



# Java Server Faces

## Einführung

- Motivation
- Umsetzung
- Einordnung in technische Umwelt

### **Servlet API**

- Programmiermodell zur Erstellung von Webapplikationen
- Keine GUI-spezifischen Vorgaben
  - Oberflächenkomponenten und deren Zustand
  - Interaktionen mit dem Benutzer

### **Request-Response-Modell**

- Ereignis am Client erzeugt Request
- Request muss am Server verarbeitet werden
  - Analysieren des Request, Zuordnung zum Kontext
  - Ausführen einer Aktion
  - Generieren einer Antwort, Anpassen des GUI

***- Standardisiertes Framework -***

## Typische Schritte bei der Requestverarbeitung

### Terminanfrage

#### JavaServer Faces – Programmierung mit Eclipse

Bitte füllen Sie die notwendigen Felder aus. Geben Sie weitere Kontaktinformationen in das Nachrichtenfeld ein, falls erwünscht.

Name*	Titel	Vorname*	Nachname*
<input type="text" value="Frau"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Firma*		Abteilung / Beruf*	
<input type="text"/>		<input type="text"/>	
Straße	Land	PLZ	Stadt
<input type="text"/>	<input type="text" value="D"/>	<input type="text"/>	<input type="text"/>
Email*	Telefon*	Fax	
<input type="text"/>	<input type="text" value="+49-"/>	<input type="text"/>	
Wunschtermin*	Anzahl der Teilnehmer*	Lokation*	
<input type="text"/>	<input type="text" value="1"/>	<input type="text" value="Wir stellen selbst einen Raum bereit."/>	
Ihre Nachricht			
<input type="text"/>			
Berechnen Sie bitte die Differenz von eins und vier:*			
<input type="text"/>			

Absenden

\*Für fett markierte Felder ist die Eingabe erforderlich.

## Typische Schritte bei der Requestverarbeitung

### 1. Anfrage-Parameter bzw. -Header

a) Auslesen

b) Auf Nullwerte prüfen

c) Konvertieren

d) Validieren

!! HTTP 200

### 2. Aktion ausführen

!! HTTP 500 !!

### 3. Antwort generieren

??

### Reaktionen auf Fehler:

Terminanfrage

JavaServer Faces – Programmierung mit Eclipse

Bitte füllen Sie die notwendigen Felder aus. Geben Sie weitere Kontaktinformationen in das Nachrichtenfeld ein, falls erwünscht.

Name*	Titel	Vorname*	Nachname*
Frau			
Firma*		Abteilung / Beruf*	
Straße	Land	PLZ	Stadt
	D		
Email*	Telefon*	Fax	
	+49-		
Wunschtermin*	Anzahl der Teilnehmer*	Lokation*	
	1	Wir stellen selbst einen Raum bereit.	
Ihre Nachricht			
<div></div>			

Berechnen Sie bitte die Differenz von eins und vier:\*

Absenden

\*Für fett markierte Felder ist die Eingabe erforderlich.

Absenden

\*Für fett markierte Felder ist die Eingabe erforderlich.

## UI State Saving

- Verwaltung der Benutzeroberfläche als Komponentenbaum
  - Klassen/Interfaces für UI-Komponenten
  - Programmatischer Zugriff
    - zum Auslesen der UI
    - zur Manipulation der UI
    - zum Registrieren von Event Handlern

```
FacesContext ctx = FacesContext.getCurrentInstance();
Application app = ctx.getApplication();

HtmlInputText txtInput =
    (HtmlInputText) app.createComponent(
    HtmlInputText.COMPONENT_TYPE);
txtInput.setAccesskey("a");

ctx.getViewRoot().getChildren().add(txtInput);
```

