

Web Frontends mit Angular 7

von Bernd Hort, 26.03.2019

EntwicklerCamp



Komplexe, wartbare Anwendungen

„Mit großer Kraft kommt
große Verantwortung.“





Angular in a Nutshell

Angular

- Angular ist eine Plattform zum Erstellen von Anwendungen, die auf Web-Standards basieren
 - Single Page Application
 - Auf Performance optimiert
 - Unterstützung für PWA - Progressive Web Apps

Angular Voraussetzungen

- Node.js
 - Version 8.x oder 10.x
 - Download von nodejs.org
- npm package manager
 - Wird installiert mit Node.js



Angular installieren

- Angular basiert auf dem Angular CLI (Command Line Interface)
- Installieren mit

```
$npm install -g @angular/cli
```



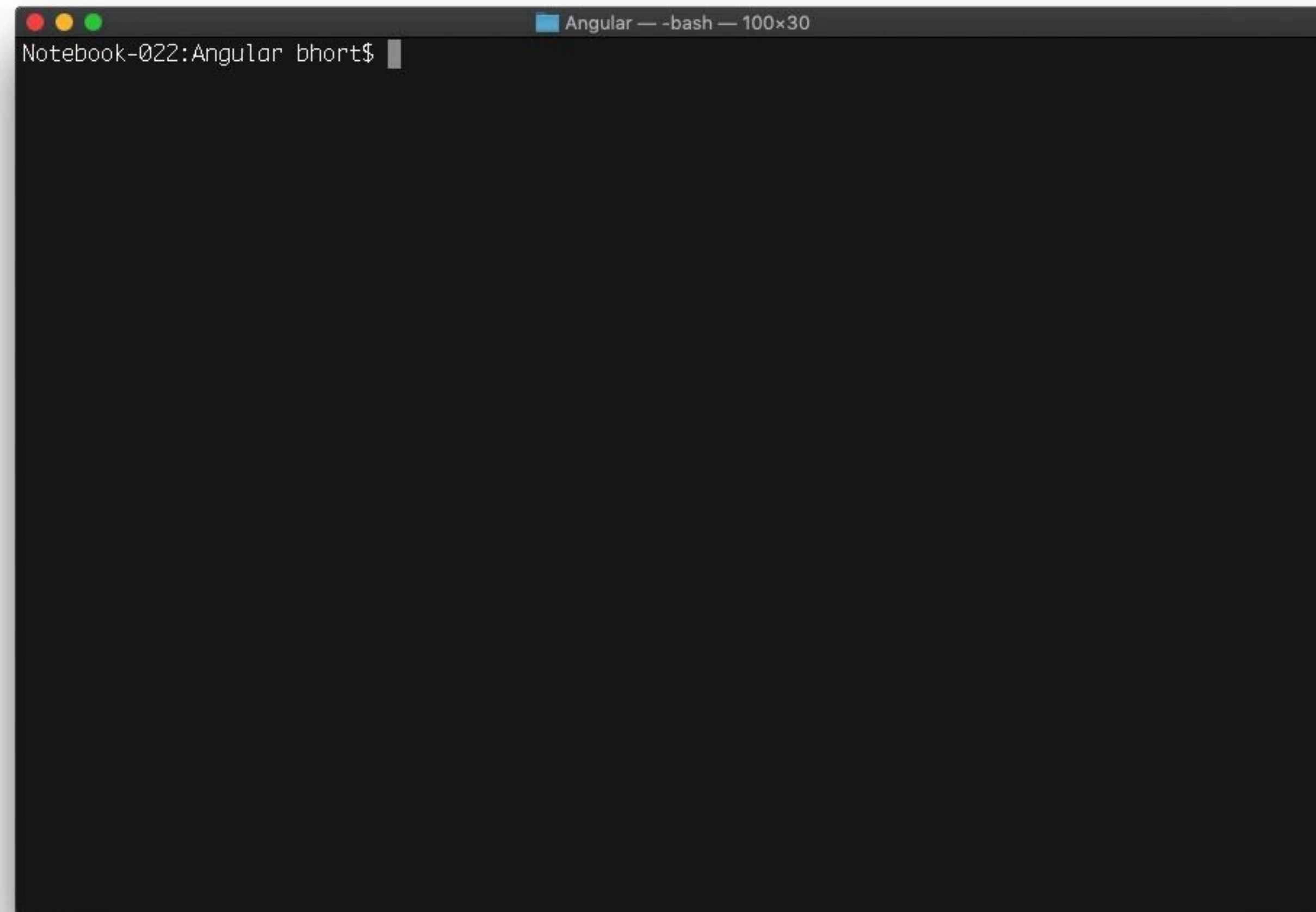
Neues Angular Projekt

- Neues Projekt anlegen mit

```
$ng new {Projektname}
```

- Erstellt automatisch Ordnerstruktur
- Initiiert ein Git-Repository
 - Vordefinierte Datei .gitignore
 - Check alle „sinnvollen“ Dateien ein

