplication.java
        |   |
        |   +---resources
        |   |       log4j.properties
        |   |
        |   \---webapp
        |       \---WEB-INF
        |               web.xml
        |
        \---test
            \---java
                \---com
                    \---mycompany
                            Start.java
```

Links, Models, Ajax

KLICKS ZÄHLEN

Klicks zählen - HTML

```
<div class="container">
   <h2>Anzahl der Klicks</h2>
   Dieser <a wicket:id="link-link" href="#">Link</a>
   wurde <span wicket:id="link-label">4711</span> mal geklickt!

   Dieser <a wicket:id="ajax-link-link" href="#">Ajax-Link</a>
   wurde <span wicket:id="ajax-link-label">4711</span> mal geklickt!
</div>
```

Klicks zählen - Java

anderScore trust in competence

GFU Cyrus AG

```
public ClickCounterPage() {
    linkClicks = 0;
    ajaxLinkClicks = 0;

    Model<Integer> linkLabelModel = Model.of(linkClicks);
    Model<Integer> ajaxLinkLabelModel = Model.of(ajaxLinkClicks);

    linkLabel = new Label("link-label", linkLabelModel);

    ajaxLinkLabel = new Label("ajax-link-label", ajaxLinkLabelModel);
    ajaxLinkLabel.setOutputMarkupId(true);

    link = new Link<Void>("link-link"){
        @Override
        public void onClick() {
            linkClicks++;
            linkLabelModel.setObject(linkClicks);
        }
    };

    ajaxLink = new AjaxFallbackLink<Void>("ajax-link-link") {
        @Override
        public void onClick(Optional<AjaxRequestTarget> optional) {
            if (optional.isPresent()) {
                ajaxLinkClicks++;
                ajaxLinkLabelModel.setObject(ajaxLinkClicks);

                optional.get().add(ajaxLinkLabel);
            }
        }
    };

    add(linkLabel, ajaxLinkLabel);
    add(link, ajaxLink);
}
```

Ergebnis eines Formulars anzeigen

ECHO SERVER

Echo Server

anderScore trust in competence

GFU Cyrus AG

- Java

```
public EchoFormPage() {
    Model<String> inputModel = new Model<>();
    TextField<String> textField = new TextField<>("echo-input", inputModel);


    form = new Form<>("form");
    form.add(textField);


    message = new Label("echo-message", inputModel);


    add(form);
    add(message);
}
```

- HTML

```
<div class="container">
    <h2>Echo-Formular</h2>
    <form wicket:id="form">
        <div class="form-group">
            <label for="echo-input">Geben Sie einen Text ein:</label>
            <input id ="echo-input" wicket:id="echo-input" class="form-control" type="text"/>
        </div>
        <button type="submit" class="btn btn-default">Submit</button>
    </form>
    <h2>Server-Antwort</h2>
    <pre wicket:id="echo-message"></pre>
</div>
```

Aufgabe: Wicket Start

- Aufgabenstellung
  - Erzeugen Sie ein neues Wicket-Projekt.
  - Nutzen Sie `wicket-archetype-quickstart`.
  - Erstellen Sie zwei Links (synchron, ajaxfallback). Geben Sie die Anzahl der Klicks aus
  - Erstellen Sie ein Formular (Textfeld + Submit Button). Wenn der Benutzer das Formular absendet, dann wird der Eingabetext ausgegeben.
  - Zählen Sie die Seitenaufrufe der `HomePage` mit einem `static`-Counter - keine Persistenz. Geben Sie die Anzahl aus

- Lösungshinweise
  - https://wicket.apache.org/start/quickstart.html
  - Testserver im Debug-Modus → HTML- und Java-Code Änderungen werden ohne Neustart angewendet.