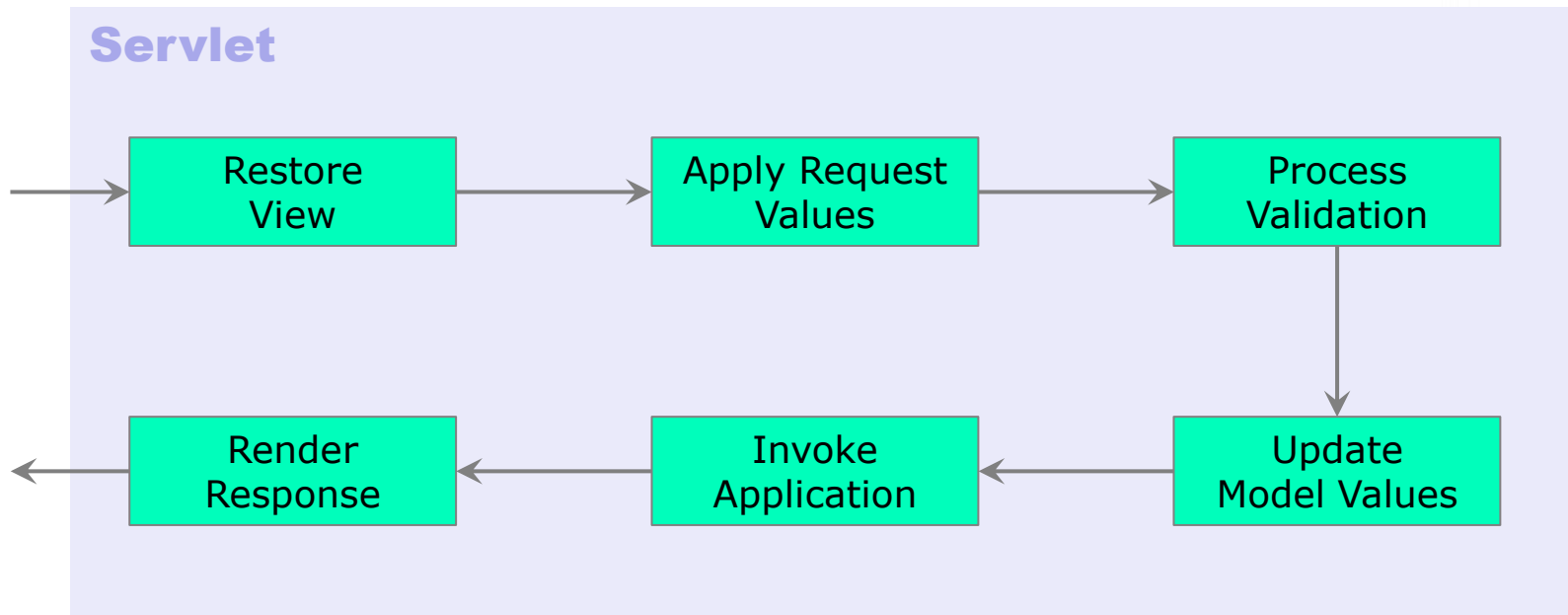
## UI State Saving

- (Initiale) Deklaration des Komponentenbaumes einer Seite
  - per JSP (JSF 1.x)
  - per Facelet (ab JSF 2)

```
<h:head>
  <h:outputStylesheet library="css" name="default.css" />
  <title>My Sample Page</title>
</h:head>
<h:body>
  <h:graphicImage library="images" name="logo.gif" alt="Logo" />
  <h:outputText value="Hello there!" />
</h:body>
```

- Speichern des Komponentenbaumes (UI State) in der Session
  - History-Funktion: Komponentenbäume der letzten n-Seiten