Neues Angular Projekt



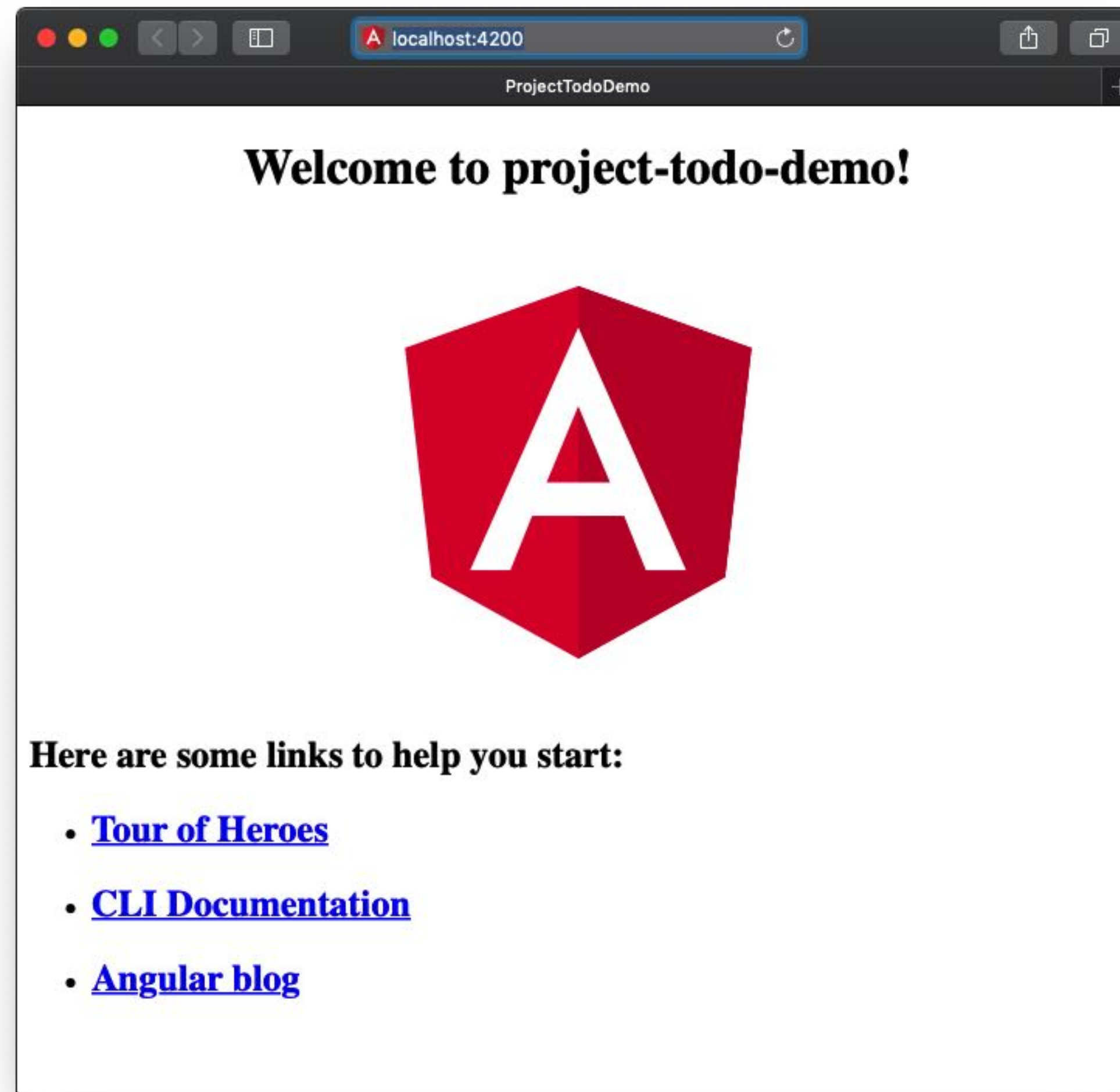
Neues Angular Projekt

- Live-Server starten

```
$npm start
```

- Compiled TypeScript
- Started lokalen Web-Server auf Port 4200
- Öffnen der Seite mit `http://localhost:4200`

Neues Angular Projekt



A collage of business-related items. In the foreground, a tablet displays a presentation titled "MARKETING RESEARCH". The presentation includes a bar chart on the left and two pie charts, labeled "FIGURE 2" and "FIGURE 3". "FIGURE 2" is a pie chart with segments of 83%, 15%, and 2%. "FIGURE 3" is a pie chart with segments of 35%, 22%, 10%, 15%, 6%, and 1%. To the right of the tablet, a black calculator is partially visible. A silver pen rests on a financial spreadsheet in the background, which contains various numerical data points and formulas. The overall scene suggests a professional or academic setting focused on data analysis and business research.

Componenten

„Mehr als die Summe der Teile“

Componenten

„Mehr als die Summe der Teile“

Components

- Aufteilung der Anwendung auf Komponenten und Module
- Komponenten bestehen aus
 - HTML
 - CSS
 - TypeScript (Logic)
 - TypeScript (Test-Code)
- Inhaltliche zusammenhängende Komponenten bilden Module
- Sinnvoll für „Lazy Loading“ und Wiederverwendbarkeit



Komponente erzeugen

- Angular CLI wird für das Erstellen von Anwendungsbestandteilen verwendet

```
$ng generate [Schemaname]
```



- Eine Komponente erstellen

```
$ng g component [Componentname]
```



Direktiven

Verbindung zwischen
HTML and TypeScript

Direktiven

- Erweiterungen von HTML, um Angular spezifische Ausdrücke
 - Eigene HTML-Elemente
 - Bestimmung von Attributen
 - Zugriff auf Inhalte aus TypeScript
 - Schleifen, If-Abfragen und ähnliches
- Angular Cheat Sheet

Dependencies Injection

Abhängigkeiten
identifizieren als
Voraussetzung für
Performance



Dependencies Injection

- Definition von Abhängigkeiten in der app.module.ts Datei
 - Import-Statements
 - Definition der Verwendung im @NgModule({imports: ... })
- Module definieren ihre eigenen Abhängigkeiten
 - Import-Statements für das Modul
 - Verwendung und Bereitstellung in @NgModule({imports: ... }, {exports: ...})



Bootstrap CSS einbinden

- Wir benötigen nur das CSS von Bootstrap

```
$npm install --save bootstrap-css-only
```

- In angular.json einbinden

```
"styles": [  
  "src/styles.css",  
  "node_modules/bootstrap-css-only/css/bootstrap.min.css"  
],
```

Bootstrap hinzufügen

- Für Bootstrap existiert ein Projekt ohne Abhängigkeit zu jQuery: ng-bootstrap

```
$npm install --save @ng-bootstrap/ng-bootstrap
```

- Lädt Bootstrap 4 herunter

```
npm WARN @angular/common@7.2.9 requires a peer of rxjs@^6.0.0 but none is installed. You must install peer dependencies yourself.  
npm WARN @angular/core@7.2.9 requires a peer of rxjs@^6.0.0 but none is installed. You must install peer dependencies yourself.  
npm WARN @angular/forms@7.2.9 requires a peer of rxjs@^6.0.0 but none is installed. You must install peer dependencies yourself.  
npm WARN @angular/router@7.2.9 requires a peer of rxjs@^6.0.0 but none is installed. You must install peer dependencies yourself.  
npm WARN @ng-bootstrap/ng-bootstrap@4.1.0 requires a peer of rxjs@^6.3.0 but none is installed. You must install peer dependencies yourself.
```

- Anschließend fehlende Abhängigkeiten installieren

```
$npm install
```

Bootstrap hinzufügen

- Wie auf der ng-bootstrap Projektseite beschrieben
- Datei src/app/app.module.ts anpassen
 - Import hinzufügen

```
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';
```

```
@NgModule({  
  ...  
  imports: [..., NgbModule],  
  ...  
})
```



