SEW V – Rest

PROF. RALPH JANK, MSC

Architekturkonzepte

Roy Fielding identifizierte in seiner Dissertation (2000) fünf Architekturkonzepte:

- *Addressable Resources*: Ressourcen sind über URI ansprechbar
- *A Uniform, Constrained Interface*: Beschränkte Anzahl von Operationen zur Manipulation
- *Representation-Oriented*: Ressourcen stehen in verschiedenen Repräsentationen zur Verfügung
- *Communicate Statelessly*: Zustandslose Anwendungen skalieren besser
- *Hypermedia As The Engine Of Application State*: Ressourcen enthalten Links auf mögliche Operationen

Diese Konzepte werden unter dem Namen **Re**presentation **S**tate **T**ransfer (REST) zusammengefasst

Uniform, Constrained Interface

Für jede Ressource steht eine beschränkte Anzahl von Operationen zur Verfügung (HTTP)

- GET
  - Lesen der Ressource
  - Idempotent und sicher (Zustand wird nicht verändert)
- PUT
  - Aktualisieren der übergebenen Ressource (speichern bzw. einfügen, falls nicht vorhanden)
  - Idempotent: Mehrmaliges Speichern ändert den Zustand der Ressource nicht
- POST
  - Hinzufügen der übergebenen Ressource
  - Nicht Idempotent: Jede POST-Operation verändert Zustand
- DELETE
  - Löschen der Ressource
- HEAD
  - Wie GET, jedoch werden nur Rückgabe-Codes geliefert
- Options
  - Welche Methoden unterstütz eine Ressource

Uniform, Constrained Interface - Vorteile

Einfachheit

- Interface ist leicht zu verstehen
- Bedarf für Interface-Beschreibung (WSDL) ist geringer
- Keine Proxys und keine komplexe Client-Bibliothek notwendig

Interoperabilität

- HTTP-Client-Biobliothek ist ausreichend und existiert praktisch für jede Programmiersprache
- Keine Probleme mit Hersteller-Interoperabilität (Kompatibilitätsporbleme bei Implementierungen von WS-*-Protokollen)

Skalierbarkeit

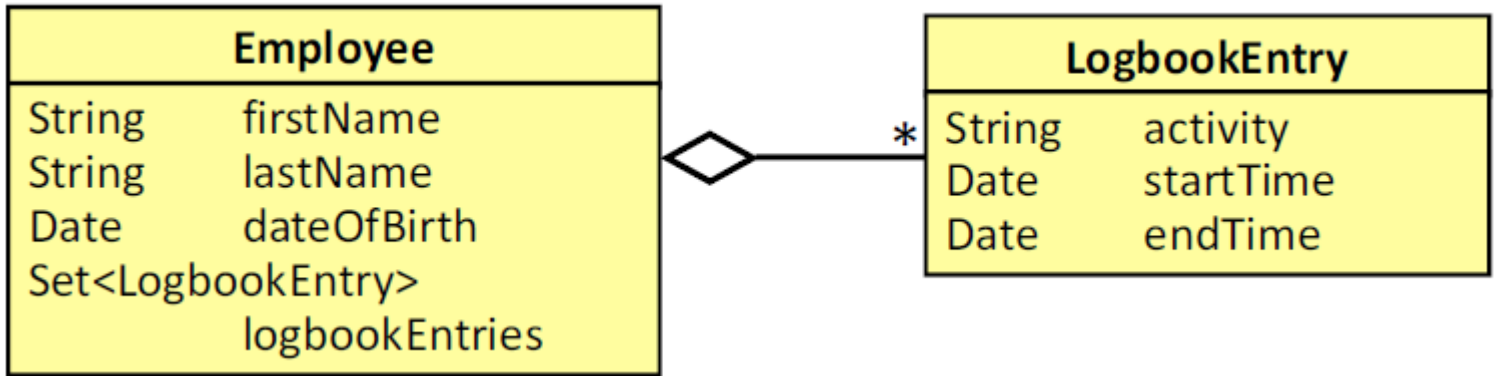
- Ergebnisse von lesenden Operationen können zwischengespeichert (Caching) werden: Client-seitig oder von Proxy-Server

Representation-Oriented

- Client und Server tauschen mit HTTP-Anfragen und –Antworten Repräsentationen der Ressourcen untereinander aus.
- Die Repräsentation der Daten (Ressourcen) ist im Gegensatz zu SOAP-basierten Web-Services nicht standardisiert.
- Verbreitete Datenformate:
  - XML (Java)
  - JSON
  - YAML (Pearl, Python, Ruby)
- Mit dem HTTP-Header Content-Type wird definiert, in welcher Repräsentation die Ressource übertragen wird
- Im HTTP-Header Accept kann der Client das gewünschte Datenformat angeben (z.B. Accept: application/XML)

Entwurf von RESTful Web-Services

Festlegen des Objektmodells (Domänenmodells)



Modellierung der URIs

- /employees: alle Angestellten
- /employees/{id}: ein einzelner Angestellter
- /logbookentries: alle Arbeitszeiteinträge
- /logbookentries/{id}: ein einzelner Arbeitszeiteintrag
- /employees/{id}/logbookentries: alle Arbeitszeiteinträge eines Angestellten

SOAP-basierte vs. RESTful Web-Services

Vorteile von RESTful Web-Services

- REST-basierte Frameworks sind einfacher zu implementieren
- Für Clients ist nur eine HTTP-Bibliothek erforderlich
- Ressourcen können in verschiedenen Repräsentationen ausgeliefert werden (Client kann passende Repräsentation wählen)
- HTTP-Antworten können zwischengespeichert werden (Caching).

Nachteile von RESTful Web-Services

- Keine Standardisierung der Datenrepräsentation
- Keine Unterstützung von Standards auf Nachrichtenebene (WS-Security, WS-Transactions etc.)
- Parsen und Generieren der Ressourcenpräsentation kann aufwändig sein (service- und clientseitig)

Welchen Ansatz man wählt, hängt stark vom Anwendungsgebiet ab