Software Engineering II

Einführung und Organisatorisches

Prof. Dr. Anna-Lena Lamprecht

Team

- Prof. Dr. Anna-Lena Lamprecht (Vorlesungen, Prüfung, Kundin)
- Dr. Mario Frank (Koordination, Tutor)
- Andy Großhennig (Tutor)
- Lennart Grote (Tutor)
- Jana Schulz (Kundin)
- Sebastian Müller (Kunde)
- Bharat Byan (Tech Support)

Software Engineering II

- SE1 letztes Semester:
  - klein(er)es Projekt in Zweierteams
  - zusätzlich zu Vorlesung und Übungen

- SE2 jetzt:
  - größeres Projekt in Teams von ca. 10 Personen
  - weniger Vorlesungen, weniger Übungen, Fokus auf Projekt
  - gemeinsame Entwicklung einer Webanwendung
  - Vorgehensmodell Scrum

Voraussetzungen

Vorausgesetzte Kenntnisse:

- Grundlagen der Programmierung
- Praxis der Programmierung
- Software Engineering I

Lernziele

- Grundkenntnisse Software Engineering festigen und vertiefen
- Anwendung auf komplexe(re) Softwareprojekte trainieren
- Webanwendungen planen, entwerfen und implementieren
- Moderne Webtechnologien verwenden
- Vorgehensmodell Scrum praktisch umsetzen
- Teamarbeit erfolgreich gestalten

Moodle-Kurs

- Software Engineering II (SE2_25)
- https://moodle2.uni-potsdam.de/course/view.php?id=44608
- Kein Einschreibschlüssel

Semesterplan (Änderungen möglich)

Woche	Gruppentermine (Mo, Di)	Vorlesungstermin (Do)	Abgaben (Fr)
15 (7.4.-11.4.)	Selbständig: Übung 0 (Arbeitsumgebung)	Einführung, Organisatorisches, erste Schritte mit Spring Boot	
16 (14.4.-18.4.)	Setup, “Hello World” mit Spring Boot	Mehr zu Web Development (nächste Schritte mit Spring Boot, MongoDB, Maven etc.)	
17 (21.4.-25.4.)	Vorbereitung Mini-Projekt, Registrierungs-App als Beispiel	Scrum	
18 (28.4.-2.5.)	Hilfe Mini-Projekte	*(1. Mai!)*	Mini-Projekte (PNL)
19 (5.5.-9.5.)	Projektplanung (Rollen etc.), Grundlegende Architektur- und Designentscheidungen	Kundengespräch (Design Review)	
20 (12.5.-16.5.)	Sprint Planning	Git	
21 (19.5.-23.5.)	Daily Scrum und gemeinsame Arbeitszeit	Kundengespräch (Sprint Review)	Sprintbericht 1
22 (26.5.-30.5.)	Retrospektive und Sprint Planning	*(Himmelfahrt!)*	
23 (2.6.-6.6.)	Daily Scrum und gemeinsame Arbeitszeit	Kundengespräch (Sprint Review)	Sprintbericht 2
24 (9.6.-13.6.)	*(Pfingstwoche)*	*(Pfingstwoche)*	
25 (16.6.-20.6.)	Retrospektive und Sprint Planning	Herausforderungen bei der Teamarbeit angehen	
26 (23.6.-27.6.)	Daily Scrum und gemeinsame Arbeitszeit	Kundengespräch (Sprint Review)	Sprintbericht 3
27 (30.6.-4.7.)	Retrospektive und Sprint Planning	Scrum in der Praxis (UP Transfer und Externe)	
28 (7.7.-11.7.)	Daily Scrum und gemeinsame Arbeitszeit	Kundengespräch (Sprint Review)	Sprintbericht 4
29 (14.7.-18.7.)	Retrospektive	Abschlusspräsentationen	Finales Release

Prüfung

- Prüfungsnebenleistung
  - Individuelles Auftaktprojekt („Mini-App“) auf Übungsblatt 2
  - Bestanden bei Erreichen von 50% der Punkte
  - Erforderlich zum Abschluss des Moduls

- Mündliche Prüfung (20 min)
  - Am Ende der Vorlesungszeit
  - Präsentation von Arbeitsergebnissen
  - Fragen zu theoretischen/technischen Inhalten

Umfang

- Lehrveranstaltung hat Umfang von 6 LP, also ca. 180 Stunden insgesamt (1 LP entspricht ca. 30 Stunden)
- Bei 15 Wochen Vorlesungzeit: durchschnittlich 12 Stunden pro Woche

- **4 Stunden für Vorlesung und Übung/Tutorium/Gruppentermin**
- **8 Stunden für selbständige Projektarbeit im Team**