Module

Aufteilung der Anwendung
in logische Bereiche

Module

- Module bilden logische Strukturen innerhalb der Anwendung ab
- Über eigene [name].module.ts-Datei Steuerung der Abhängigkeiten und nach Außen sichtbaren Komponenten, Services etc.
- Navigation innerhalb eines Modules über Routing definierbar
- Ein Module mit eigenem Routing anlegen

```
$ng g module [Modulname] --routing
```



Services

„Stets zu Diensten“

Services

- Services stellen Funktionalitäten für Module innerhalb einer Anwendung zur Verfügung
- Sie werden über Dependencies Injection eingebunden
- Beispiele sind
 - Zentrale Konfiguration
 - Services, um Daten zu liefern oder speichern

Reactive Programming

Asynchrone Verarbeitung



Reactive Programming

- Implementiert das Observer-Pattern
- Stream - eine Reihe von Ereignissen (Events)
- Observables - löst eine Reihe von Ereignissen (Events) aus, auf die reagiert werden soll
 - onNext - wird aufgerufen, wenn das nächste Ereignis eintritt
 - onError - wird im Fehlerfall aufgerufen
 - onCompleted - wird aufgerufen, wenn die Sequenz von Ereignissen endet



RxJS

Reactive Programming

- Observer – melden sich an einem Observable mit `subscribe()` an (Subscription)
- Wird ein Event ausgelöst, wird die entsprechende Methode aufgerufen
 - `next(value)` – wird mit einem neuen Wert aufgerufen
 - `error(errorObject)` – wird mit einem Fehlerobjekt aufgerufen
 - `complete()` – wird am Ende der Sequenz aufgerufen



RxJS

Reactive Programming

- Subject - implementiert sowohl Observable und Observer
 - gute Basis für Services



RxJS



Routing

Pfade innerhalb der
Anwendung definieren

Routing

- Routes definieren Pfade innerhalb der Anwendung
- Die Pfade sind wichtig, um Standard-Navigations-Methoden des Browsers zu unterstützen
 - Back-Button
 - Lesezeichen
 - Deep-Links
- Entweder global oder auf Modul-Ebene

Formulare



Formulare

- Template-Driven Forms
 - Two-way Databinding
 - AngularJS
 - Für einfache Formulare okay, aber nicht sehr performant
- Reactive Forms
 - HTML-Form und Model über Observable verbunden
 - Sehr performant und mächtig



HTTP-Client

Zugriff auf REST-Services

HTTP-Client-Module

- Zugriff auf REST-Services mit HTTP-Client-Modul
- Implementiert alle REST-Methoden
 - GET
 - POST
 - PUT
 - DELETE
- Verwendet Observables

#domino forever

Domino

Forever young!

Testen auf Login

- Prüfen, ob die Cookies `DomAuthSessId` oder `LtpaToken` vorhanden sind
- Verwendung des npm-Packages `ng2-cookies`
- Etablierung eines Guards für die Routen
 - Die Route darf nur „betreten“ werden, wenn die Cookies vorhanden sind

Testen HTTP-Zugriff gegen Domino

- Beim Zugriff auf Domino basierten REST-Services häufig Cross-Side-Scripting Fehler
- Abhilfe auf Server-Seite
 - Implementierung Cross-Origin Resource Sharing (CORS)
- Abhilfe für lokalen Entwicklungs-Server
 - Benutzung eines Proxy

Konfiguration Proxy (1)

- Anpassung package.json

```
{  
  "name": "project-todo-demo",  
  "version": "0.0.0",  
  "scripts": {  
    "ng": "ng",  
    "start": "ng serve --proxy-config proxy.conf.json",  
    "build": "ng build",  
    "test": "ng test",  
    "lint": "ng lint",  
    "e2e": "ng e2e"  
  }, ...  
}
```

Konfiguration Proxy (2)

- Erstellung proxy.conf.json

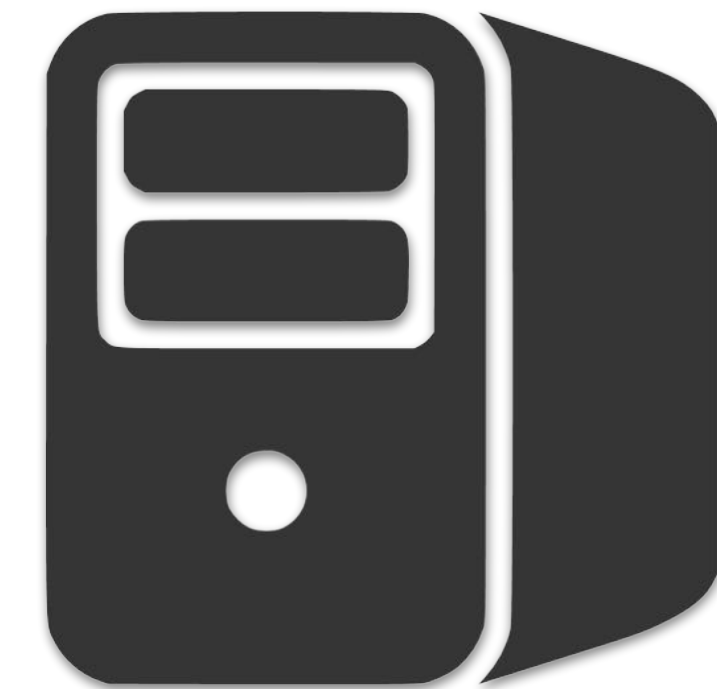
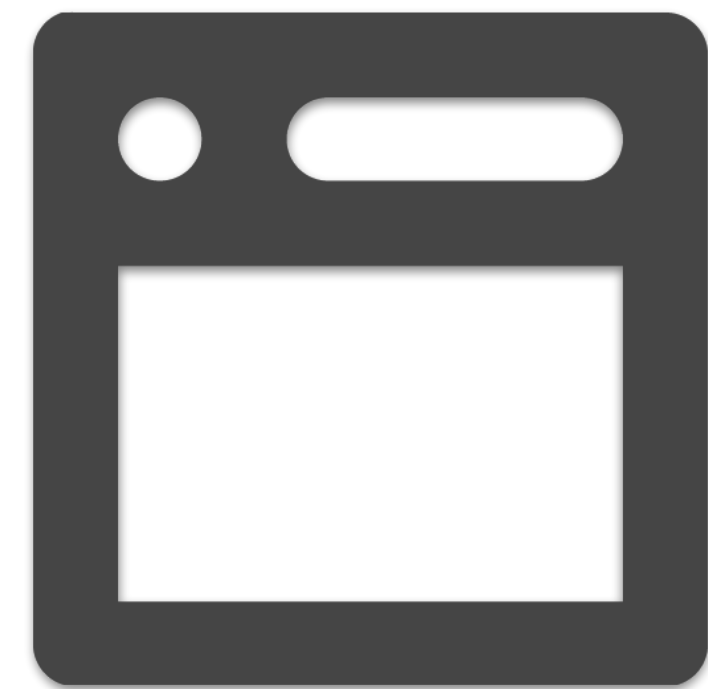
```
{
  "/Projekte/Konferenzen/todo.nsf/xsp/rest/": {
    "target": "http://bht9.assono.local",
    "secure": false,
    "logLevel": "debug"
  },
  "/names.nsf": {
    "target": "http://bht9.assono.local",
    "secure": false,
    "logLevel": "debug"
  }
}
```

