HTWK Leipzig

Fakultät für Informatik, Mathematik & Naturwissenschaften

Bachelor-Thesis

Generierung und Design einer Client-Bibliothek für einen RESTful Web Service am Beispiel der Spreadshirt-API

Author:

Andreas Linz

10INB-T

admin@klingt.net

Leipzig, 25. Juni 2013

Gutachter:

Dr. rer. nat. Johannes Waldmann

HTWK Leipzig – Fakultät für Informatik, Mathematik & Naturwissenschaften

waldmann@imn.htwk-leipzig.de

HTWK Leipzig, F-IMN, Postfach 301166, 04251 Leipzig

Jens Hadlich

Spreadshirt HQ, Gießerstraße 27, 04229 Leipzig

jns@spreadshirt.net

Andreas Linz
Nibelungenring 52
04279 Leipzig
admin@klingt.net
www.klingt.net

Generierung und Design einer Client-Bibliothek für einen RESTful Web Service am Beispiel der Spreadshirt-API
Bachelor Thesis, HTWK-Leipzig, 25. Juni 2013

made with XeTeX, LaTeX and BibTeX.

Selbständigkeitserklärung

Ich erkläre hiermit, dass ich diese Bachelor-Thesis selbstständig ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel verfasst habe. Alle den benutzten Quellen wörtlich oder sinngemäß entnommenen Stellen sind als solche einzeln kenntlich gemacht.
Diese Arbeit ist bislang keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht worden.
Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

.....
Andreas Linz
Leipzig, 25. Juni 2013

Danksagungen

Mein Dank gilt Dr. rer. nat. Johannes Waldmann für die Betreuung der Arbeit, sowie Spreadshirt und insbesondere Jens Hadlich für die Unterstützung.

Ohne das Korrekturlesen von Elisa Jentsch hätte ich so manchem Fehler nicht entdeckt, dafür auch noch einmal vielen Dank.

Abstract

Schlüsselwörter

Codegenerierung, RESTful Web Service, Modellierung, Client-Bibliothek, Spreadshirt-API, Polyglot

Diese Seite wurde mit Absicht leer gelassen.

Inhaltsverzeichnis

Diese Seite wurde mit Absicht leer gelassen.

1 Einführung

> „Essentially, all models are wrong, but some are useful."
>
> *Empirical Model-Building and Response Surfaces. p. 424*
> ***George E. P. Box, Norman R. Draper** (1987)*

Die zwei wichtigsten Konstanten in der Anwendungsentwicklung sind laut [Her03] folgende:

- Die Zeit eines Programmierers ist kostbar
- Programmierer mögen keine langweiligen und repetitiven Aufgaben

Codegenerierung greift bei beiden Punkten an und kann zu einer Steigerung der *Produktivität* führen, die mit durch herkömmliches schreiben von Code nicht zu erreichen wäre. Änderungen können an zentraler Stelle vorgenommen und durch die Generierung automatisch in den Code übertragen werden, was mit verbesserter *Wartbarkeit* einhergeht. Die gewonnenen Freiräume kann der Entwickler nutzen um sich mit den Grundlegenden Herausforderungen und Problemen seiner Software zu beschäftigen. Durch die Festlegung eines Schemas für Variablennamen und Funktionssignaturen, wird eine hohe *Konsistenz*, über die gesamte Codebasis hinweg, erreicht. Diese Einheitlichkeit vereinfacht auch die Nutzung des Generats[1], da beispielsweise nicht mit Überraschungen bei den verwendeten Bezeichnern zu rechnen ist. Als Eingabe für den Generator dient ein *abstraktes Modell* des betreffenden Geschäftsbereiches. Die Erstellung eines solchen Modells vertieft das Verständnis des Entwicklers für das Geschäftsfeld und gibt gleichzeitig Spezialisten aus dem Fachbereich die Möglichkeit Fragestellungen anhand dieses Modells zu formulieren. Codegene-

[1] Ergebnis des Codegenerierungsvorganges

rierung kann auch ein geeignetes Mittel für die nötige Effizienzsteigerung zu sein, um immer kürzere Entwicklungszyklen einzuhalten.

