Modellierung der Schnittstelle - REST

1. Definition von Ressourcen:

Das zentrale Konzept von REST sind Ressourcen. Eine Ressource ist eindeutig identifizierbar und hat eine oder mehrere Repräsentationen, die sie ihrer Außenwelt zur Verfügung stellt und über die sie bearbeitet wird. Repräsentationen sind Darstellungen einer Ressource in einem definierten Format. Wir sehen nie die Ressourcen, sondern nur ihre Repräsentationen. Tatsächlich können wir die Ressource sogar darüber definieren: als eine gemeinsame ID zusammengehaltene Menge von Repräsentationen.
Häufig ist die Grenze zwischen einer Ressource mit mehreren Repräsentationen auf der einen und separaten Ressourcen auf der anderen Seite fließend.
Nicht umsonst werden Architekturen, die sich am REST-Stil orientieren, auch als ROA (Resource Oriented Architecture) bezeichnet: Ressourcen stehen im Mittelpunkt Ihres Entwurfs. Dem richtigen Design Ihrer Ressourcen kommt daher eine besonders hohe Bedeutung zu.

Wenn die Ressourcen für den Anwendungsfall entworfen werden, werden die Schnittstelle damit zur Außenwelt beschrieben. Zur besseren Strukturierung werden die unterschiedliche Ressourcenkategorien unterschieden.

- Primärressourcen:

Primärressourcen sind die bei der Betrachtung einer Domäne offensichtlich zu modellierenden Entitäten. Für den Nutzer der Ressourcen, sei es ein Webbrowser oder eine andere Anwendung, ist die Implementierung, die hinter den Ressourcen steht, vollständig transparent.
Beispiel für eine Primärressource in unserem Projekt ist der Arbeitsplan. Mit GET könnte der Vorgesetzter sich über den aktuellen Arbeitsplan informieren oder mit PUT könnte er den Arbeitsplan verändern.

- Listenressourcen:

Es gibt für die meisten Primärressourcen in der Regel auch eine zugehörige Listenressource: die Menge alle Angestellten, alle Büroraumen usw. Dass diese ebenfalls eigene Ressourcen sind, bedeutet, dass auch sie eindeutig identifiziert werden und möglicherweise mehr als eine Repräsentation haben.
Ein Beispiel für eine Listenressource haben wir in unserem Projekt gesehen: die Liste der Angestellten der Firma. Mit GET können alle Angestellten ausgegeben werden und über ein POST könnte der Admin einen neuen Angestellten als Element der Liste erstellen.

Ressourcen sind identifizierbar: Über eine ID, im Web also eine URI, wird genau eine Ressource benannt. Der Umkehrschluss gilt jedoch nicht: Eine Ressource kann unter mehreren IDs gefunden werden.
Eine HTTP-URI besteht aus einzelnen Komponenten, die durch eines der von der URI-Spezifikation definierten Trennzeichen voneinander separiert werden. Zu den wichtigsten Trennzeichen gehören „:“, “ /“, „?“ und „#“.

- Path-Parameter:

Den ersten URI-Bestandteil, der auf den Hostnamen und den optionalen Port folgt, bezeichnet man als „Pfad“. Beispiel:

`http://example.com/main/subsegment1/components/3`

Neben absolut Pfaden, die mit einem „//“ beginnen, gibt es auch relative Pfade – ebenfalls analog zu einem Dateisystem. Sie beginnen mit einem einfachen „/“ und können „..“ und „.“ enthalten. Damit lassen sich Ressourcen relativ zueinander adressieren.

- Query-Parameter:

Der Pfadanteil einer URI wird durch ein (optionales) „?“ von einem (ebenfalls optionalen) weiteren Anteil getrennt, der als Query bezeichnet wird. Ein Beispiel ist den folgenden Link, um die Wetterdaten in Köln mit der ID 2886242 durch der Seite Openweathermap mit meinem Schlüssel zu erhalten. (alles wird in meinem probierenden Wetter API angezeigt.)

**http://api.openweathermap.org/data/2.5/weather?id=2886242&APPID=fdecbfe26d1ea60cb2b8fd91b0e2571b**

Der Query-Anteil besteht aus der Zeichenkette **id=2886242&APPID=fdecbfe26d1ea60cb2b8fd91b0e2571b**, der in diesem Fall zwei Query-Parameter enthält, die durch das Kaufmanns-Und (&) verkettet werden. Der Ordnung halber: Weder die REST-Dissertation noch der HTTP- oder URI-Standard definieren Query-Parameter; sie sind jedoch der Standard bei HTML-Formularen, deren Inhalte per GET an den Server übermittelt werden.

## 2. REST Prinzipien:

- Eindeutige Identifikation: Jede Entität erhält einen eindeutigen Identifier

Beispiel:

http://example.com/customers/1234
http://example.com/orders/2007/10/776654

- Hypermedia: Verlinkung von anderen Ressourcen auf unserer Ressource

Beispiel:

```
<arbeitsplan href='http://burohund.com/angestellter/1234' >
   <beginn>8</beginn>
   <ende>17</ende>
   <pausenzeiten href='http://burohund.com/angestellter/1234/pausengestaltung/1' />
</arbeitsplan>
```

- Standardmethoden: verwende die Standard Verben

| | |
|---|---|
| GET | Rufe Information ab (Caching möglich) |
| PUT | Aktualisiere oder Erzeuge Information zu einer bekannten Entität |
| POST | Erstelle eine Entität, die einer bekannten Entität untergeordnet ist |
| DELETE | Entferne eine Entität |

- Ressourcen und Repräsentationen: sieh unterschiedliche Repräsentationen vor

Beispiel:

```
GET /angestellter/1234
Host: burohund.com
Accept: text/XML

GET /angestellter/1234
Host: burohund.com
Accept: application/json

{“name”      :
      {
            “vorname“    : „Trang“,
            „nachname“   : „Nguyen“
      }
},

{“adresse”   :
      {
            “strasse”    : “Am Sandberg 7”,
```

```
                "stadt"      : "Gummersbach",
                "bundesland" : "NRW",
                "PLZ"        : "51643"
        }
}
```

- Statuslose Kommunikation: jeder Schritt eines Dialoges soll unabhängig von anderen sein und nicht auf dem Server gespeichert werden.

## 3. REST API – Response Code and Statuses:

| Code | Status | Description |
|---|---|---|
| 200 | OK | The request was successfully completed. |
| 201 | Created | A new resource was successfully created. |
| 400 | Bad Request | The request was invalid. |
| 401 | Unauthorized | The request did not include an authentication token or the authentication token was expired. |
| 403 | Forbidden | The client did not have permission to access the requested resource. |
| 404 | Not Found | The requested resource was not found. |
| 405 | Method Not Allowed | The HTTP method in the request was not supported by the resource. For example, the DELETE method cannot be used with the Agent API. |
| 409 | Conflict | The request could not be completed due to a conflict. For example, POST ContentStore Folder API cannot complete if the given file or folder name already exists in the parent location. |
| 500 | Internal Server Error | The request was not completed due to an internal error on the server side. |
| 503 | Service Unavailable | The server was unavailable. |