Domino- Architektur

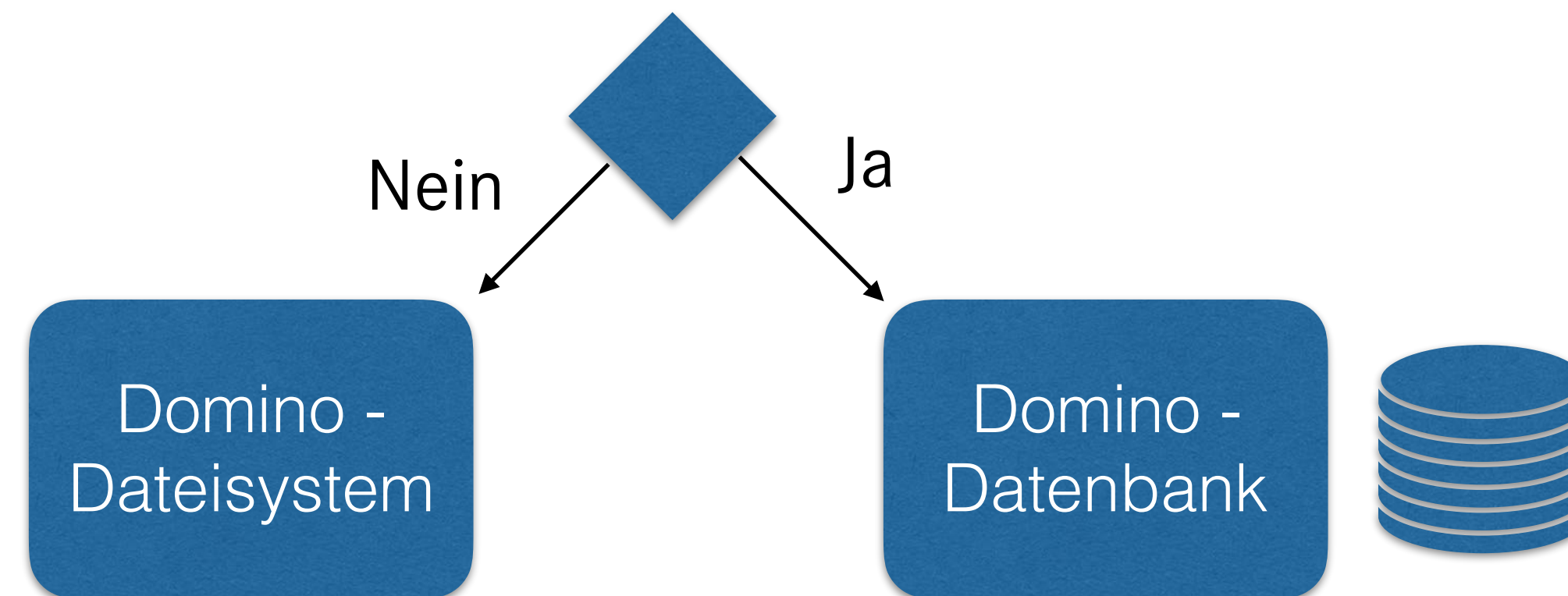
Angular Web App
in Notes Datenbank



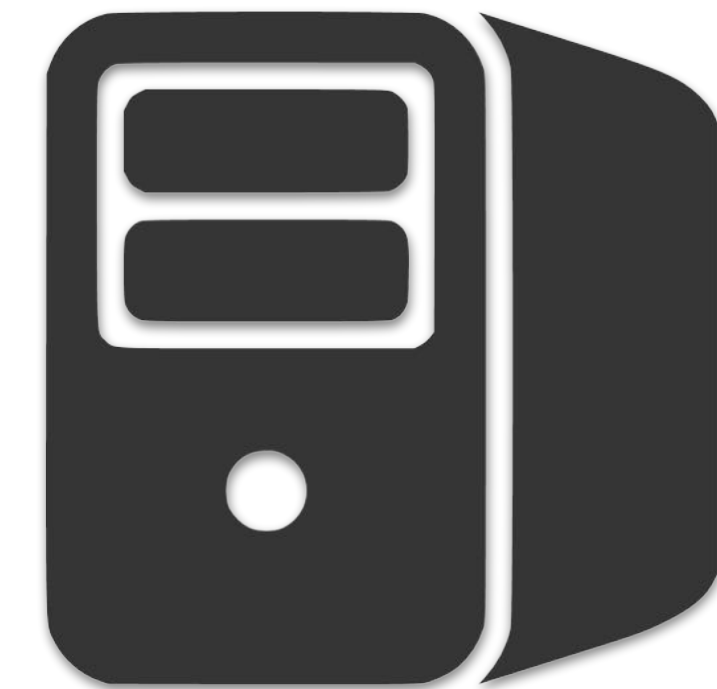
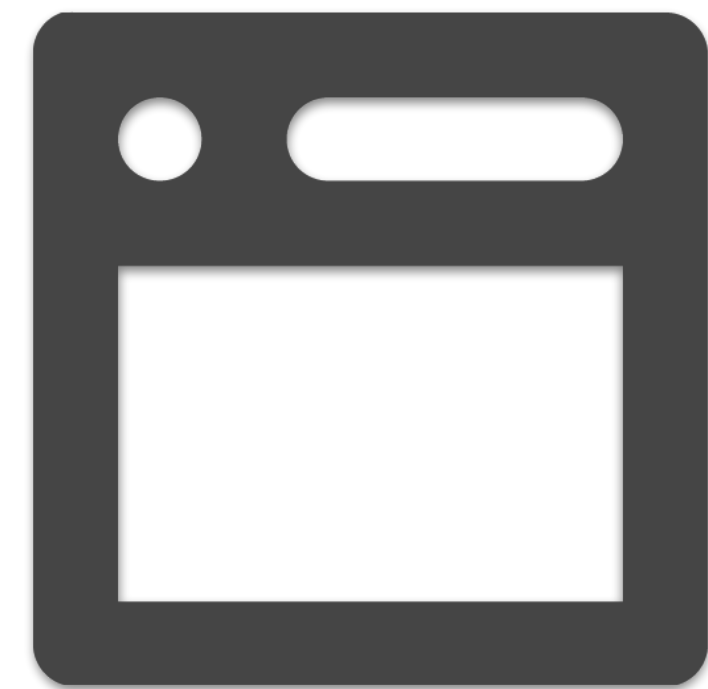
Verarbeitung von HTTP-Requests anhand der URL