1.1 Was ist Spreadshirt?

Abbildung 1.1: Spreadshirt Logo

Spreadshirt ist eines der führenden Unternehmen für personalisierte Kleidung und zählt zu den *Social Commerce*[2]-Unternehmen. Es gibt Standorte in Europa und Nordamerika, der Hauptsitz ist in Leipzig. Den Nutzern wird eine Online-Plattform geboten um Kleidungsstücke selber zu gestalten oder zu kaufen, oder auch um eigene Designs, als Motiv oder in Form von Produkten, zum Verkauf anzubieten. Es wird jedem Nutzer ermöglicht einen eigenen Shop auf der Plattform zu eröffnen und ihn auf der eigenen Internetseite einzubinden. Derzeit gibt es rund 400.000 Spreadshirt-Shops mit ca. 33.000.000 Produkten. Für die Spreadshirt-API können Kunden eigene Anwendungen schreiben, bspw. zufallsshirt.de [3] oder soundslikecotton.com [4] . Neben dem Endkunden- bedient Spreadshirt auch das Großkundengeschäft als Anbieter von Druckleistungen.

Die *sprd.net AG* zu der auch der Leipziger Hauptsitz gehört beschäftigt derzeit[5] 178 Mitarbeiter, davon 29 in der IT.

[2] Handelsunternehmen bei dem die aktive Beteiligung und persönliche Beziehung, sowie Kommunikation der Kunden untereinander, im Vordergrund stehen.
[3] http://zufallsshirt.de/
[4] http://www.soundslikecotton.com/
[5] Stand Juni 2013

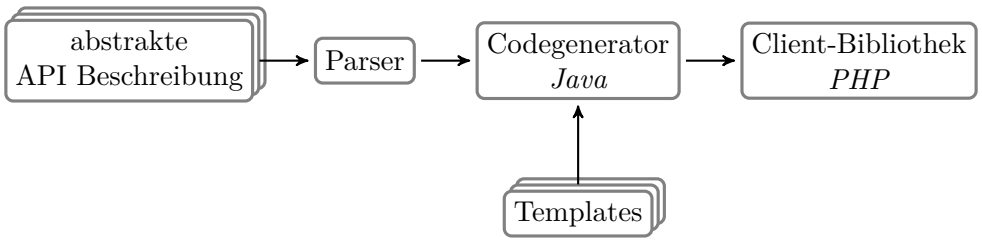

Abbildung 1.2: Aufbau des Generatorsystems

1.2 Ziel der Arbeit

Es ist ein Codegenerator zu erstellen, der aus der abstrakten Beschreibung der RESTful Spreadshirt API eine Client-Bibliothek erstellt.

Der Generator soll eine flexible Wahl der Zielsprache bieten, wobei mit „Zielsprache“ im folgenden die Programmiersprache der erzeugten Bibliothek gemeint ist. Für das Bibliotheksdesign ist eine DSL (Domain-Specific Language) zu realisieren, mit dem Ziel die Nutzung der API zu vereinfachen.

Als Programmiersprache für den Generator wird *Java* verwendet, als Zielsprache der Bibliothek dient *PHP*. Um die gewünschte Flexibilität bezüglich der Zielsprache zu erreichen, wird eine Template-Engine verwendet.

2 Grundlagen

„The purpose of computing is insight, not numbers.“

Numerical Methods for Scientists and Engineers. Preface
***Richard Hamming** (1962)*

Dieses Kapitel dient der Begriffsklärung und der Erläuterung weiterer Grundlagen die zum Verständnis dieser Arbeit notwendig sind. Es folgt eine Einführung in REST sowie der Beschreibungsformate WADL und XSD. Außerdem werden verschiedene Codegenerierungsformen vorgestellt.

2.1 XML

Listing 2.1: ”EBNF eines XML-Elementes”

```
Element ::= "<" , Tagname , {Attribute , Whitespace} , ">" ,
                Element ,
            "</" , Tagname , ">"
            | "<" , Tagname , "/>";
Attribute ::= Key , "=" , Value | "";
Tagname ::= {UTF8-character};
Key ::= {UTF8-character};
Value ::= {UTF8-character};
```

Listing 2.2: ”Minimalbeispiel für eine XML-Datei”

```
<xml version="1.0" encoding="UTF-8" ?>
<tagname key="value">
    <emptyElement />
</tagname>
```

