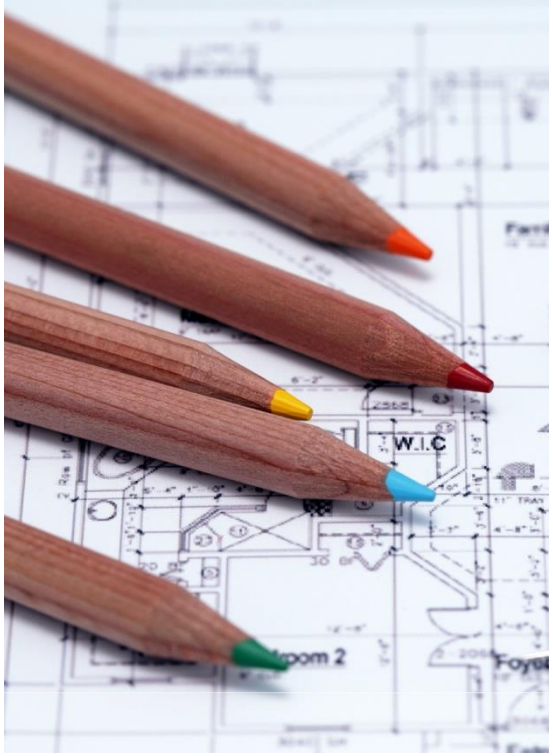


ARS

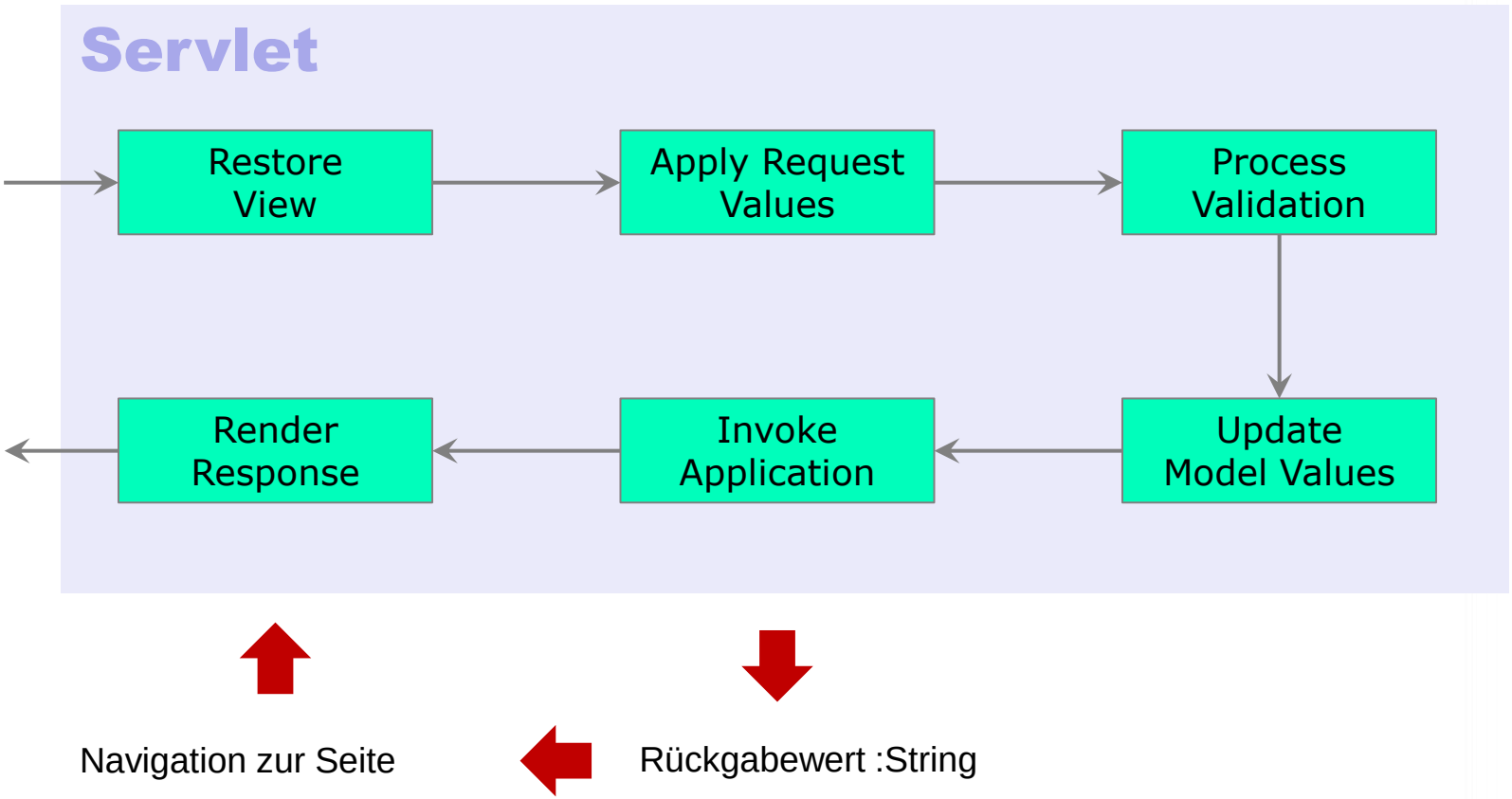
Java Server Faces

Navigation

Überblick

- Allgemeines
  - Anwendung von Navigation
  - Dynamische vs. Statische Navigation
  - Explizite vs. Implizite Navigation
- Navigationsregeln
  - Navigation von einer Seite zur nächsten (1:1)
  - Navigation von mehreren Seiten zu einer (n:1)
  - Navigation von einer Seite zu mehreren (1:n)
- Navigation Handler

Anwendung von Navigation



ARS

Allgemeines

Dynamische Navigation

- Aufruf einer Aktionsmethode (z.B. Method Binding an Button)
  - Rückgabewert (String) abhängig von der Ausführung der Methode

```
<h:commandButton action="#{person.sprechen}">
   […]
</h:commandButton>
```

Statische Navigation

- Angabe des Rückgabewertes direkt in Facelet
- Kein Aufruf einer Aktionsmethode

```
<h:commandLink action="menu">
   […]
</h:commandLink>
```

Allgemeines

Explizite Navigation

- Navigationsregel für den Rückgabewert mit Angabe einer neuen Seite
- Vorteil: Bessere Wartbarkeit durch zentralisierte Angabe von Pfaden

```
<h:commandLink action="page2">
   […]
</h:commandLink>
```

`page2` → `/index2.xhtml`

`/index2.xhtml`

Implizite Navigation

- Angabe der neuen Seite direkt (ohne Navigationsregel)
- Fallback, wenn keine Navigationsregel gefunden wird
- Vorteil: weniger Navigationsregeln (`faces-config.xml` schlanker)

```
<h:commandLink action="page2">
   […]
</h:commandLink>
```