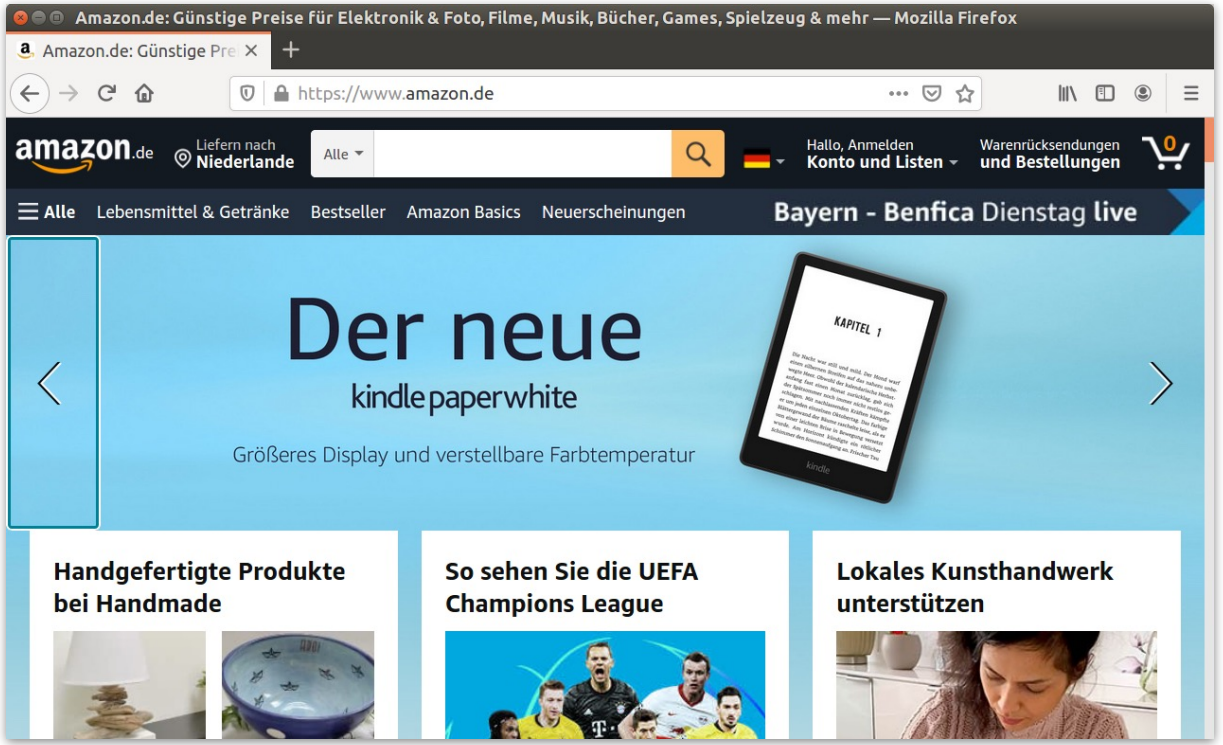
Webanwendungen

- Auch: Online-Anwendungen, Webapplikationen oder kurz Web-Apps
- Anwendungsprogramme nach dem Client-Server-Modell
- Datenverarbeitung findet teilweise auf einem entfernten Webserver statt
- Ergebnisse übertragen an den lokalen Client-Rechner des Benutzers
- Nutzung zumeist über einen Webbrowser

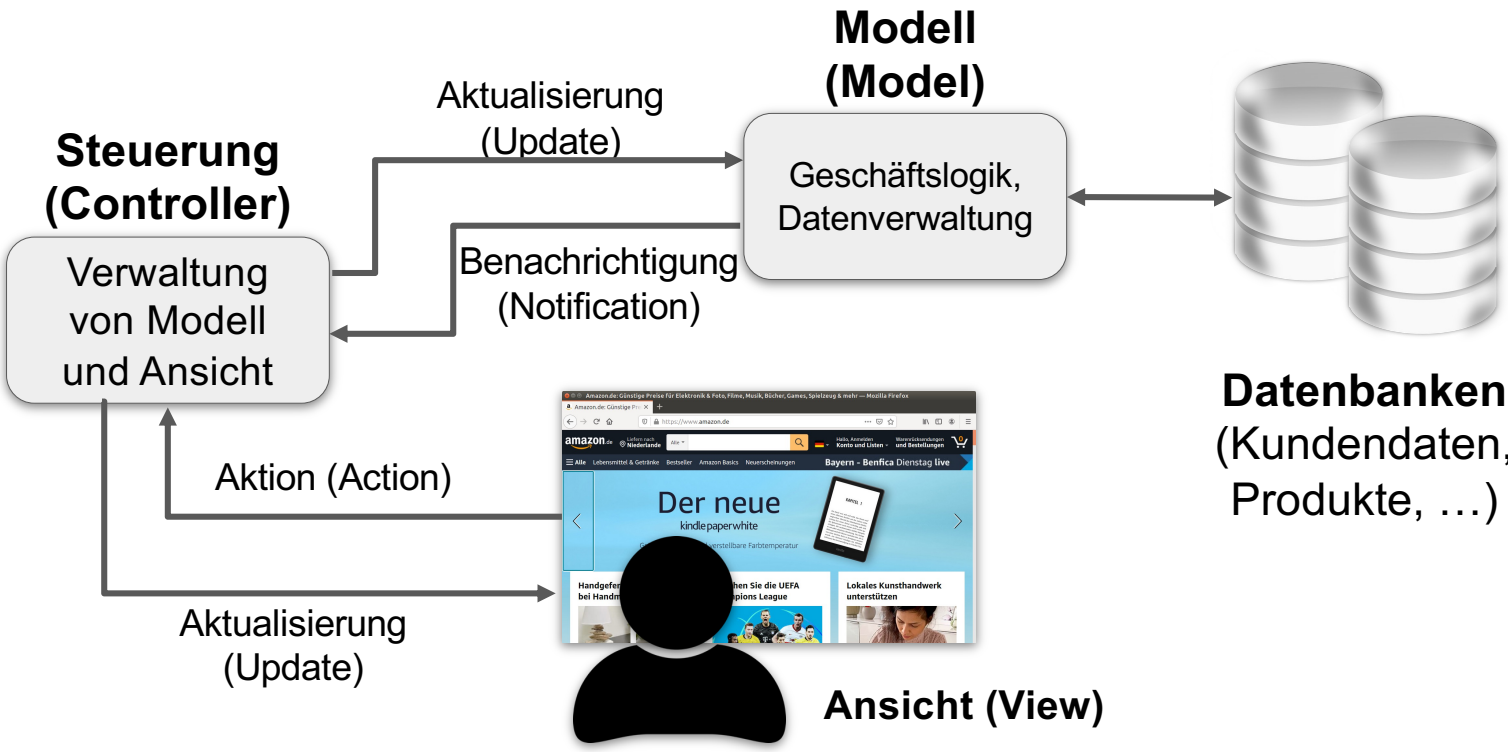
Architektur von Webanwendungen

Wie sind eigentlich Webanwendungen aufgebaut?