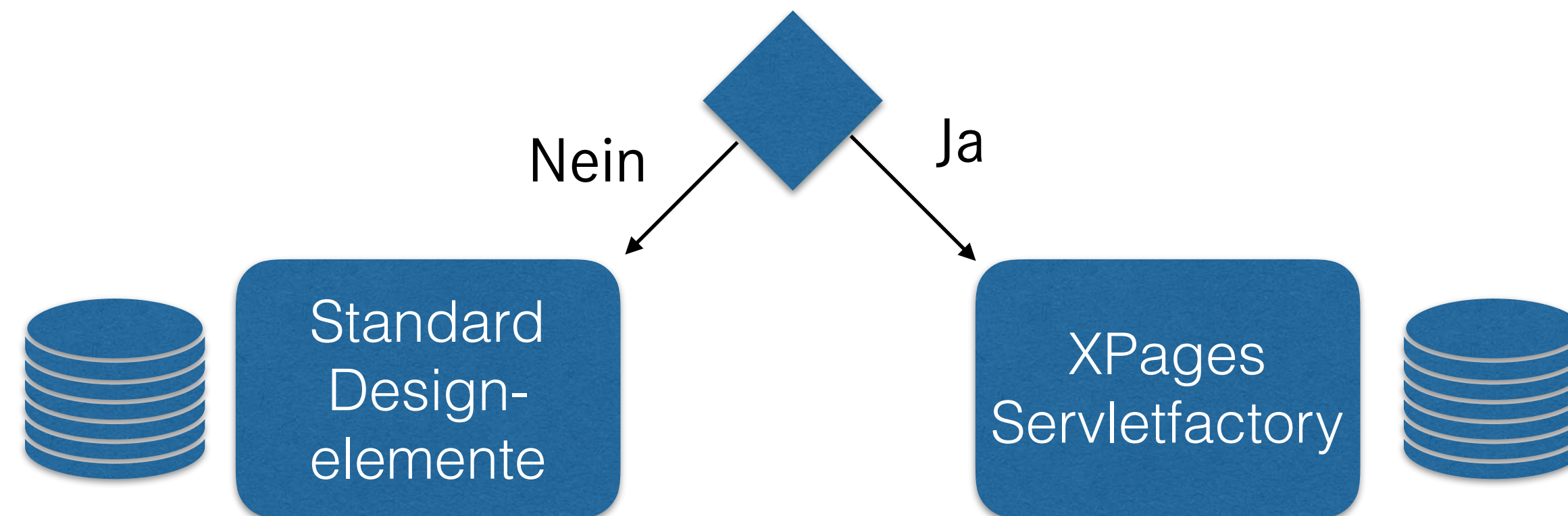
URL enthält *.nsf



Verarbeitung von HTTP-Requests anhand der URL

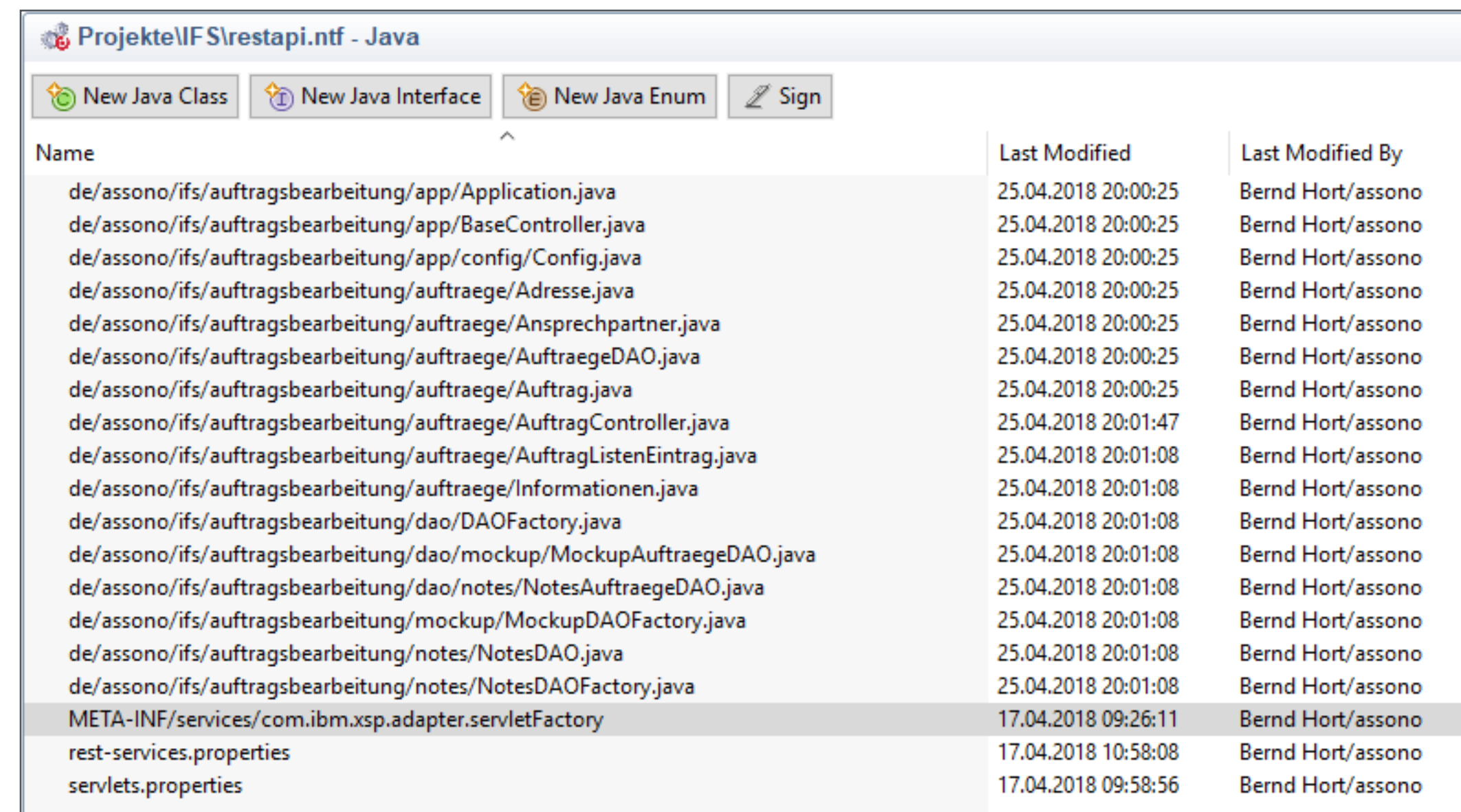


URL enthält *.nsf und xsp



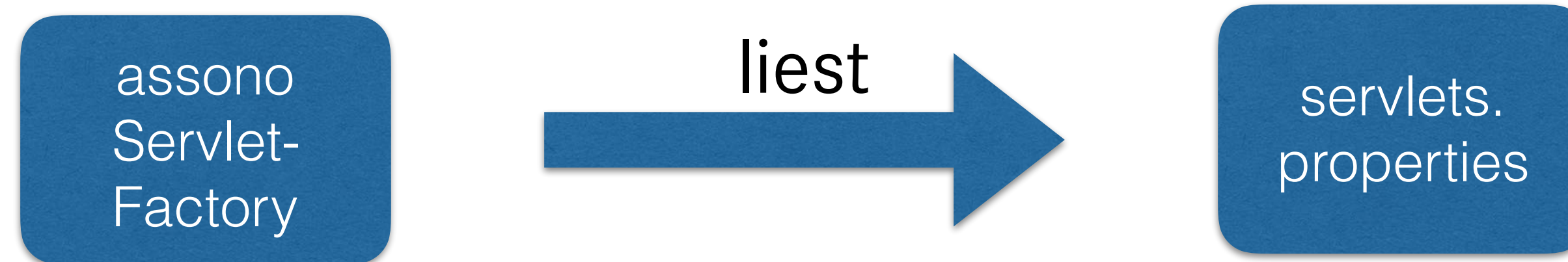
XPages Servletfactory

- Bei allen Anfragen, die „xsp“ in der URL haben, entscheidet die Servlet-Factory welches Servlet die Anfrage bedienen soll.
- Die Text-Datei „META-INF/services/com.ibm.xsp.adapter.servletFactory“ enthält den Klassennamen der eigentlichen Servlet-Factory
`de.assono.service.servlet.ServletFactory`



Name	Last Modified	Last Modified By
de/assono/ifs/auftragsbearbeitung/app/Application.java	25.04.2018 20:00:25	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/app/BaseController.java	25.04.2018 20:00:25	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/app/config/Config.java	25.04.2018 20:00:25	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/auftraege/Adresse.java	25.04.2018 20:00:25	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/auftraege/Ansprechpartner.java	25.04.2018 20:00:25	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/auftraege/AuftraegeDAO.java	25.04.2018 20:00:25	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/auftraege/Auftrag.java	25.04.2018 20:00:25	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/auftraege/AuftragController.java	25.04.2018 20:01:47	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/auftraege/AuftragListenEintrag.java	25.04.2018 20:01:08	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/auftraege/Informationen.java	25.04.2018 20:01:08	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/dao/DAOFactory.java	25.04.2018 20:01:08	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/dao/mockup/MockupAuftraegeDAO.java	25.04.2018 20:01:08	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/dao/notes/NotesAuftraegeDAO.java	25.04.2018 20:01:08	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/mockup/MockupDAOFactory.java	25.04.2018 20:01:08	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/notes/NotesDAO.java	25.04.2018 20:01:08	Bernd Hort/assono
de/assono/ifs/auftragsbearbeitung/notes/NotesDAOFactory.java	25.04.2018 20:01:08	Bernd Hort/assono
META-INF/services/com.ibm.xsp.adapter.servletFactory	17.04.2018 09:26:11	Bernd Hort/assono
rest-services.properties	17.04.2018 10:58:08	Bernd Hort/assono
servlets.properties	17.04.2018 09:58:56	Bernd Hort/assono

- Die Servlet-Factory liest die servlets.properties-Datei bei den Java-Design-Elementen