2.2 RESTful Web Service

Representational State Transfer (deutsch: „Gegenständlicher Zustandstransfer“) ist ein Softwarearchitekturstil für Webanwendungen, welcher von Roy Fielding erstmals beschrieben wurde [Fie00]. Als RESTful bezeichnet man dabei eine Webanwendung die den Prinzipien von REST entspricht. Die Daten liegen dabei in eindeutig addressierbaren sogenannten *resources* vor. *Resource* bezeichnet jede Art von Entität, die durch die API bereitgestellt wird. Die Interaktion basiert auf dem Austausch von *representations* – also ein Dokument was den aktuellen oder gewünschten Zustand einer resource beschreibt. Beispiel-URL für das Item *84* aus dem Warenkorb *42*:
`http://api.spreadshirt.net/api/v1/baskets/84/item/42`

2.3 WADL

Die *Web Application Description Language* (kurz WADL) ist eine maschinenlesbare Beschreibung einer HTTP-basierten Webanwendung, einschließlich einer Menge von*XML Schematas* [Had06]. Die aktuelle Revision ist vom 31. August 2009 [1] , im weiteren beziehe ich mich aber hier auf die in der Spreadshirt-API verwendeten Version vom 9. November 2006[2].

Die Beschreibung eines Webservices durch WADL besteht nach [Had06] im groben aus den folgenden vier Bestandteilen:

Set of resources Analog einer Sitemap, die Übersicht aller verfügbaren Ressourcen

Relationships between resources Die kausale und referentielle Verknüpfung zwischen Ressourcen

Methods that can be applied to each resource Die von der jeweiligen Ressource unterstützten HTTP-Methoden, deren Ein- und Ausgabe, sowie die unterstützden Formate

[1] http://www.w3.org/Submission/wadl/
[2] Die Unterschiede zwischen beiden Revisionen können unter der folgenden URL nachvollzogen werden `http://www.w3.org/Submission/wadl/#x3-41000D.1`.

Resource representation formats Die unterstützten MIME-Typen und verwendeten Datenschemas (Abschnitt 2.4.1)

Listing 2.3: Beispielaufbau einer WADL-Datei anhand der Spreadshirt-API Beschreibung