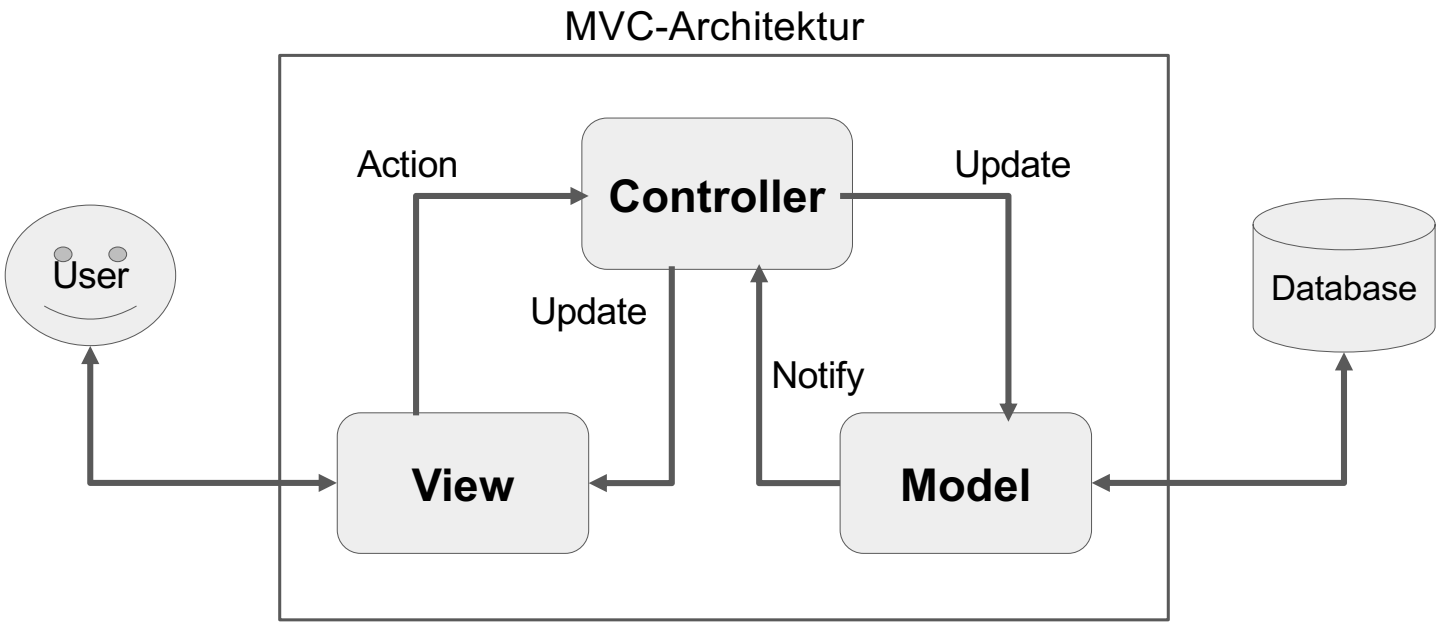
Z.B. der Online-Shop von Amazon



Model + View + Controller



Architekturmuster MVC (Model View Controller)



Webframeworks

- Webframeworks unterstützen die Entwicklung von dynamischen Webseiten, Webanwendungen oder Webservices.
- Darauf ausgelegt, sehr schnell lauffähige Webanwendungen zu erstellen.
- Vordefinierte und vorgefertigte Klassen für häufig gebrauchte Funktionen (z.B. Mailversand, sichere Authentifizierung und Authentisierung) und grundlegende Funktionen für Webformulare.
- Unterstützen saubere Trennung von Präsentation und Code durch Verwendung des MVC-Architekturmusters.
- Die meisten Webframeworks bieten auch einen Datenbankzugriff.

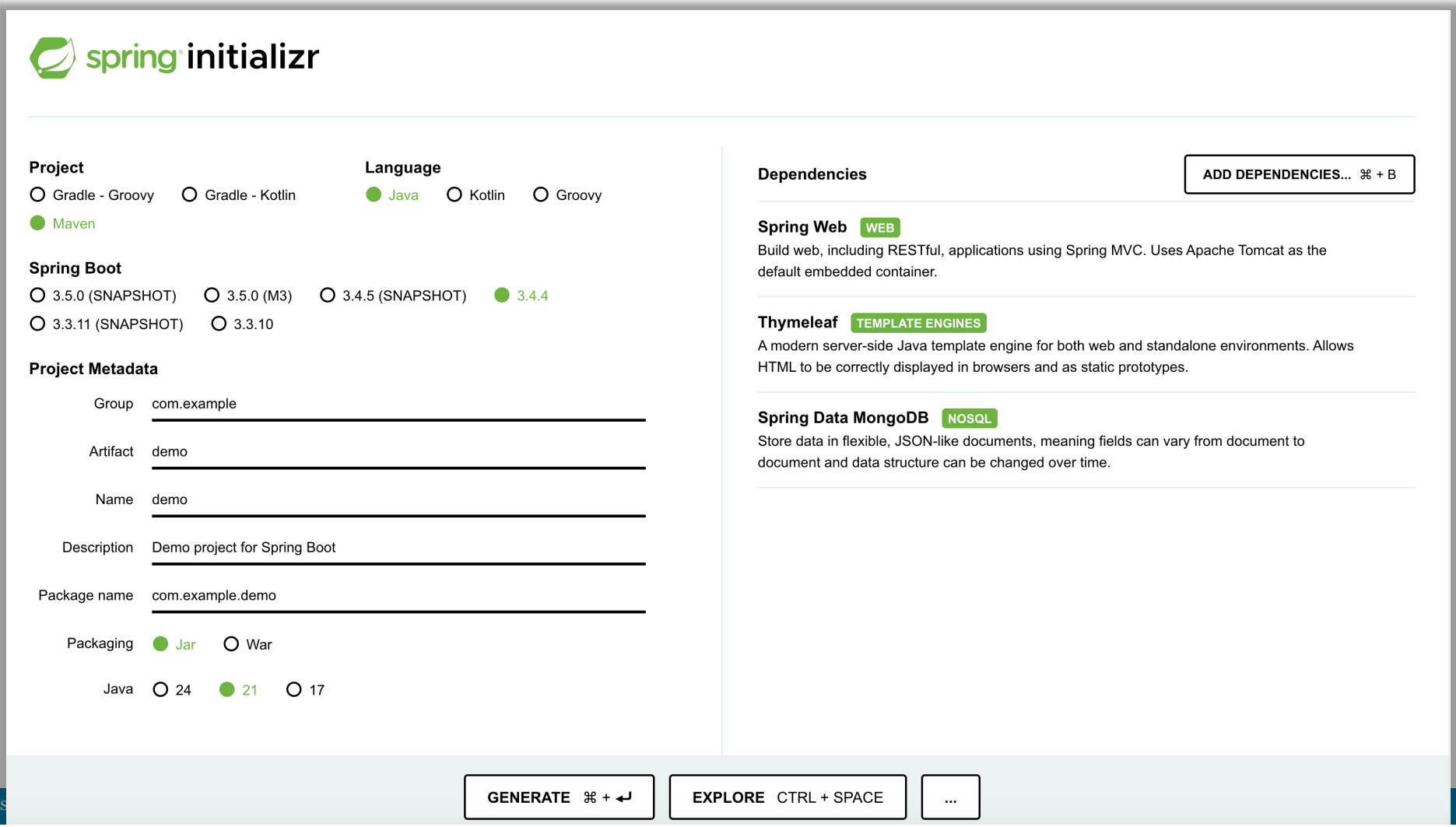
- Es gibt zahlreiche Webframeworks mit verschiedenen Schwerpunkten, siehe z.B. https://de.wikipedia.org/wiki/Liste_von_Webframeworks

Frameworks in SE2

Spring / Spring Boot (https://spring.io/projects/spring-boot)