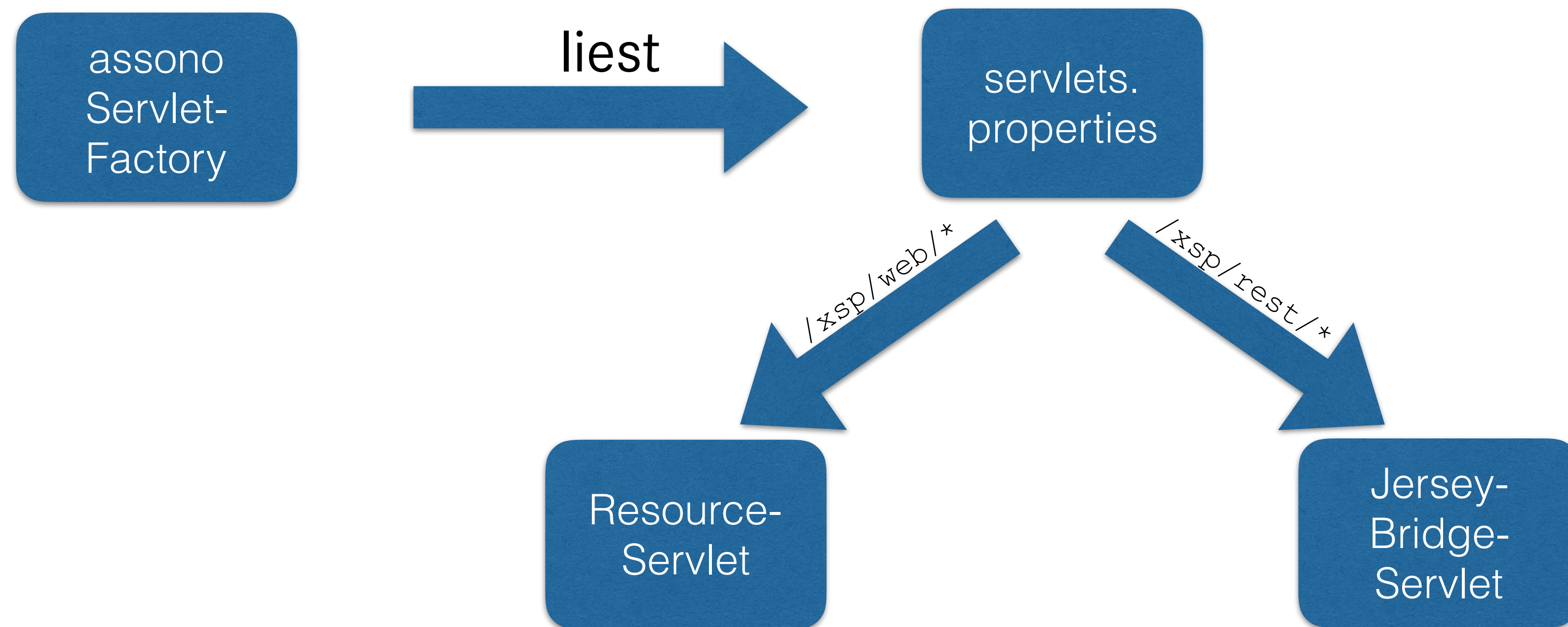


```
#servlet.class.[identifiere]=[Servlet-Classname]
#servlet.name.[identifiere]=[Servlet-Name]
#servlet.path.[identifiere]=[Servlet-Path]
#servlet.parameter.[identifiere].[parameter-name]=[parameter-value]

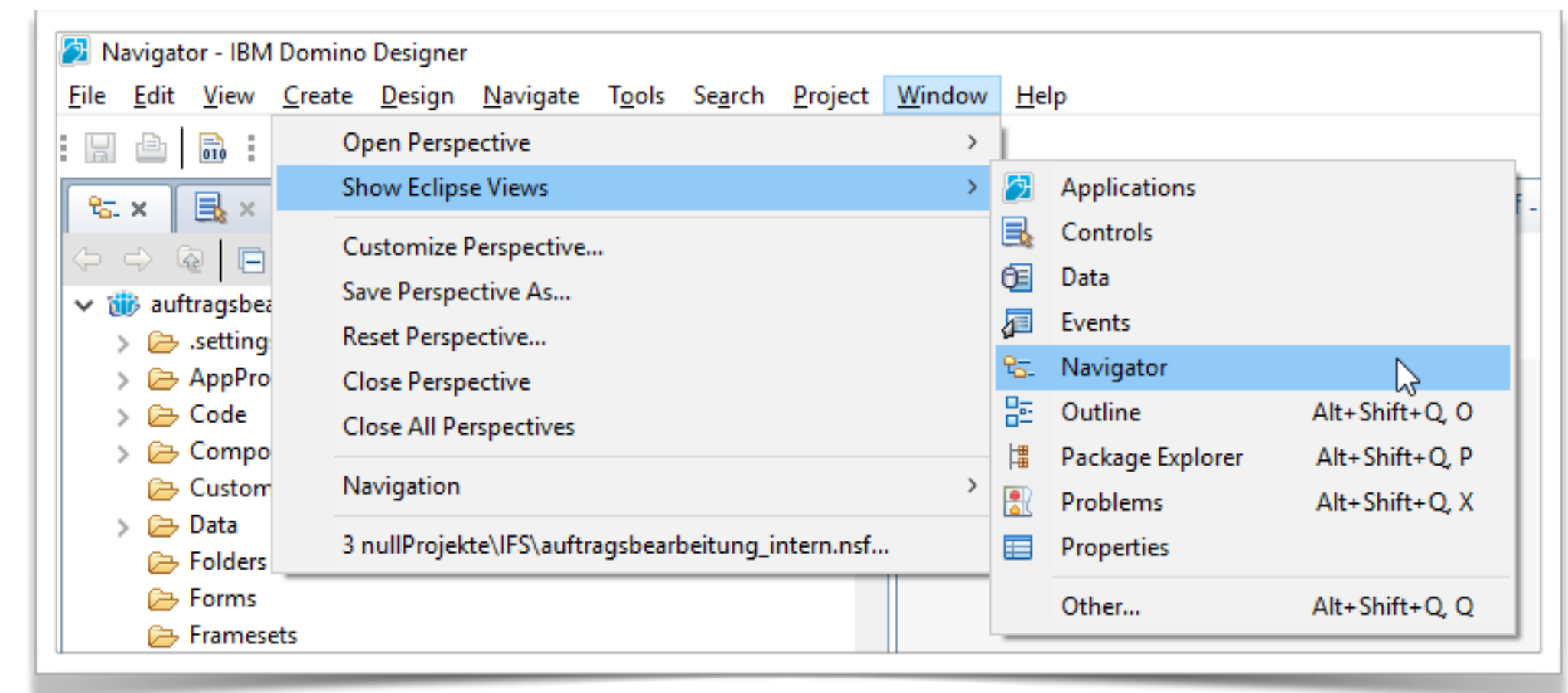
servlet.class.jersey.restful.servlet=de.assono.service.restful.jersey.JerseyBridgeServlet
servlet.path.jersey.restful.servlet=/xsp/rest/*
servlet.parameter.jersey.restful.servlet.com.sun.jersey.api.json.POJOMappingFeature=true
servlet.parameter.jersey.restful.servlet.javax.ws.rs.Application=de.assono.service.restful.jersey.JerseyApplicationBridge
servlet.parameter.jersey.restful.servlet.com.sun.jersey.config.property.packages=de.assono.service.restful;

servlet.class.web.resources.servlet=de.assono.resources.ResourceServlet
servlet.path.web.resources.servlet=/xsp/web/*
```