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?> ❶
<application xmlns="http://research.sun.com/wadl/2006/10"> ❷
    <grammars> ❸
        <include href="http://api.spreadshirt.net/api/v1/metaData/api.xsd">
            <doc>Catalog XML Schema.</doc>
        </include>
        ...
    </grammars>
    <resources base="http://api.spreadshirt.net/api/v1/"> ❹
        <resource path="users/{userId}"> ❺
            <doc>Return user data.</doc>
            <method name="GET"> ❻
                <doc>...</doc>
                <request> ❼
                    <param
                        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                        name="mediaType"
                        style="query"
                        type="xsd:string">
                    <doc>...</doc>
                    </param>
                    ...
                </request>
                <response> ❽
                    <representation
                        xmlns:sns="http://api.spreadshirt.net"
                        element="sns:user"
                        status="200"
                        mediaType="application/xml">
                    <doc title="Success"/>
                    </representation>
                    <fault status="500" mediaType="text/plain">
                        <doc title="Internal␣Server␣Error"/>
                    </fault>
                        ...
                        }
                ...
```

Die Datei beginnt mit der Angabe der XML-Deklaration ❶. Die Attribu-

te des Wurzelknotens `<application>` enthalten *namespace* Definitionen, u. a. auch den der verwendeten WADL-Spezifikation ❷. Innerhalb des `<grammars>` Elements werden die benutzten *XML Schemas* angegeben ❸. Um die Ressourcen der Webanwendung ansprechen zu können wird noch die Angabe der Basisadresse benötigt ❹. Innerhalb des `<resources>` Elements findet sich Beschreibung der einzelnen Ressourcen. Diese sind gekennzeichnet, durch eine zur Basisadresse relativen URI ❺. In `{…}` eingeschlossene Teile einer URI, werden durch den Wert des gleichnamigen *request* Parameters ersetzt um die URI zu bilden (generative URIs). Im Folgenden werden die von der Ressource unterstützten HTTP-Methoden beschrieben ❻, deren Anfrageparameter `<request>` ❼, sowie die möglichen Ausgaben der Methode `<response>` ❽.

Die Dokumentations-Tags `<doc>` sind für alle XML-Elemente optional. Um das Listing nicht unnötig zu verlängern habe ich die schließenden *Tags* weggelassen.

Kapitel 2 von [Had06] beschreibt die Elemente im Detail.

2.4 Document Type Definition (DTD)

2.4.1 XML Schema Description (XSD)

2.4.2 RelaxNG

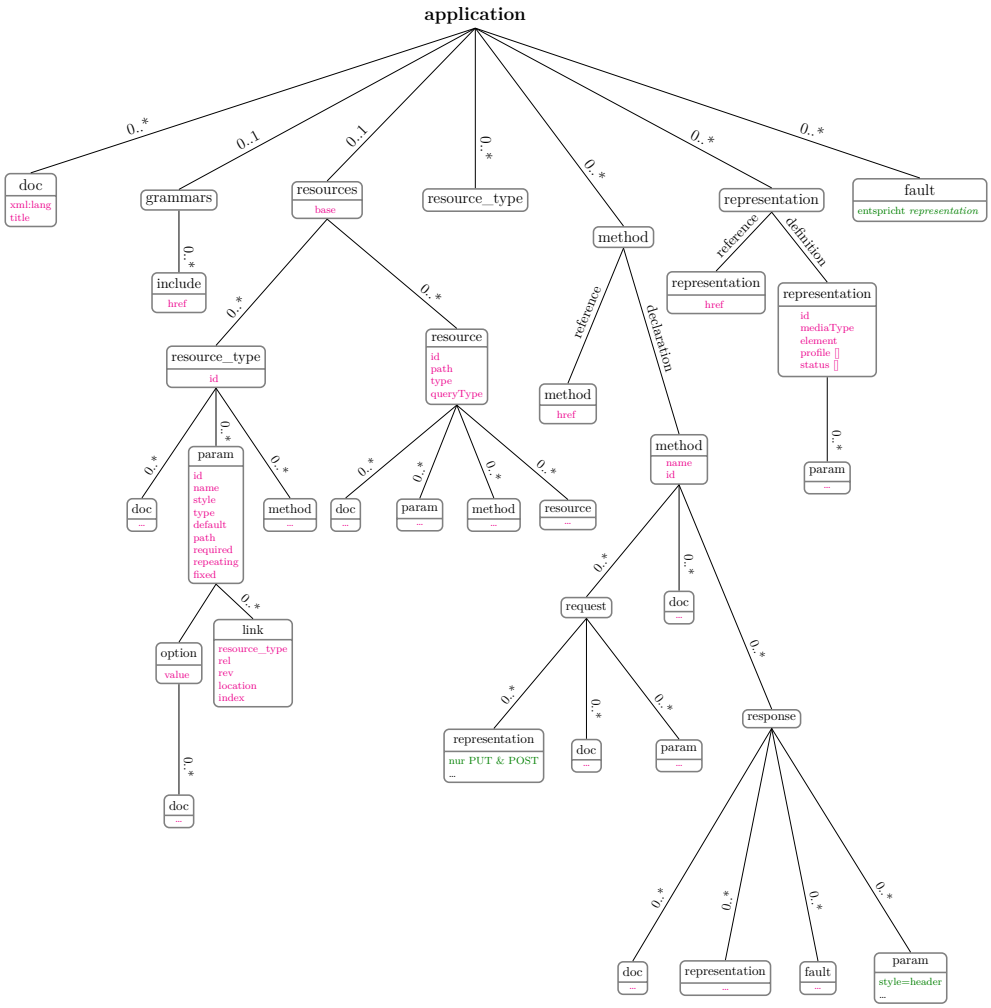

Abbildung 2.1: Struktur einer WADL-Datei, nach Kapitel 2 [Had06]

3 Implementierung

4 Zusammenfassung

4.1 Fazit

4.2 Ausblick

Glossar

API
: *Application Programming Interface* (deutsch: „Schnittstelle zur Anwendungsprogrammierung“) spezifiziert wie Softwarekomponenten über diese Schnittstelle miteinander interagieren können . 3, A

DSL
: *Domain Specific Language* (deutsch: „Domänenspezifische Sprache“) ist eine Programmiersprache die nur auf eine bestimmte Domäne oder auch Problembereich optimiert ist. . 3, A

DTD
: *Document Type Definition*, manchmal auch *Data Type Definition*, ist eine Menge von Angaben die einen Dokumenttyp beschreiben. Es werden konkret Element- und Attributtypen, Entitäten und deren Struktur beschrieben. Die bekanntesten Schemasprachen für XML-Dokumente sind XSD und RelaxNG. A, *siehe* XSD

JSON