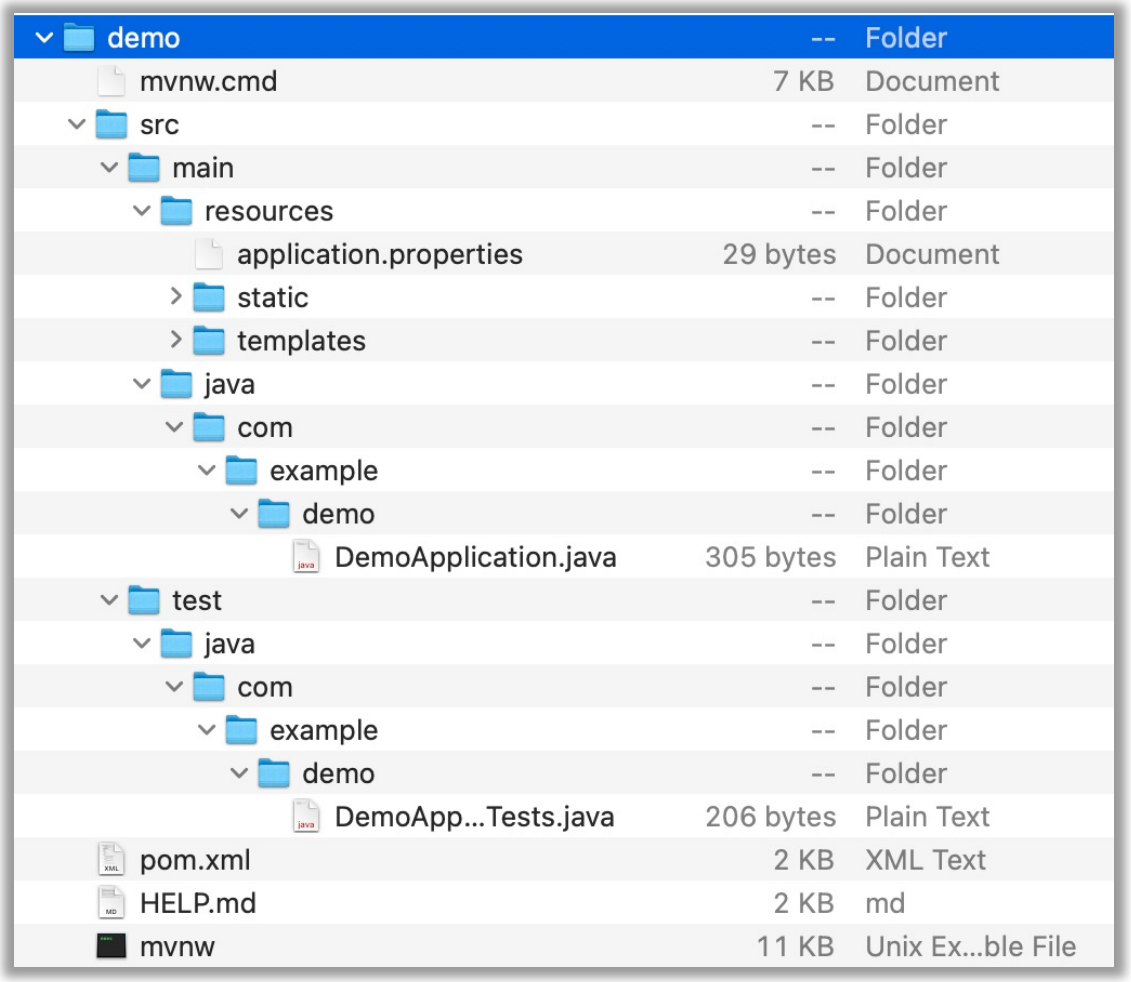
- Spring: beliebtes Open-Source-Framework zur Erstellung eigenständiger Java-Anwendungen, die auf der Java Virtual Machine (JVM) laufen.
- Spring Boot: vereinfacht die Entwicklung von Webanwendungen und Mikroservices aus Basis von Spring

Spring initializr (https://start.spring.io/)

