webforJ

Java-Modernisierung mit webforJ

effizient & zukunftssicher

Über mich...

webforJ

Stephan Wald
Head of Development

BASIS International Ltd.
Albuquerque, NM (USA)

EU-Standort: Saarbrücken

Tools und Frameworks für
Professionelle Businessanwendungen seit 1985

Do It All In Java
Write in Java, run on the Web

“Wer keine alten Handtücher hat, der hat auch keine neuen.”

Spruch meiner Oma

Die alten Handtücher in der IT…

webforJ

Unsere Legacy-Systeme
oft sowas alte Handtücher:

Code, den niemand anfassen möchte.

Es geht aber darum, das Alte zu respektieren,
und das Neue mit Bedacht einzuführen
– und vor allem, beide miteinander zu verbinden.

Alte Handtücher trocknen besser ab!

Die vier zentralen Herausforderungen

webforJ

- **Legacy Java-Systeme (z.B. Swing):**
  Technologische Isolation - nur Desktop, kein Browser-Zugriff
- **“Shiny Object Syndrome”:**
  Verlockung neuer Web-Frameworks vs. bewährte Java-Skills
- **Team-Motivation:**
  Java-Expertise respektieren, ohne Web-Modernisierung zu fürchten
- **Zukunftsfähigkeit:**
  Legacy-Wartung wird immer teurer und riskanter

Die Risiken des Legacy-Stillstands

webforJ

- **Steigende Kosten:**
  Legacy-Spezialisten werden rar und teuer
- **Sicherheitsrisiken:**
  Veraltete Client-Installationen, schwierige Updates
- **Innovationsbremse:**
  Keine mobile Nutzung, eingeschränkte Deployment-Optionen

Je länger wir bei Legacy-Systemen bleiben,
desto größer wird das Problem!

webforJ

Wie Sie Ihr Java-System mit webforJ klug in die Zukunft führen können

Was ist webforJ?

- **Java UI-Framework**
  für moderne Web-Anwendungen
- **Kernprinzip:**
  Komplette Entwicklung in Java - keine REST-APIs nötig
- **Zielgruppe:**
  Professionelle Business-Anwendungen
- **Legacy-Migration:**
  Bewährter Weg von Java-Desktop zu Web ohne JavaScript

Komplexität – gewollt und ungewollt

webforJ

Essential

Nicht vermeidbare Komplexität, die erforderlich ist, um ein Problem zu lösen oder das Produkt zu entwickeln.

Accidental

Von Entwicklern **ungewollt** erzeugte Komplexität. Befördert durch die **unüberschaubare Zahl** möglicher Werkzeuge und Wege.

Traditioneller Ansatz erzeugt Accidental Complexity

webforJ

- **Frontend:**
  React/Angular (JavaScript)
- **Backend:**
  Java, ggfs. Spring (Boot) o.ä.
- **Integration:**
  REST-APIs, JSON

- **Teams:**
  Frontend- und
  Backend-Spezialisten

- **-> Problem:**
  Koordinationsaufwand,
  Doppelte Wartung

webforJ eliminiert Accidental Complexity

- **Alles in Java:**
  UI + Business Logic
- **Direkte Methodenaufrufe**
  statt API-Calls
- **Ein Team, eine Sprache, ein Deployment**

Vorteil: 60% weniger Code, 40% schnellere Entwicklung

webforJ

Weniger Teams
Mehr Output

Teamgröße-Vorteil

webforJ

Weniger Technologien

= weniger und kleinere Teams

- **10 Entwickler traditionell:**

  5 Frontend + 5 Backend
- **7 Entwickler webforJ:**

  Vollständige Java-Teams

Teamgröße-Vorteil

webforJ

Weniger Technologien

= weniger un

- **10 Entwi**
  5 Fronte
- **7 Entwic**
  Vollstän

Kommunikations-Effizienz

- **Weniger Entwickler**
  kommunizieren effektiver miteinander
- **Weniger Themen:**
  Keine Frontend/Backend-Abstimmung
- **Ein Stack:**
  Alle sprechen "Java"