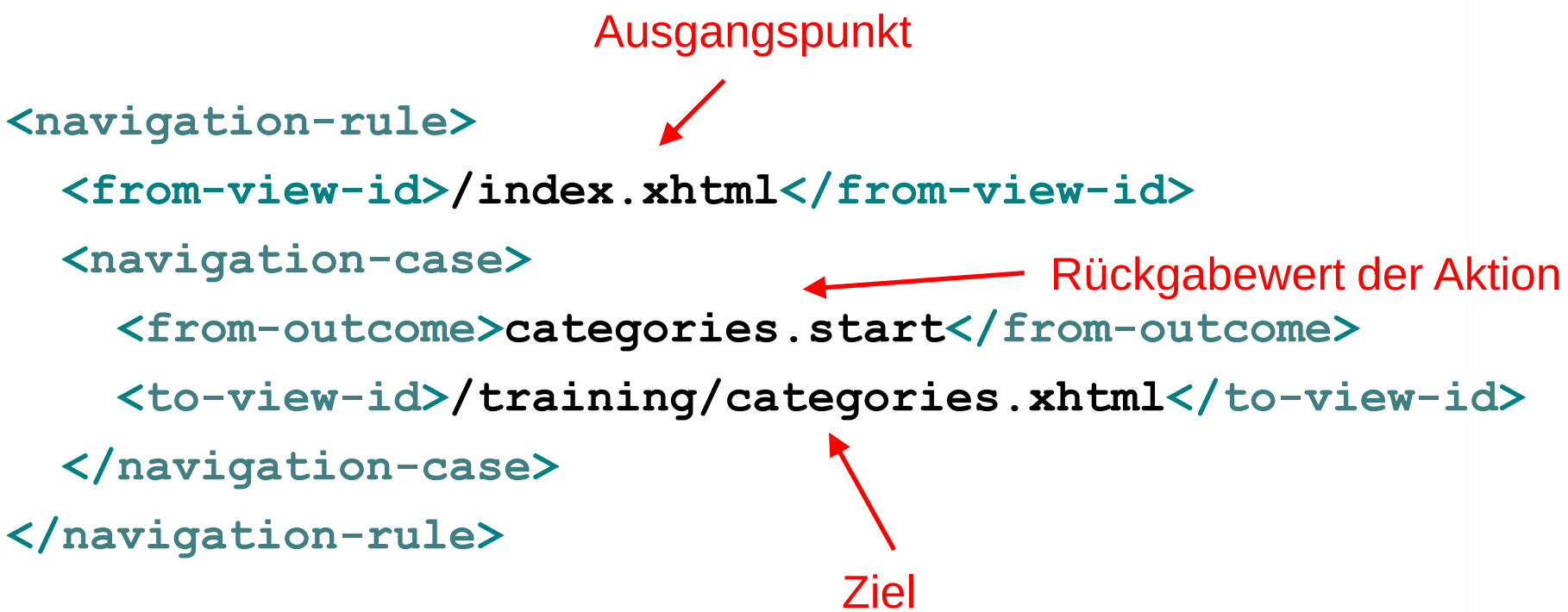
`???`

`/page2.xhtml`

Navigationsregeln – Konfiguration

Navigation von einer Seite zur nächsten

- Eintrag in der Konfigurationsdatei `faces-config.xml`



Navigationsregeln – Konfiguration

Navigation von mehreren Seiten zu einer

- Navigation zur gleichen Ressource in der Anwendung
- Ausgangspunkt wird nicht festgelegt
  - Element `<from-view-id>` wird ausgelassen
- Alternativ Verwendung des Platzhalters (*)
  - `<from-view-id>*</from-view-id>`

```
<navigation-rule>
  <navigation-case>
    <from-outcome>categories.start</from-outcome>
    <to-view-id>/training/categories.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

Navigationsregeln – Konfiguration

Navigation von einer Seite zu mehreren

- Definition mehrerer Navigationsfälle für eine Ressource
- Verwendung von mehreren `<navigation-case>`-Elemente

```
<navigation-rule>
  <from-view-id>/index.xhtml</from-view-id>
  <navigation-case>
    <from-outcome>module.start</from-outcome>
    <to-view-id>/individualcourse/modules.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-outcome>search.start</from-outcome>
    <to-view-id>/search/search.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

Navigationsregeln – Konfiguration

Redirect

- Verwendung des Elements `<redirect/>` im Navigationsfall
- Aktuelle Anfrage wird als abgeschlossen betrachtet
- Zugriff auf die nächste Seite in Form eines HTTP Redirects

```
<navigation-case>
  <from-outcome>search.start</from-outcome>
  <to-view-id>/search/search.xhtml</to-view-id>
  <redirect/>
</navigation-case>
```

(Redirect für implizite Navigation)