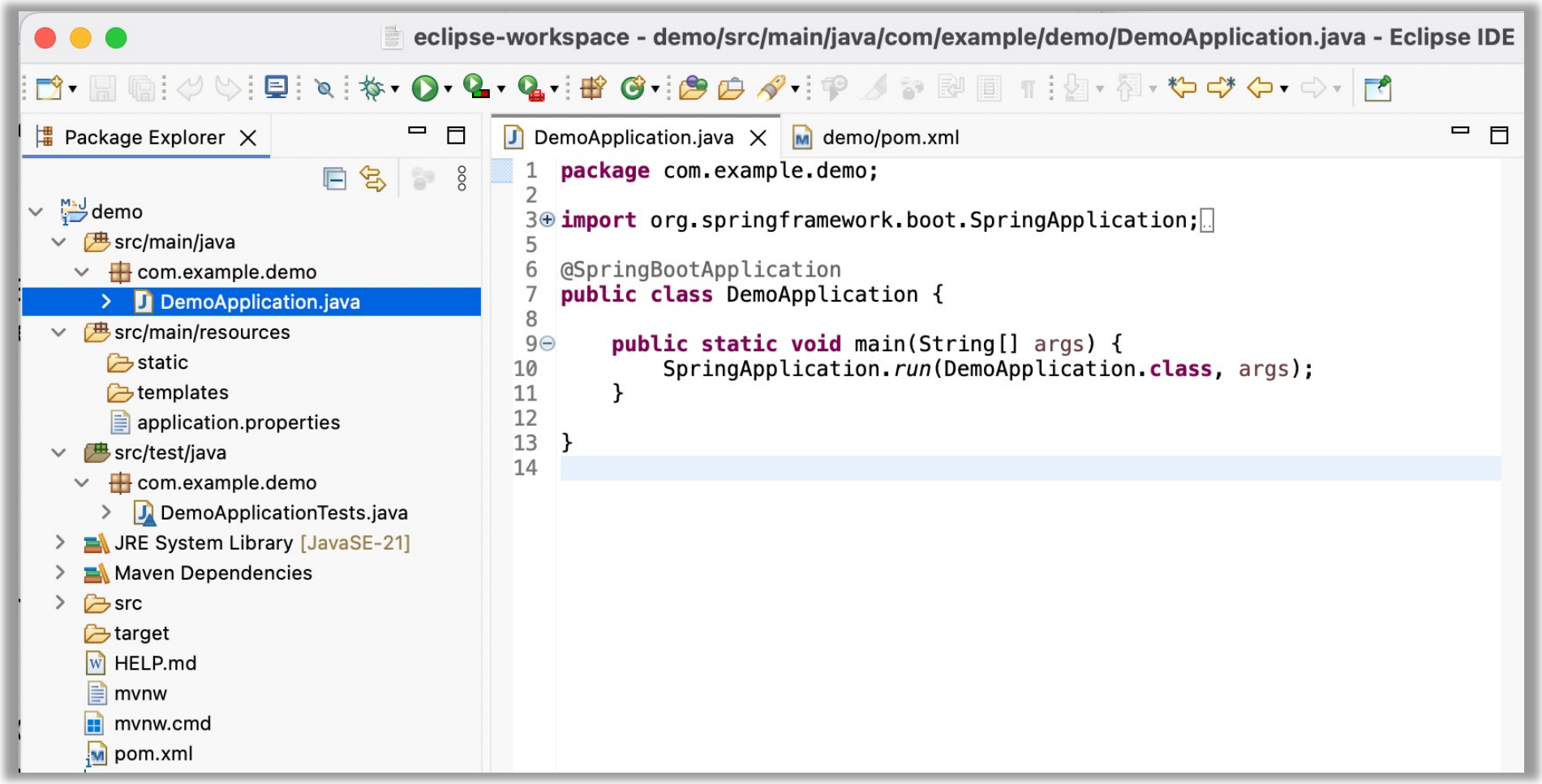


Generierte Anwendung

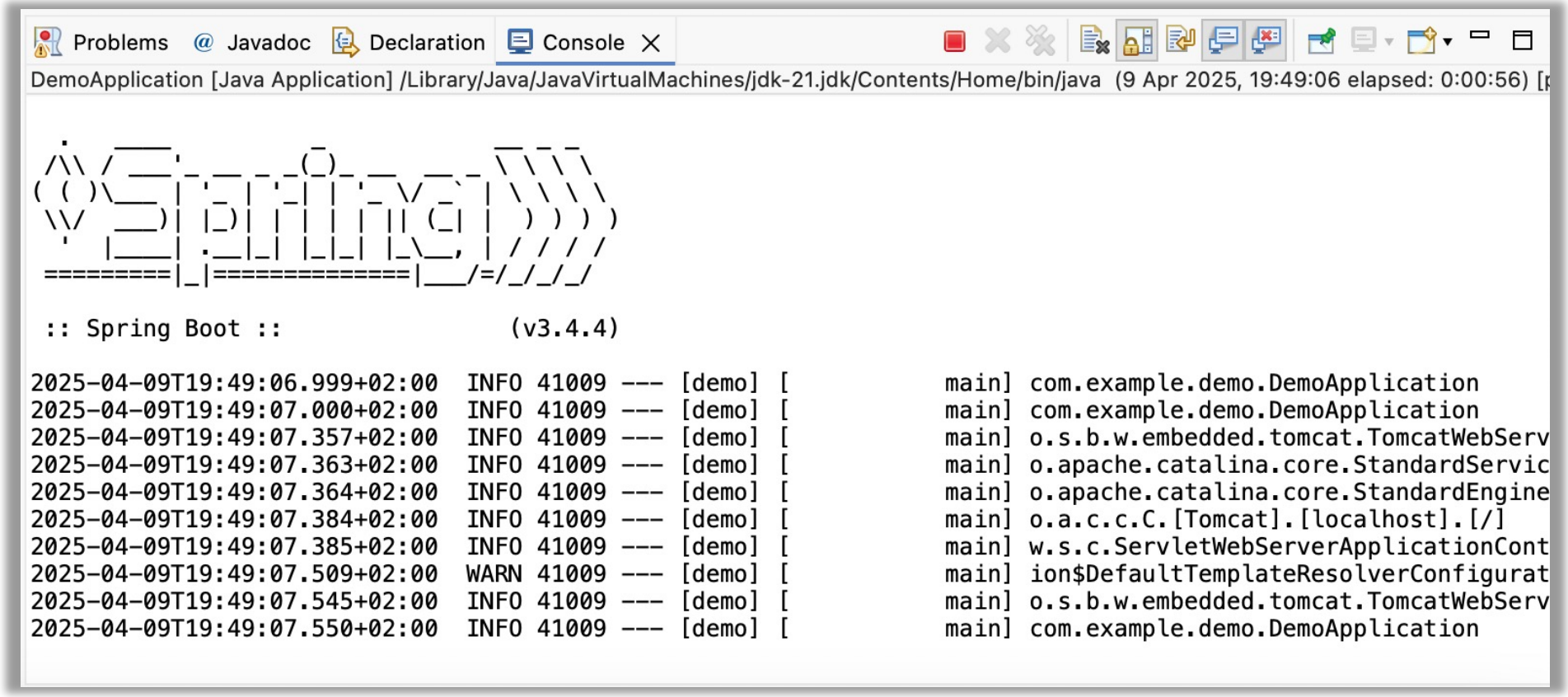
- Allgemeines Codegerüst für eine Webanwendung
- Aber noch ohne Inhalt und Funktionalität



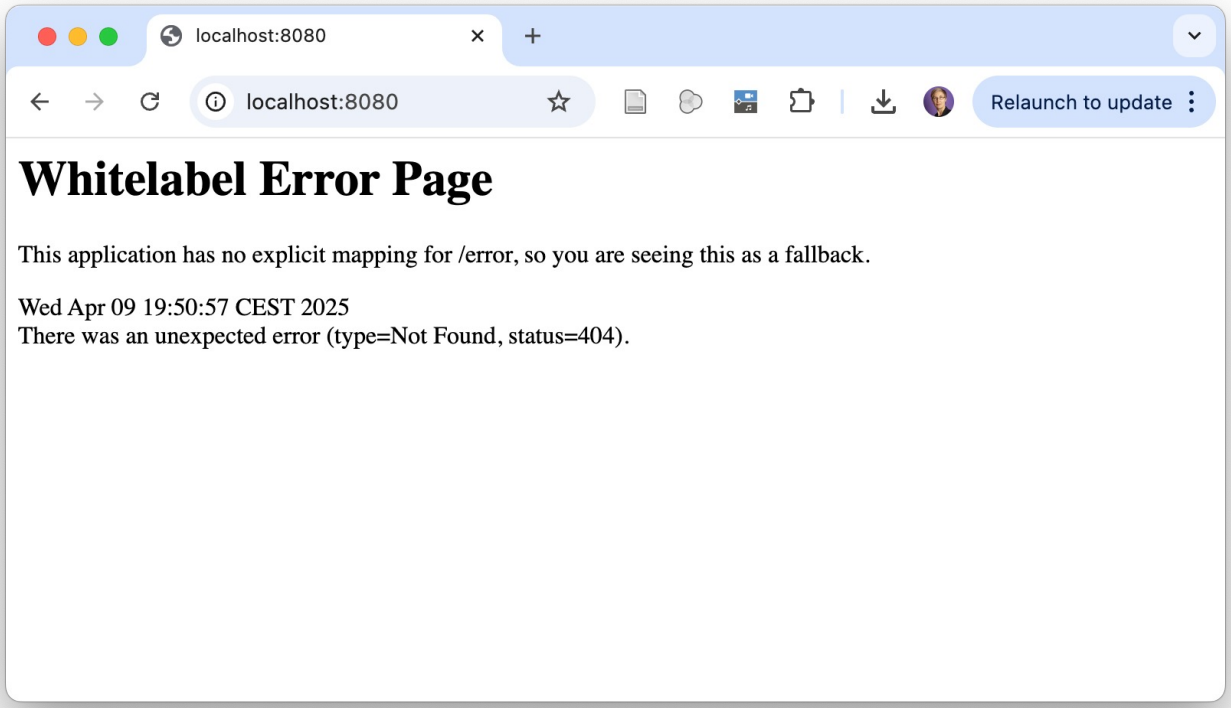
Weiterbearbeitung in IDE (z.B. Eclipse)