Teamgröße-Vorteil

webforJ

Weniger Technologien

= weni

- **1**
  5
- **7**
  V

Kommunikations-Effizienz

miteinander

Abstimmung

Team-Kohäsion:

- **Einfacheres Management:**
  Ein Team, ein Lead, ein Ziel
- **Weniger Fingerpointing:**
  Keine Schuldzuweisungen zwischen Teams
- **Weniger Stress:**
  Klare Verantwortlichkeiten

Teamgröße-Vorteil

webforJ

Weniger Technologien

= weniger und kleinere Teams

- **10 Entwickler traditionell:**
  5 Frontend + 5 Backend
- **7 Entwickler webforJ:**
  Vollständige Java-Teams

Kommunikations-Effizienz

- **Weniger Entwickler**
  kommunizieren effektiver miteinander
- **Weniger Themen:**
  Keine Frontend/Backend-Abstimmung
- **Ein Stack:** Alle sprechen "Java"

Team-Kohäsion:

- **Einfacheres Management:**
  Ein Team, ein Lead, ein Ziel
- **Weniger Fingerpointing:**
  Keine Schuldzuweisungen zwischen Teams
- **Weniger Stress:**
  Klare Verantwortlichkeiten

Nachhaltige Migration

Die flexible Modernisierungs-Journey

webforJ

Option A:

Direkt-Migration (Legacy Java → webforJ)

- Direkt von Java-Desktop zu webforJ Web
- Für Teams mit verfügbaren Ressourcen

Die flexible Modernisierungs-Journey

webforJ

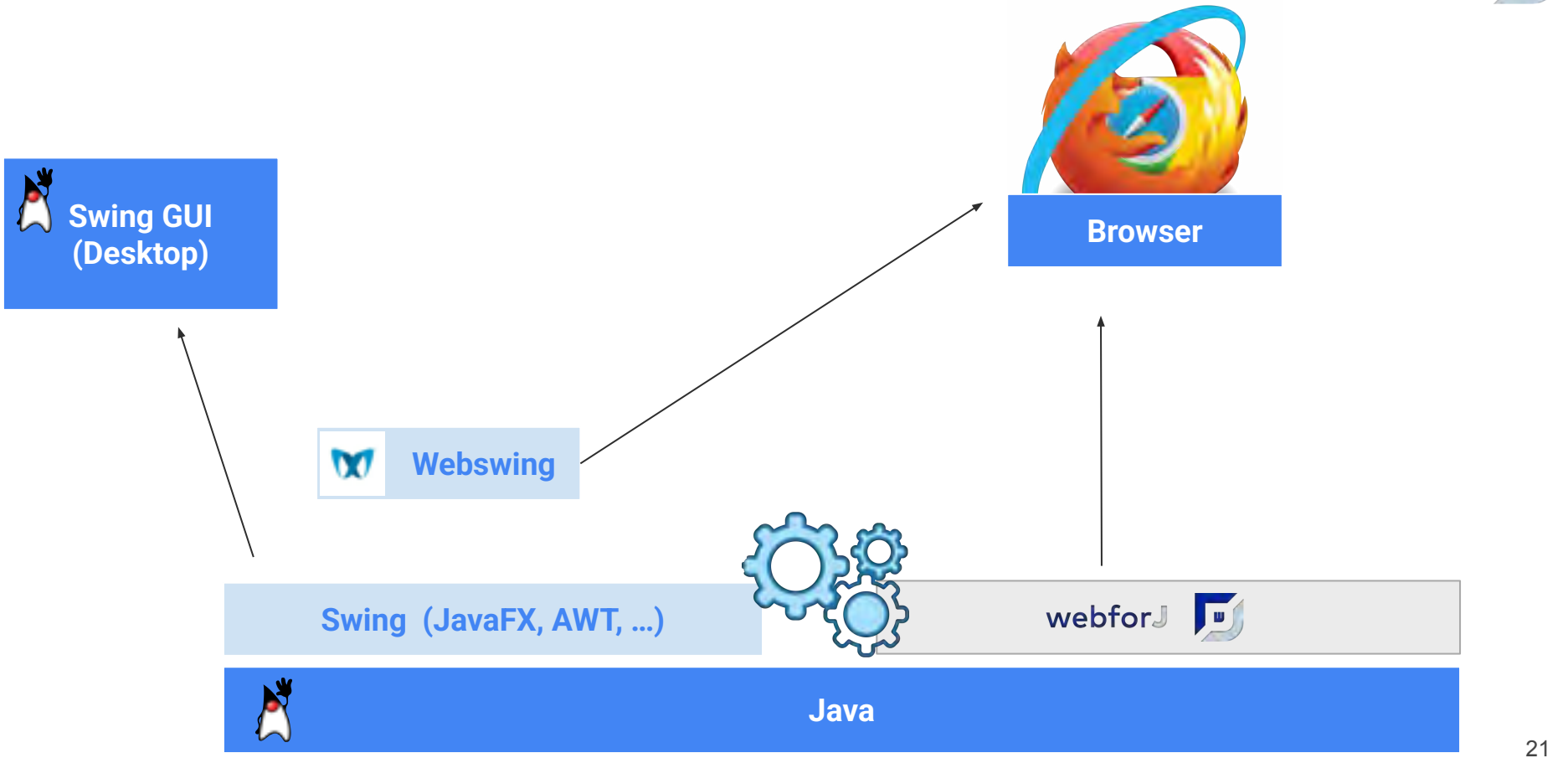
Option B:

Stufenweiser Journey (für kritische Systeme)

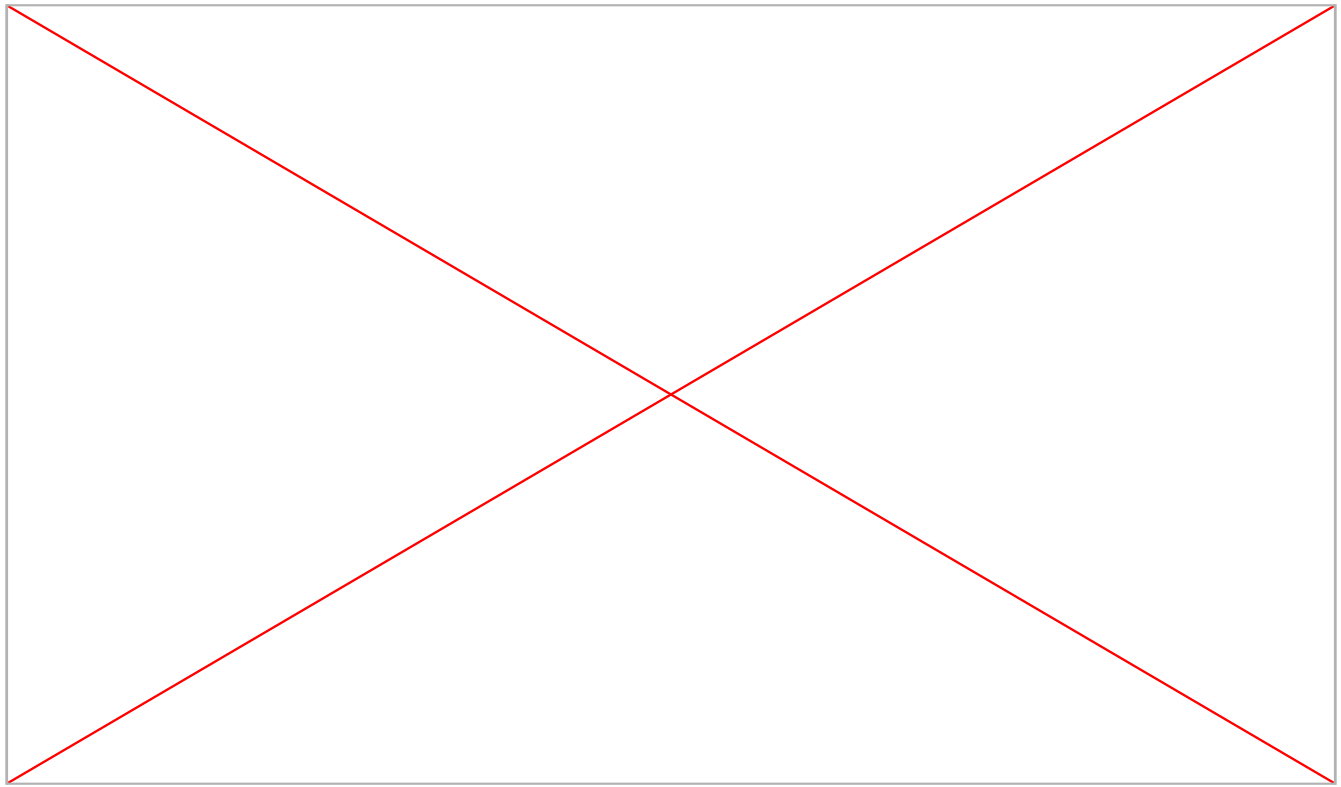
- **Webswing (bei Swing-Apps):**
  Legacy-App sofort im Browser (Wochen)
- **Modernisierung:**
  Schrittweise UI-Erneuerung mit webforJ (Monate)
- **Optimierung:**
  Vollständige webforJ-Anwendung (Jahr…)