: *JavaScript Object Notation* ist ein Mensch- und Maschinenlesbares Format zu Codierung und Austausch von Daten. Bietet im Gegensatz zu XML keine Erweiterbarkeit und Unterstützung für Namesräume, ist aber kompakter und einfacher zu parsen. A, *siehe* XML

Metaprogramming
: beschreibt das erstellen von Programmen welche sich selbst, oder andere Programme, modifizieren oder die einen Teil des Kompilierungsschrittes übernehmen (bspw. der C-Präprozessor) . A

MIME
: *Multipurpose Internet Mail Extensions* dienen zu Deklaration von Inhalten (Typ des Inhalts) in verschiedenen Internetprotokollen. . 6, A

Polyglot
: *mehrsprachig* . A

REST

Representational State Transfer (deutsch: „Gegenständlicher Zustandstransfer“) ist ein Softwarearchitekturstil für Webanwendungen, welcher von Roy Fielding in seiner Dissertation [1] beschrieben wurde. Die Daten liegen dabei in eindeutig addressierbaren *resources* vor. Die Interaktion basiert auf dem Austausch von *representations* – also ein Dokument was den aktuellen oder gewünschten Zustand einer resource beschreibt. Beispiel-URL für das Item *84* aus dem Warenkorb *42*: `http://api.spreadshirt.net/api/v1/baskets/84/item/42` . 4, 5, A

RESTful

Als *RESTful* bezeichnet man einen Webservice der den Prinzipien von REST entspricht. 3, 5, A, *siehe* REST

Template-Engine

Eine *Template-Engine* ersetzt markierte Bereiche in einer Template-Datei (i. Allg. Textdateien) nach vorgegebenen Regeln . 3, A

URI

Unified Resource Identifier ist ein Folge von Zeichen, die einen Name oder eine Web-Ressource identifiziert. . 7, A

URL

Unified Resource Locator sind eine Untermenge der *URIs*. Der Unterschied besteht in der expliziten Angabe des Zugrissmechanismus und des Ortes („Location“) durch *URLs*, bspw. `http` oder `ftp`. A, *siehe* URI

WADL

Web Application Description Language ist eine maschinenlesbare Beschreibung einer HTTP-basierten Webanwendung. 4, 5, A, *siehe* XML

XML

Extensible Markup Language (deutsch: „erweiterbare Auszeichnungssprache“) ist ein Mensch- und Maschinenlesbares Format für Codierung und Austausch von Daten, spezifiziert vom W3C [2] . A

[1] http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf
[2] http://www.w3.org/TR/REC-xml

XSD

XML Schema Description enthält Regeln für den Aufbau und zum Validieren einer XML-Datei. 4, A, *siehe* XML

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

[CE00] K. Czarnecki und U. Eisenecker. *Generative programming: methods, tools, and applications*. Addison Wesley, 2000. ISBN: 9780201309775. URL: http://books.google.de/books?id=cCZXYQ6Pau4C.

[Fie00] Roy Thomas Fielding. „Architectural styles and the design of network-based software architectures“. AAI9980887. Diss. 2000. ISBN: 0-599-87118-0.

[Fow10] M. Fowler. *Domain-Specific Languages*. Addison-Wesley Signature Series (Fowler). Pearson Education, 2010. ISBN: 9780131392809. URL: http://books.google.de/books?id=ri1muolw_YwC.

[Had06] Sun Microsystems Inc. Hadley Marc J. *Web Application Description Language (WADL)*. abgerufen am 21.06.2013. 9. Nov. 2006. URL: https://wadl.java.net/wadl20061109.pdf.

[Her03] J. Herrington. *Code Generation in Action*. In Action Series. Manning, 2003. ISBN: 9781930110977. URL: http://books.google.de/books?id=VHVC8WnSgbYC.

[KK06] M. Klar und S. Klar. *Einfach generieren: Generative Programmierung verständlich und praxisnah*. Hanser Fachbuchverlag, 2006. ISBN: 9783446404489. URL: http://books.google.de/books?id=6LS7OwAACAAJ.

[KT08] S. Kelly und J.P. Tolvanen. *Domain-Specific Modeling: Enabling Full Code Generation*. Wiley, 2008. ISBN: 9780470249253. URL: http://books.google.de/books?id=GFFtRFkuU_AC.