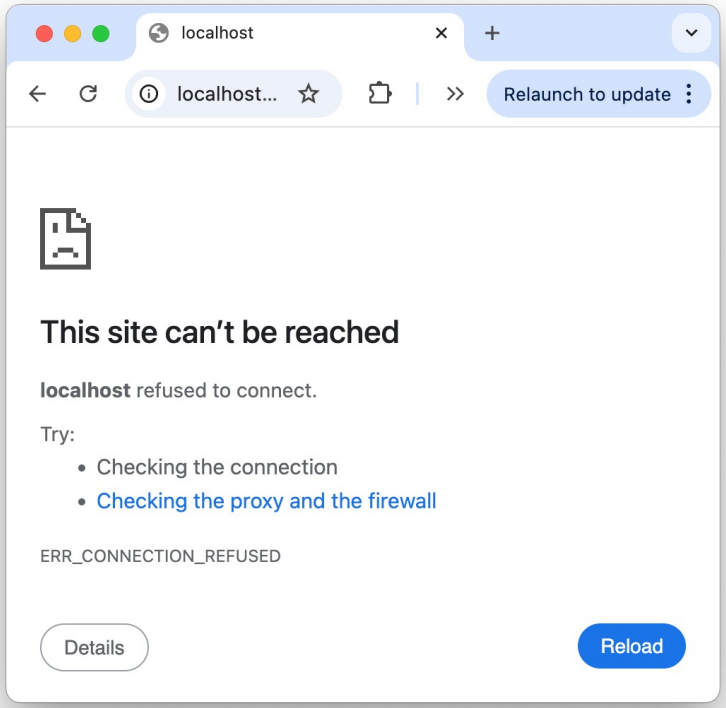
Ausführen der Anwendung startet den Server (lokal)