Die flexible Modernisierungs-Journey

webforJ



Webswing und webforJ kombiniert



webforJ

Praxisbeispiel

Beispiel Prodin BV

webforJ

Prodin
Business Solutions

- Prodin BV bietet ein ERP System für ausgewählte Zielbranchen

- Kunden in den Niederlanden, Belgien, Deutschland, USA und Kanada

Beispiel Prodin BV

webforJ

- Ausgangslage:
  - SWING GUI
  - Logik: Java ERM Framework
- Ab 2016: Suche nach Modernisierungs-Pfad
  - Versuche mit Low Code Plattformen

    -> gescheitert, zu unflexibel
  - Versuche mit Angular-Frontend

    -> gescheitert, passt nicht zur komplexen vorhandenen Backend-Logik + Technologie-Bruch im Team

Beispiel Prodin BV

webforJ

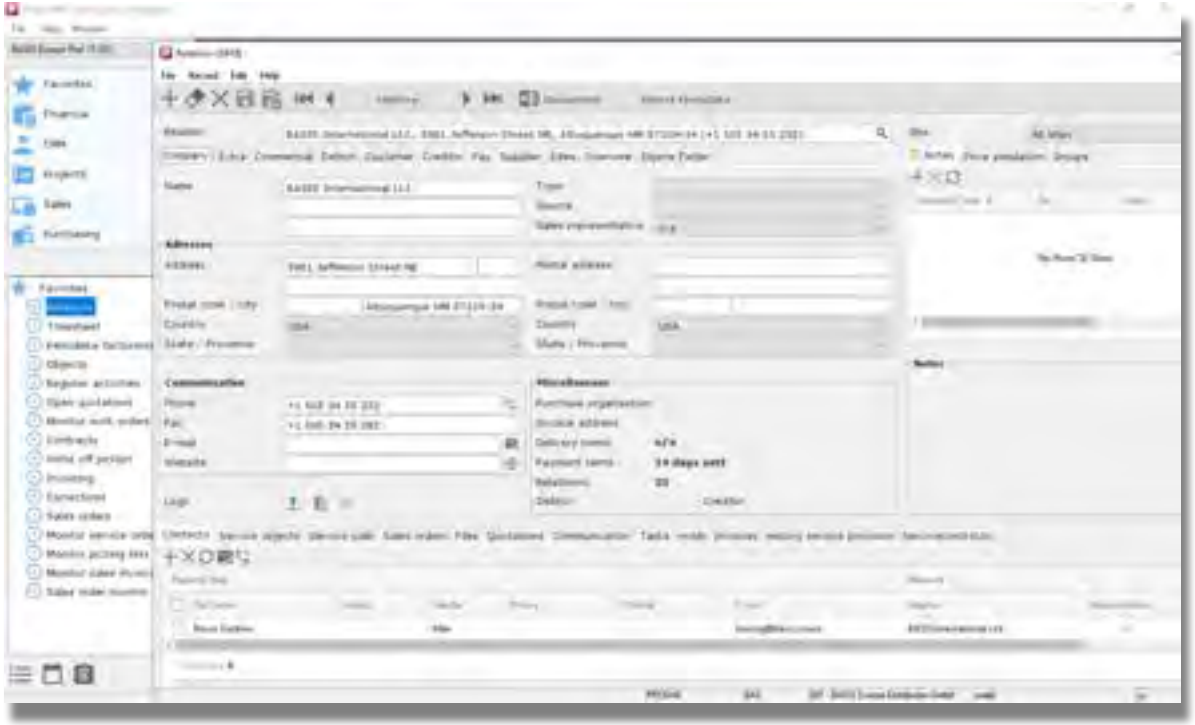
Seit 2022: konsequente Modernisierung der Oberfläche mit webforJ:

- Alles bleibt Java
- Bestehende Logik kann sehr weitgehend übernommen werden
- Bestehendes Team kann effektiv an der Modernisierung mitwirken
- Neue Team-Mitglieder sind früher wirksam
- Rollout des Facelift in kleinen, verdaulichen Schritten
- Alte und neue Screens kombinierbar

=> Geringeres Risiko, geringere Kosten, schneller sichtbare Erfolge.

Beispiel Prodin BV

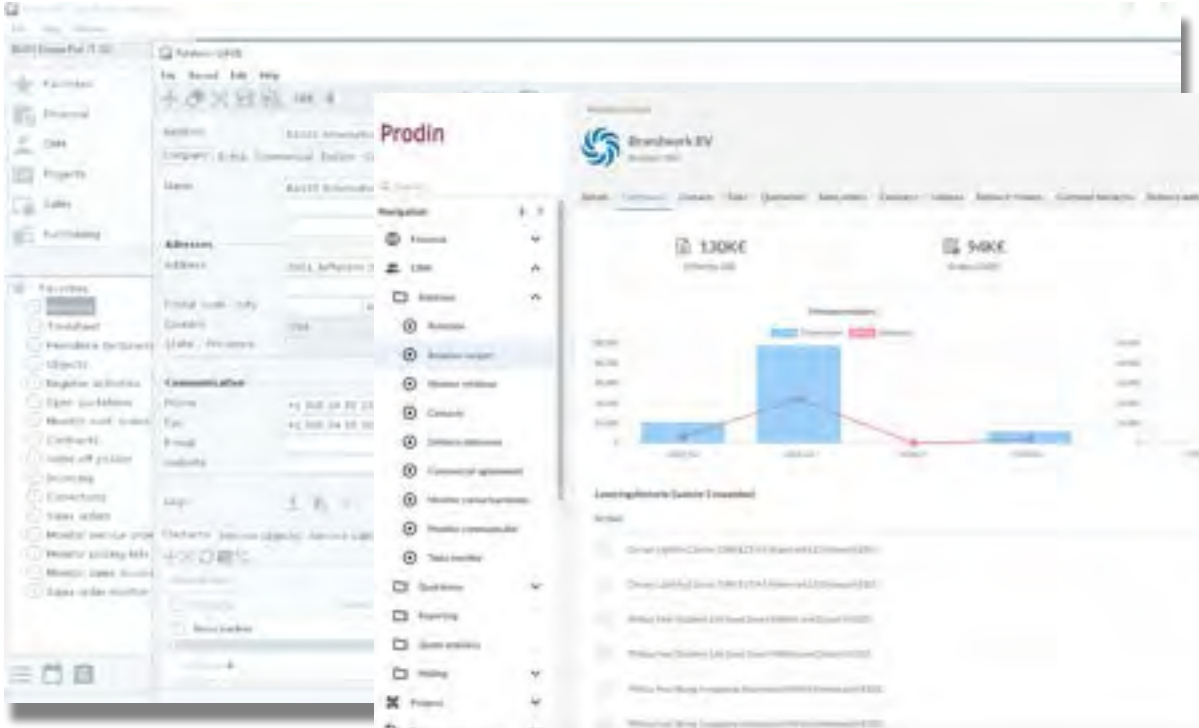
webforJ



vorher

Beispiel Prodin BV

webforJ



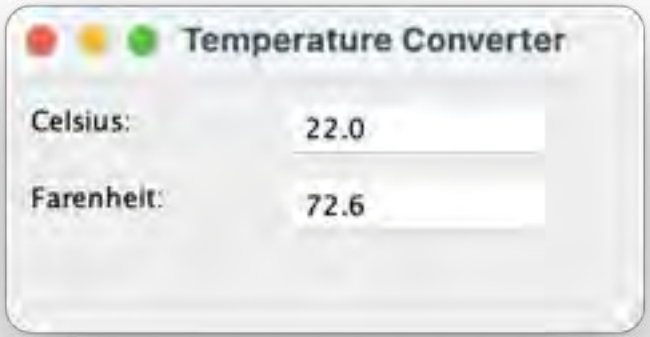
vorher

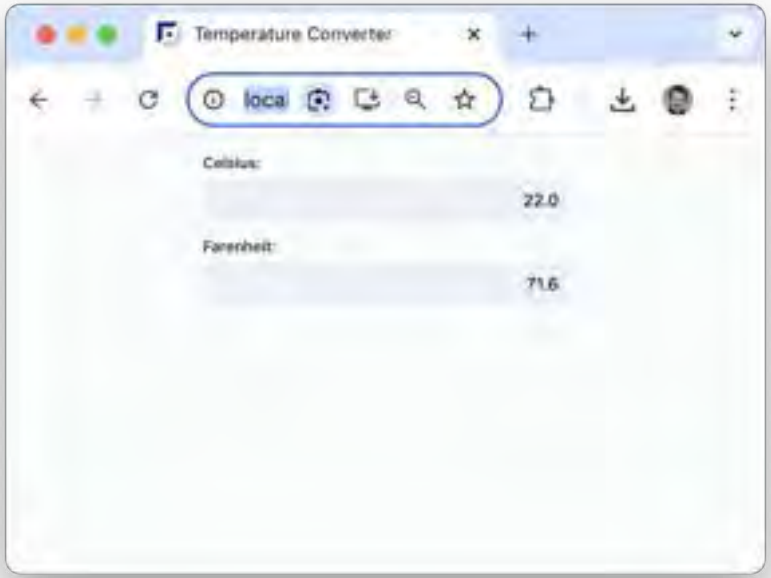
nachher

Wie komplex muss ein einfaches Formular sein?

Wie komplex muss ein einfaches Formular sein?

webforJ





Wie komplex muss ein einfaches Formular sein?

webforJ

Backend: Spring Boot

```
temperature-converter-backend
├── src
│   ├── main
│   │   ├── java/com/example/converter
│   │   │   ├── model
│   │   │   │   └── TemperatureConverter.java
│   │   │   ├── controller
│   │   │   │   └── TemperatureController.java
│   │   │   └── Application.java
│   │   └── resources
│   │       └── application.properties
```

Frontend: Angular

```
temperature-converter
├── src
│   ├── app
│   │   ├── converter
│   │   │   ├── converter.component.ts
│   │   │   ├── converter.component.html
│   │   │   └── converter.component.css
│   │   ├── services
│   │   │   └── temperature.service.ts
│   │   ├── app.module.ts
│   │   └── app.component.ts
```