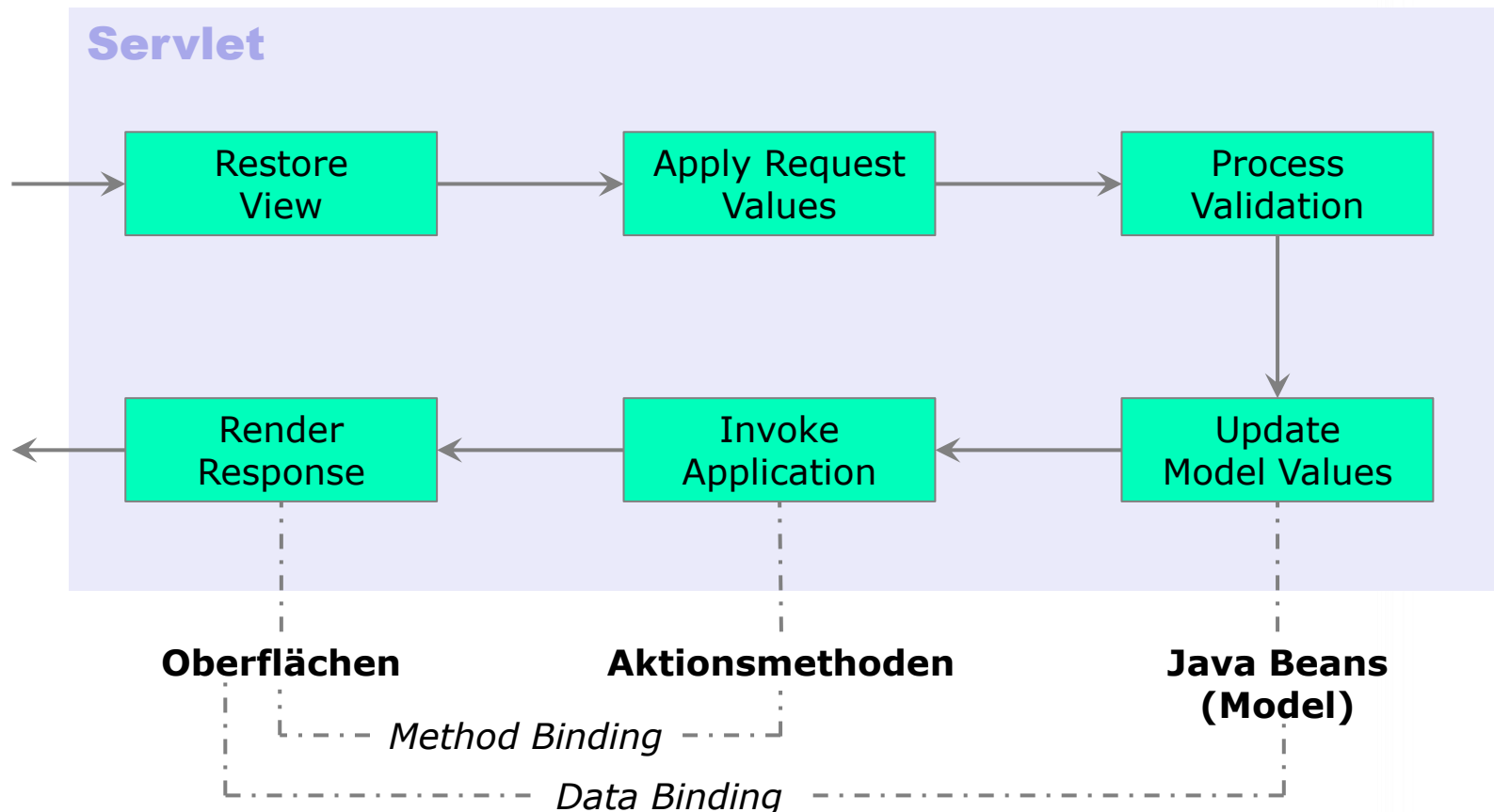
### JSF Front Controller





### Paradigma: „Desktop-Entwicklung“

- Entwicklung von Oberflächen und Ereignissen




```
...  
<h:inputText id="user" value="#{loginBean.username}"/>  
...  
<h:inputSecret id="password" value="#{loginBean.password}"/>  
...  
<h:commandButton id="submit" action="#{loginBean.doLogin}" value="Submit"/>
```

Benutzername:

Passwort:

### Navigationsregeln

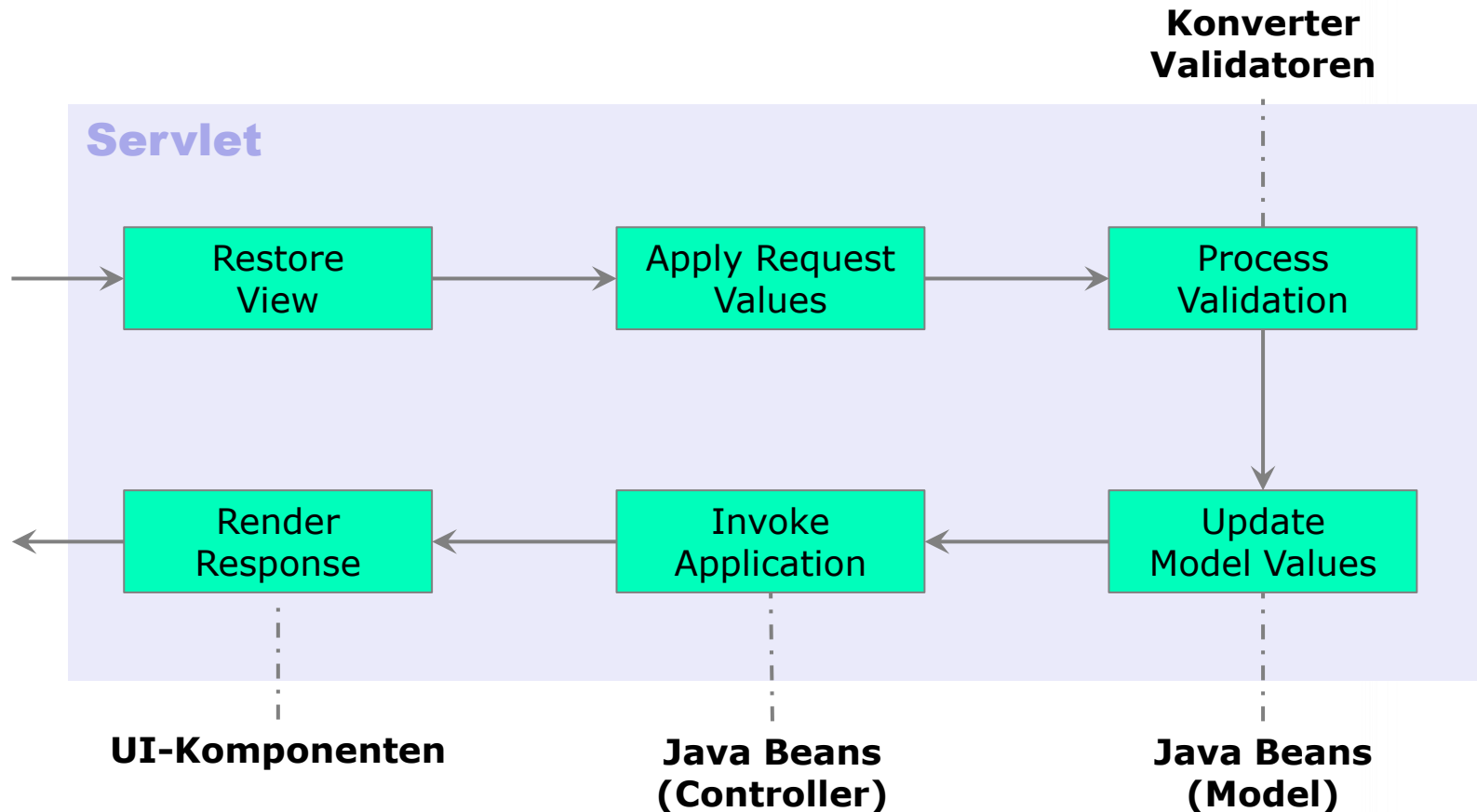
-wenn success -> Menüseite  
-wenn failure -> Loginseite



```
public class LoginBean {  
  
    public String getUsername() { [...]}  
    public String getPassword() { [...]}  
  
    public String doLogin() {  
        [...]  
        return "success";  
    }  
}
```



### Komponentisierung der Applikation



## Konfigurierbarkeit

### ■ Applikation

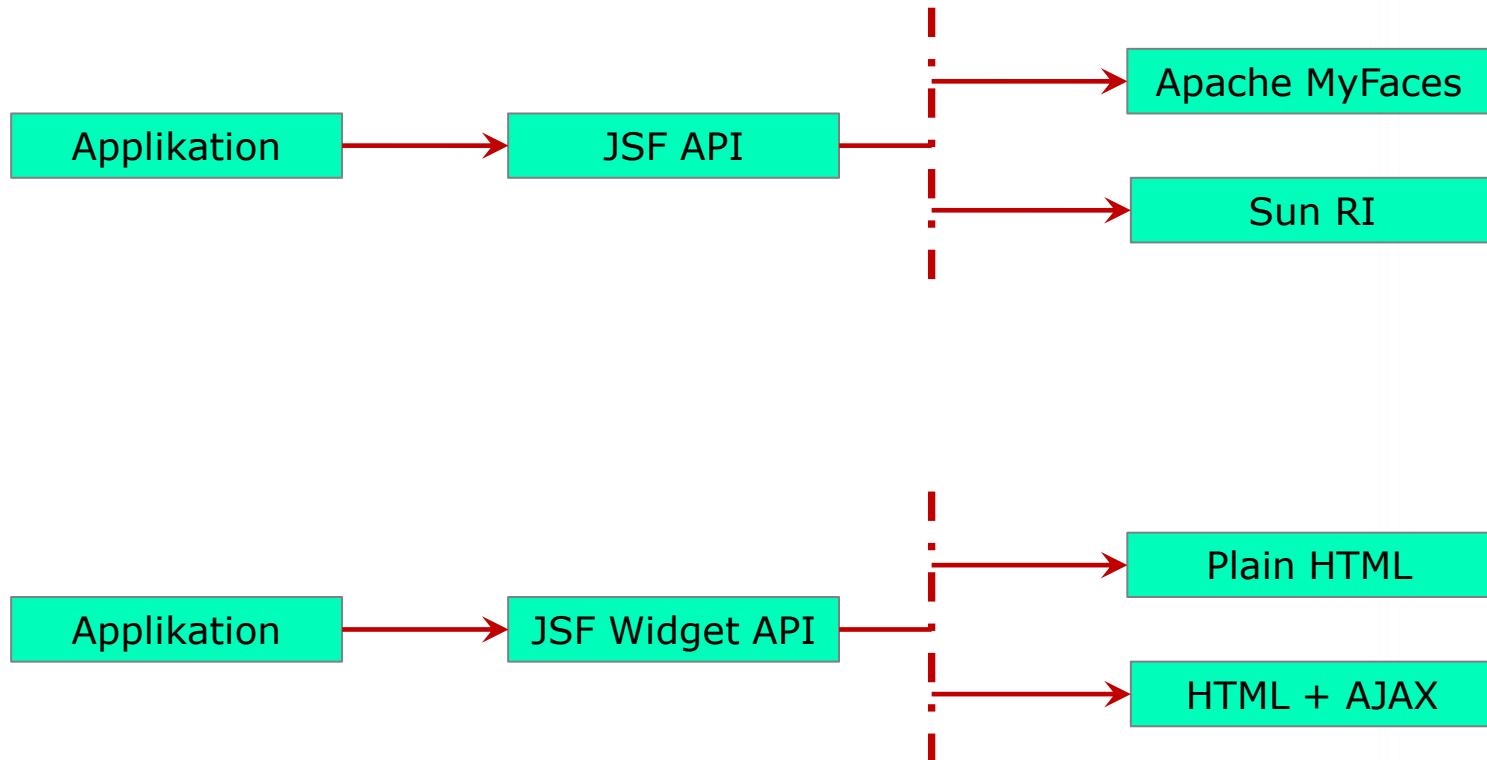
- State Manager
- Variable & Property Resolver
- Navigation & View Handler
- Error Handler