Im Browser erreichbar unter http://localhost:8080/

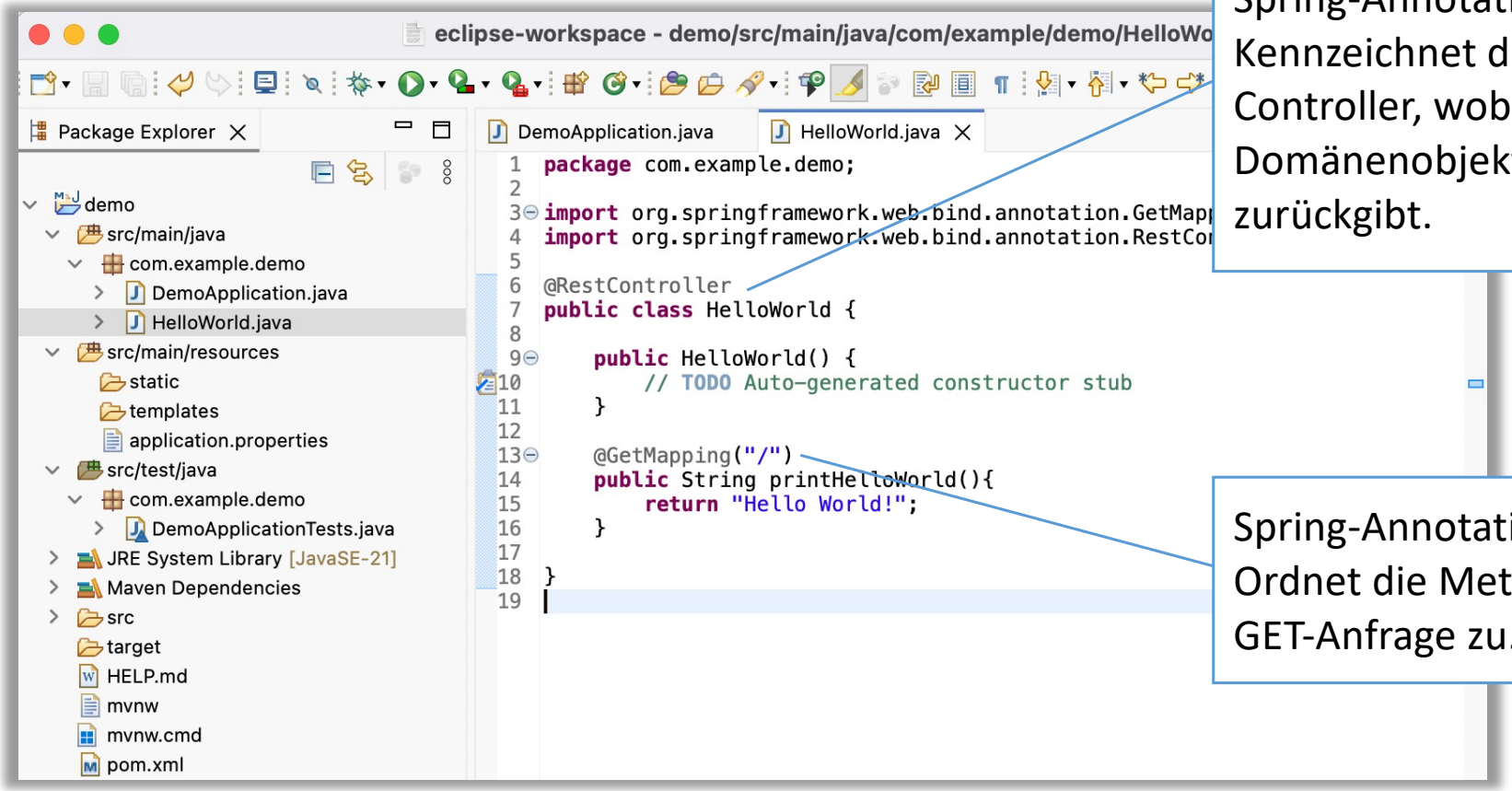


Bei laufender Webanwendung



Wenn die Anwendung nicht läuft

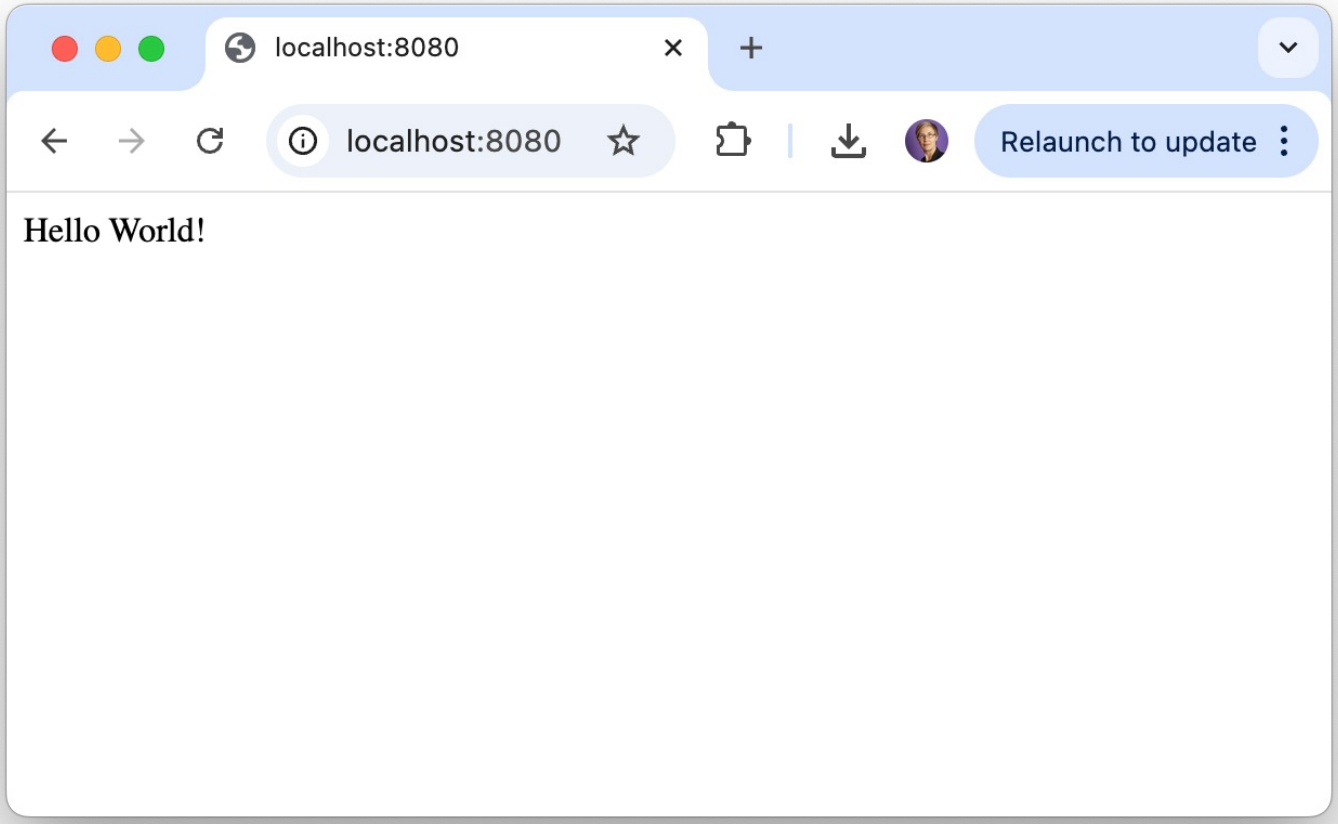
Funktionalität hinzufügen – “Hello World!”



Spring-Annotation **@RestController:** Kennzeichnet die Klasse als Controller, wobei jede Methode ein Domänenobjekt (statt einer Ansicht) zurückgibt.

Spring-Annotation **@GetMapping:** Ordnet die Methode einer HTTP-GET-Anfrage zu.

Et voilá: Hello World!



Frameworks in SE2

Spring / Spring Boot (https://spring.io/projects/spring-boot)

- Spring: beliebtes Open-Source-Framework zur Erstellung eigenständiger Java-Anwendungen, die auf der Java Virtual Machine (JVM) laufen.
- Spring Boot: vereinfacht die Entwicklung von Webanwendungen und Mikroservices aus Basis von Spring

MongoDB (https://www.mongodb.com/de-de)

- Beliebtes Datenbankmanagementsystem für Web- und Mobilanwendungen
- Dokumentenorientierte, NoSQL-Datenbank (verwaltet Sammlungen von JSON-ähnlichen Dokumenten)

Siehe auch **Übungsblatt 0** (Einrichtung der Arbeitsumgebung für den Kurs).

Mehr zu Spring und MongoDB

- … nächste Woche!

Dependency Management (Abhängigkeitsverwaltung)

- Verwaltung von Abhängigkeiten zwischen verschiedenen Komponenten innerhalb eines Softwareprojekts:
  - Abbildung der Abhängigkeitsbeziehungen zwischen Komponenten
  - Verwaltung der Versionen von Abhängigkeiten
  - Lösung von Konflikten, die entstehen, wenn verschiedene Komponenten konkurrierende Versionen derselben Abhängigkeit erfordern

- Verwendung von Automatisierungstools und Paketmanagern wie npm, Maven oder pip, um den Prozess der Installation, Aktualisierung und Entfernung von Abhängigkeiten zu optimieren,

Mehr zu Dependency Management mit Maven

- … auch nächste Woche!

Vorgehensmodell Scrum (Wdh.)

- Verbreitetes Vorgehensmodell
- Entstammt der Idee der “Lean Production”
- Weitgehend selbständige Organisation des Projektteams anhand von Ritualen (z.B. Daily Scrum, Retrospectives)
- Geht davon aus, dass Projekt komplex und reich an Unsicherheiten sind und sich daher nicht von Anfang an detailliert planen lassen.