Wie komplex muss ein einfaches Formular sein?

webforJ

Spring Boot Backend				Angular Frontend		
File	Sprache	LOC		File	Sprache	LOC
`TemperatureConverter.java`	Java	9		`temperature.service.ts`	TypeScript	14
`TemperatureController.java`	Java	16		`converter.component.ts`	TypeScript	20
`Application.java`	Java	7		`converter.component.html`	HTML	14
`application.properties`	Properties	0		`converter.component.css`	CSS	9
				`app.module.ts`	TypeScript	12
				`app.component.ts (Auto-generated)`	TypeScript	5
				`index.html (Main HTML file)`	HTML	10
				`styles.css (Global styles)`	CSS	5
Total (Backend)		32		Total (Frontend)		89

Antwort: ~ 120 Zeilen Quellcode, und 4 Sprachen

Wie komplex muss ein einfaches Formular sein?

webforJ

Single Stack Framework

```
package org.example.model;

pu
 pu

 }

 pu

 }
}
```

```
package org.example.views;

import…

@Route("/")
public class TemperatureView extends Composite<FlexLayout> {

 private FlexLayout self = getBoundComponent();
 private MaskedNumberField celsius = new MaskedNumberField("Celsius:");
 private MaskedNumberField farenheit = new MaskedNumberField("Farenheit:");
 private TemperatureConverter model = new TemperatureConverter();

 public TemperatureView(){

   self.setDirection(FlexDirection.COLUMN).setMaxWidth(300).setStyle("margin", "1em auto");

   celsius.setMask("-###0.0");
   farenheit.setMask("-###0.0");

   self.add(celsius, farenheit);
```

```
public class TemperatureView extends Composite<FlexLayout> {

 private FlexLayout self = getBoundComponent();
 private MaskedNumberField celsius = new MaskedNumberField("Celsius:");
 private MaskedNumberField farenheit = new MaskedNumberField("Farenheit:");
 private TemperatureConverter model = new TemperatureConverter();

 public TemperatureView(){

   self.setDirection(FlexDirection.COLUMN)
         .setMaxWidth(300).setStyle("margin", "1em auto");

   celsius.setMask("-###0.0");
   farenheit.setMask("-###0.0");

   self.add(celsius, farenheit);

   celsius.onModify(modifyEvent ->
         farenheit.setValue(model.celsiusToFahrenheit(celsius.getValue())));
   farenheit.onModify(modifyEvent ->
         celsius.setValue(model.fahrenheitToCelsius(farenheit.getValue())));
 }
}
```

Wie komplex muss ein einfaches Formular sein?

webforJ

Single Stack Java Framework (webforJ)		
File	Sprache	LOC
`TemperatureConverter.java`	Java	9
`TemperatureView.java`	Java	14
`Application.java (Auto-generated)`	Java	6
Total (Frontend + Backend)		29

~~Antwort: ~ 120 Zeilen Quellcode, und 4 Sprachen~~

Antwort: Weniger als 30 Zeilen und nur eine Sprache: Java

Komplexitäts-Breakdown

webforJ

Java + JS Framework UI

❌ Frontend & Backend getrennt

❌ Extra API Integration

❌ JavaScript + JSON

❌ Zwei verschiedene Expertisen

❌ Debugging:
Getrennt für UI und Backend

webforJ Single Stack Java Framework

✅ Ein Stack, eine Sprache

✅ Direkte Methodenaufrufe

✅ Keine Formatkonvertierung

✅ Alle Java Developer

✅ Debugging:
Breakpoint setzen und beides
UI und Backend in der IDE debuggen

webforJ

>60% weniger Code =
400.000 € Einsparung

Die Mathematik der Einsparungen

webforJ

Konkrete Einsparungen bei einem 10-Mitarbeiter-Team:

- **webforJ-Produktivitätsgewinne:**
  60% weniger Code = 60% weniger Entwicklungszeit
- **Nur die Hälfte der Tech-Stacks**
  50% weniger Koordinationsaufwand
- **40% schnellere Entwicklung**
  (Siehe Prodin-Beispiel)

Die Mathematik der Einsparungen

webforJ

Split-Stack Team (10 Entwickler):

- **5 Frontend-Entwickler (React/Angular):**
  70.000 € × 5 = 350.000 €
- **5 Backend-Entwickler (Java):**
  75.000 € × 5 = 375.000 €
- **Koordinationsaufwand:**
  10% = 72.500 €

TOTAL: 797.500 € pro Jahr

webforJ-Team (7 Java-Entwickler):

- **7 Java-Entwickler (Full-Stack):**
  75.000 € × 7 = 525.000 €
- **Koordinationsaufwand:**
  minimal = 15.000 €

TOTAL: 540.000 € pro Jahr

JÄHRLICHE EINSPARUNG: 257.500 €

Zusätzliche versteckte Einsparungen:

- **Wartungsaufwand:**
  -50% = 50.000 €/Jahr
- **Schulungskosten:**
  -70% = 25.000 €/Jahr
- **Deployment-Vereinfachung:**
  -60% = 20.000 €/Jahr
- **Integration-Bugs:**
  -80% = 30.000 €/Jahr

GESAMTEINSPARUNG:
über 400.000 € pro Jahr!

Die Mathematik der Einsparungen

webforJ

Beispiel - Kostenvergleich für 3-Jahres-Projekt (10 Entwickler):

Ansatz	Entwicklungskosten	Wartungskosten	Total 3 Jahre	Einsparung
Angular + Spring	2.4 Mio €	600.000 €	**3.0 Mio €**	-
Low-Code	1.8 Mio €	900.000 €	**2.7 Mio €**	300.000 €
Complete Rewrite	3.6 Mio €	450.000 €	**4.05 Mio €**	-1.05 Mio €
webforJ	**1.6 Mio €**	**300.000 €**	**1.9 Mio €**	**1.1 Mio €**

Business Case Prodin BV

webforJ

- **Time To Market:**
  Erste Module 4 Monate statt 12+ Monate bei Angular-Ansatz
- **8 Monate frühere Markteinführung**
  = Wettbewerbsvorteil unbezahlbar
- **Schnellere Feature-Iteration durch**
  Single-Stack-Entwicklung

Business Case Prodin BV

webforJ

Real-World Beweis:

- **6 Entwickler statt 10:**
  40% Teameinsparung
- **4 Monate statt 12+ Monate:**
  66% Zeit-Einsparung
- **20+ Jahre alte Logik wiederverwendet:**
  80% Code-Schutz
- **Alte und neue Screens parallel ausgerollt:**
  Zero-Risk-Migration

webforJ

Ihr Weg zu 400.000 € Einsparung

Ihr Weg zu 400.000 € Einsparung

Für Teams mit 5+ Java-Entwicklern:

- **Diese Woche:**
  ROI-Kalkulation für Ihr Team
- **Nächsten Monat:**
  Pilot-Projekt mit einem Modul starten
- **Quartal 1:**
  Erste produktive webforJ-Anwendung live

Erste Einsparungen bereits nach 3 Monaten sichtbar

Risiko-freier Start

webforJ

- **webforJ ausprobieren :**
  Kostenlos (auch produktiv!), keine Vendor-Bindung
- **Proof of Concept:**
  2-4 Wochen für ersten Prototyp
- **Parallelbetrieb:**
  Alte und neue Systeme kombinierbar
- **Schrittweise Migration:**
  Business Continuity garantiert

Kontakt & Ressourcen

webforJ

- **Heute nach dem Vortrag:**

  Direkte am Stand
- **Online-Demo und Ressourcen:**

  https://webforj.com
- **Prodin Case Study:**

  Vollständiger Report zum Download verfügbar

webforJ
Fangen Sie heute an!
400.000 € pro Jahr sparen
beginnt mit dem ersten Schritt.