### ■ Framework

- Application Factory
- Exception Handler Factory
- External Context Factory
- Faces Context Factory
- Lifecycle Factory
- Partial View Context Factory
- RenderKit Factory
- Tag Handler Delegate Factory
- View Declaration Language Factory
- Visit Context Factory

***- Convention over Configuration -***

### Schnittstelle und Implementierung



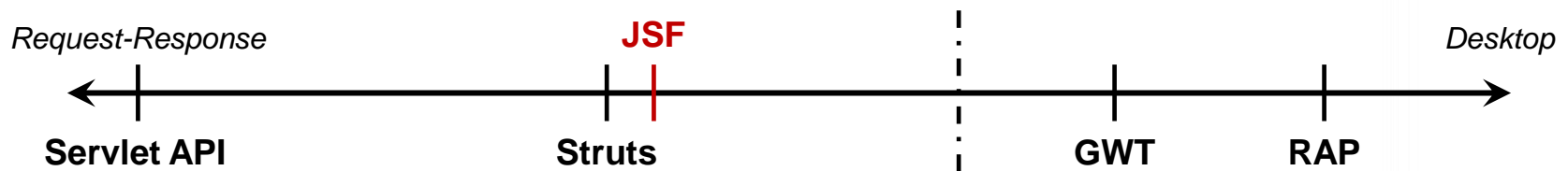
### Pluggability

- Vorbild: Servlet API
- Konfiguration von Metadaten
  - Datei `WEB-INF/faces-config.xml`
  - Annotationen
- Erweiterung in externen Bibliotheken
  - Datei `META-INF/faces-config.xml`
  - Annotationen
- Automatische Konfiguration der Applikation / des Frameworks durch Komponentenbibliothek bzw. JSF-Implementierung

### Entwicklungsgeschichte



### Entwicklermodell – Vergleich mit weiteren Web-Frameworks



### **JSF eignet sich besonders für ...**

- Komplexe Benutzeroberflächen mit Benutzerinteraktionen
- „Applikationen“

### **JSF eignet sich weniger für ...**

- Einfache Webseiten zur Informationsdarstellung

### **Kritisch betrachtet werden sollte ...**

- UI State Saving
  - HTTP Sessions oder Cookies mit Komponentenbäumen
  - Ggf. >1 Seite in Session (History, konfigurierbar)
- Barrierefreiheit (v.a. Javascript, Überschriften)
  - Auswahl der Komponentenbibliothek
  - Ggf. Austausch des RenderKit



- Was ist JSF und in welcher Beziehung steht JSF zur Servlet API?
- Welche Funktion hat die Phase *Restore View*?
- Welche Besonderheiten zeichnen JSF aus?