Rollen in Scrum

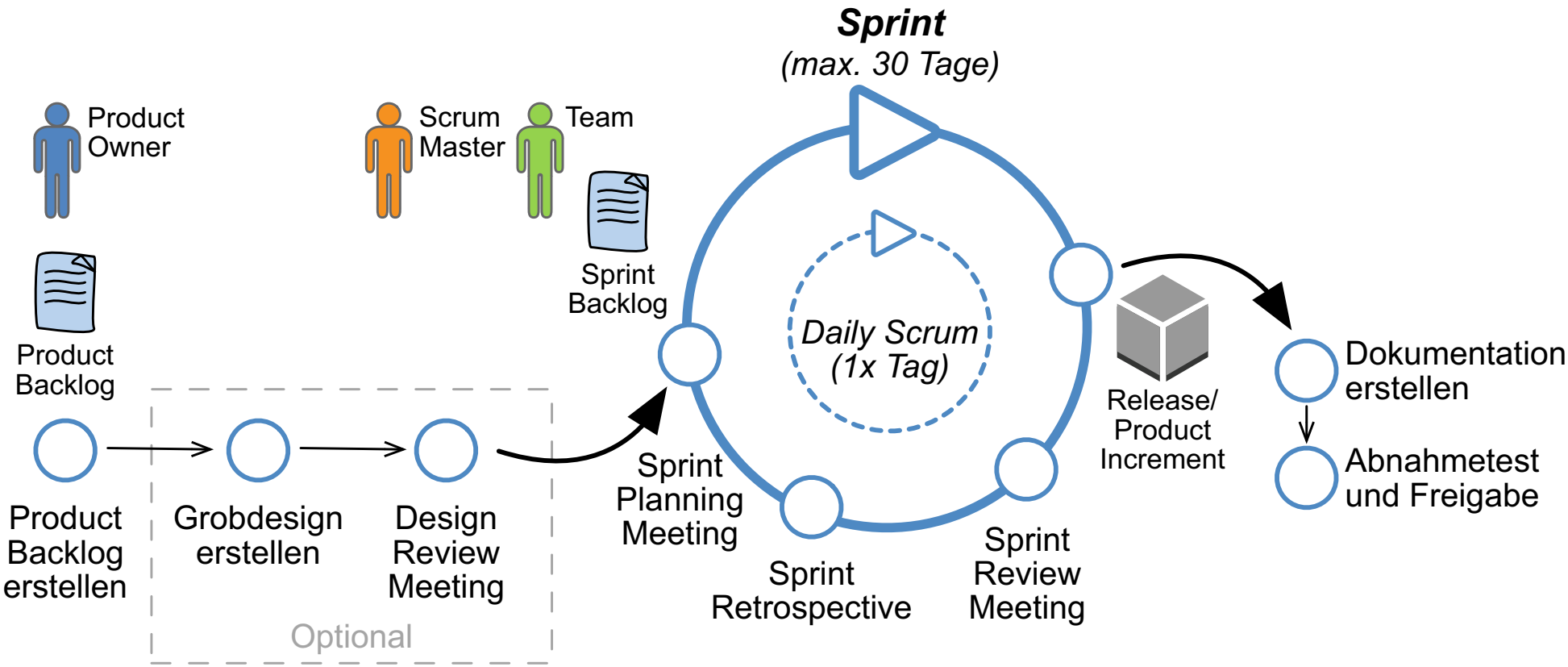
Drei Rollen:

- **Product Owner** (Verantwortung für das Ergebnis)
- **Entwicklungsteam** (Implementierung, idealerweise 3-9 Personen)
- **Scrum Master** (Zuständig für die Umsetzung des Scrum-Prozesses)

Artefakte in Scrum

- **Product Backlog** (enthält alle bekannten Anforderungen, erstellt und gepflegt durch Product Owner)
- **Sprint Backlog** (Zielvorgabe für einen Sprint, Auswahl aus dem Product Backlog, erstellt durch Projektteam)
- **Release** (Abschluss eines Sprints)

Scrum: Prozess (vereinfacht)



Scrum: Anforderungen an die Organisation

Erfordert

- hohe Eigenverantwortung
- hohen Reifegrad des Unternehmens
- Beherrschung von Techniken wie automatische Testverfahren und Refactoring
- Ausbildung der Mitarbeiter in modernen Progammiertechniken
- sozial ausgewogenes Team

Mehr zu Scrum

… in zwei Wochen!

Was ist jetzt wichtig?

- In PULS und Moodle einschreiben
- Übungsblatt 0 (Arbeitsumgebung einrichten) selbständig bearbeiten, bei Bedarf Tutoren um Hilfe bitten
- Übungsblatt 1 bearbeiten und (ggf.) erste Übung besuchen
- Nächste Woche wieder in die Vorlesung kommen. ☺

Semesterplan (Änderungen möglich)

Woche	Gruppentermine (Mo, Di)	Vorlesungstermin (Do)	Abgaben (Fr)
15 (7.4.-11.4.)	Selbständig: Übung 0 (Arbeitsumgebung)	Einführung, Organisatorisches, erste Schritte mit Spring Boot	
16 (14.4.-18.4.)	Setup, “Hello World” mit Spring Boot	Mehr zu Web Development (nächste Schritte mit Spring Boot, MongoDB, Maven etc.)	
17 (21.4.-25.4.)	Vorbereitung Mini-Projekt, Registrierungs-App als Beispiel	Scrum	
18 (28.4.-2.5.)	Hilfe Mini-Projekte	*(1. Mai!)*	Mini-Projekte (PNL)
19 (5.5.-9.5.)	Projektplanung (Rollen etc.), Grundlegende Architektur- und Designentscheidungen	Kundengespräch (Design Review)	
20 (12.5.-16.5.)	Sprint Planning	Git	
21 (19.5.-23.5.)	Daily Scrum und gemeinsame Arbeitszeit	Kundengespräch (Sprint Review)	Sprintbericht 1
22 (26.5.-30.5.)	Retrospektive und Sprint Planning	*(Himmelfahrt!)*	
23 (2.6.-6.6.)	Daily Scrum und gemeinsame Arbeitszeit	Kundengespräch (Sprint Review)	Sprintbericht 2
24 (9.6.-13.6.)	*(Pfingstwoche)*	*(Pfingstwoche)*	
25 (16.6.-20.6.)	Retrospektive und Sprint Planning	Herausforderungen bei der Teamarbeit angehen	
26 (23.6.-27.6.)	Daily Scrum und gemeinsame Arbeitszeit	Kundengespräch (Sprint Review)	Sprintbericht 3
27 (30.6.-4.7.)	Retrospektive und Sprint Planning	Scrum in der Praxis (UP Transfer und Externe)	
28 (7.7.-11.7.)	Daily Scrum und gemeinsame Arbeitszeit	Kundengespräch (Sprint Review)	Sprintbericht 4
29 (14.7.-18.7.)	Retrospektive	Abschlusspräsentationen	Finales Release