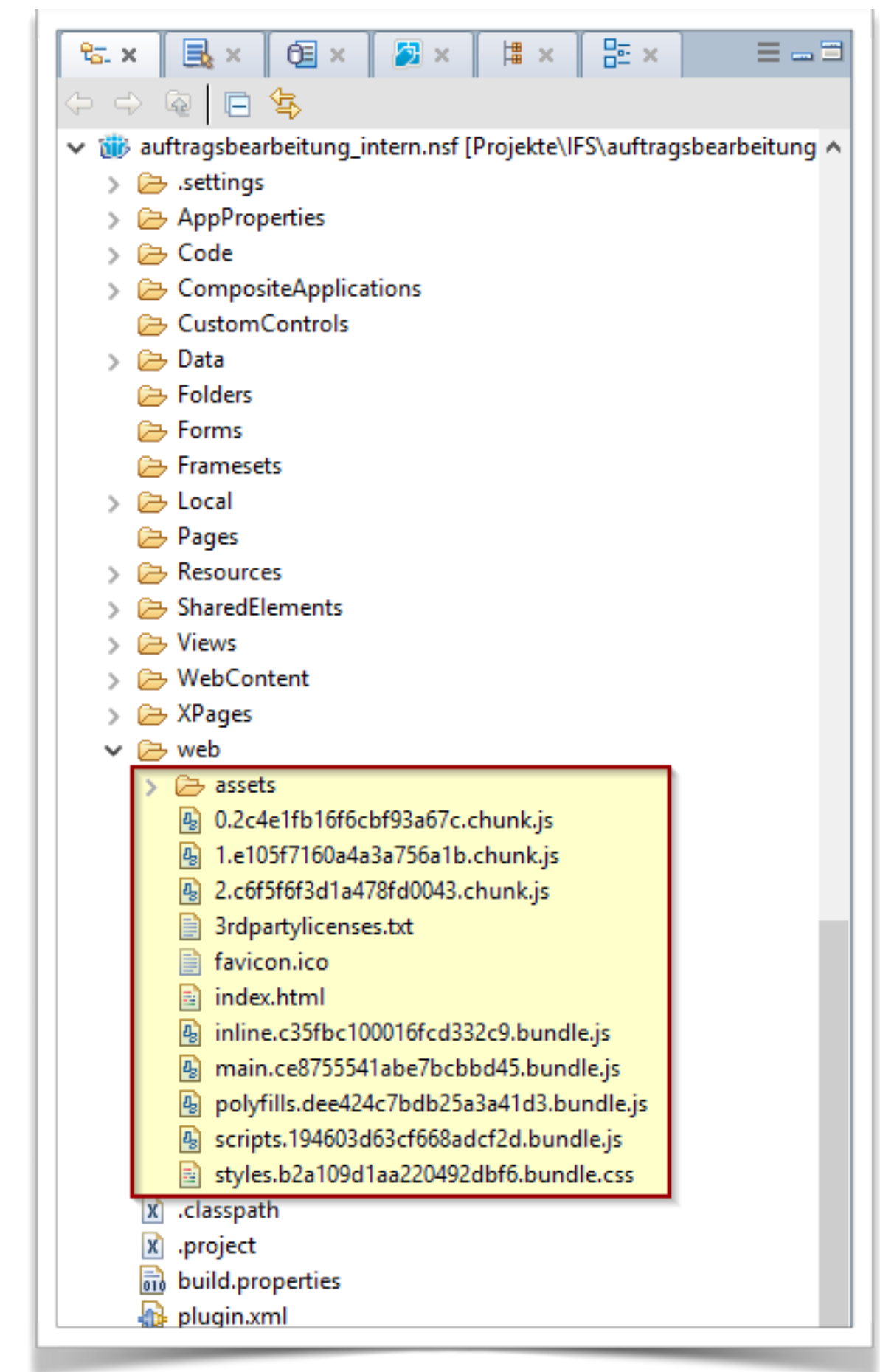
- Die servlets.properties-Datei legt die weiteren Pfade innerhalb der URL fest



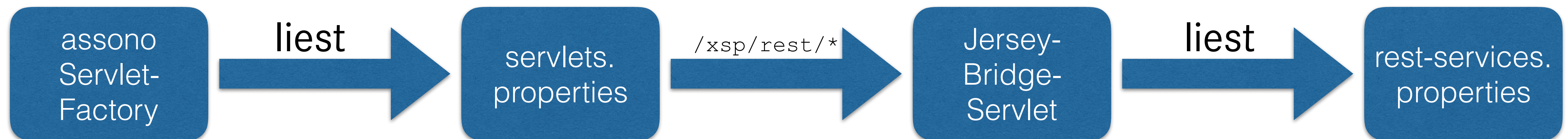
- Das Resource-Servlet liefert statische HTML-Dateien, JavaScript-Dateien etc. aus
- Es ist notwendig, weil der Domino-Server keine Web-Aufrufe von Standard-Design-Elementen wie Pages oder File Resources mit weiteren Parametern zulässt.
- Die Dateien müssen sich innerhalb der Notes-Datenbank in einem Ordner mit dem gleichen Namen befinden
Pfad /xsp/web/* => Ordernamen web
- Zugänglich über die Eclipse View „Navigator“



- In der Eclipse View „Navigator“ können die Dateien einfach per Drag & Drop aus dem Windows Explorer eingefügt werden.
- Dort können sie im Bedarfsfall auch wieder gelöscht werden.



- Jersey ist die Reference-Implementierung der Java API for RESTful Web Services (JAX RS)
 - Der Standard für das Handling von REST-Anfragen in Java
- Das Jersey-Bridge-Servlet leitet die Anfragen an JAX RS-Klassen weiter
 - Die Klassen werden in der rest-services.properties-Datei aufgelistet



- Alle JAX-RS-Klassen werden in der rest-services.properties aufgelistet
- Der Object-Mapper wandelt die Anfragen und Ergebnisse von JSON in Java-Objekte bzw. umgekehrt um
- Die Controller definieren die nach außen sichtbare fachliche API

```
# [key]=[Restful-Service-Classname]  
ObjectMapperProvider=de.assono.service.restful.jersey.ObjectMapperProvider  
  
#Auftraege  
AuftragController=de.assono.ifs.auftragsbearbeitung.auftraege.AuftragController
```



Fragen?

Bildnachweis

- Die Angular-Logos stammen von <https://angular.io/presskit>
- Teaser-Photo von Carlos Muza auf [Unsplash](#)