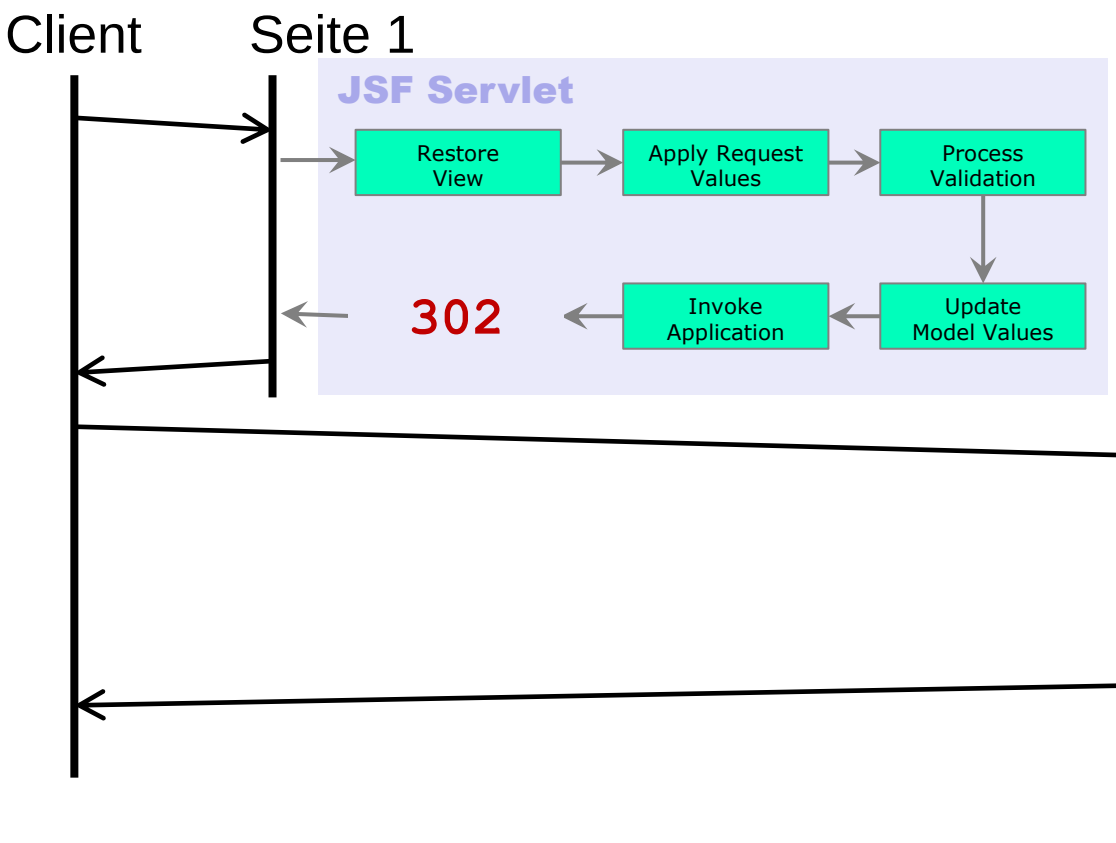
- Parameter in Rückgabewert

```
<h:commandLink action="page2?faces-redirect=true">
```

Navigationsregeln – Konfiguration

Flash Scope

- Gültigkeitsbereich für Objekte über nächsten Request hinweg
- Notwendigkeit bei `@RequestScoped` Beans für Anzeige von Daten im Facelet bei Redirect



Aktion:

```
FacesContext fc = FacesContext.getCurrentInstance();
Flash flash = fc.getExternalContext().getFlash();
flash.put( "flight", selectedFlight );
return "nextPage?faces-redirect=true";
```

Facelet:

```
#{flash.flight.price}
```

Navigationsregeln

Mehrdeutigkeit

- Eine gemeinsame Regel für mehrere Seiten
- Seiten sind mit Methoden verknüpft, die den gleichen logischen Namen zurückliefern
- Es soll zu unterschiedlichen Seiten weiter navigiert werden

Lösung

- Für jede Methode einen eigenen Navigationsfall definieren
- Im Element `<from-action>` die Methode festlegen
- Verwendung eines Ausdrucks für die Methodendefinition

Navigationsregeln

```
<navigation-rule>
  <from-view-id>/training/*</from-view-id>
  <navigation-case>
    <from-action>#{bookingAction.execute}</from-action>
    <from-outcome>success</from-outcome>
    <to-view-id>/training/registration.xhtml</to-view-id>
  </navigation-case>
  <navigation-case>
    <from-action>#{registrationAction.execute}</from-action>
    <from-outcome>success</from-outcome>
    <to-view-id>/training/overview.xhtml</to-view-id>
  </navigation-case>
</navigation-rule>
```

Navigation Handler

Navigation Handler

- Komponente, die aufgrund des Aktionsergebnisses Navigation ausführt
- Standardverhalten
  - Wenn Aktionsergebnis **`null`**, dann keine Navigation
  - Suche nach Navigationsregel, sonst implizite Navigation
- Eigener Navigation Handler erstellbar
  - Eigener Automatismus basierend auf Rückgabewerten
  - Unterklasse von `javax.faces.application.NavigationHandler`
  - Registrieren in `faces-config.xml`


```
<application>
  <navigation-handler>de.ars.jsf.MyNavigationHandler</navigation-handler>
</application>
```

Kontrollfragen

- Was ist der Unterschied zwischen statischer und dynamischer Navigation?
- Worin unterscheiden sich explizite und implizite Navigation und welche Vorteile bieten beide Varianten? Wie werden diese Varianten im FacesServlet angewandt?
- Wann muss ein eigener NavigationHandler implementiert werden?