

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

## Bachelorverteidigung

Andreas Linz

HTWK - Fakultät für Informatik, Mathematik & Naturwissenschaften

21. Oktober 2013

# Inhaltsverzeichnis

## Einleitung

- Aufgabe
- Anforderungen
- Spreadshirt
- Spreadshirt-API

## Hauptteil

- Web Services

- Dokumentbeschreibungssprachen
- Codegenerierung
- Datenmodelle & Codegenerator
- Client-Bibliothek

## Zusammenfassung

- Ausblick
- Diskussion

# Aufgabe

## Was?

Client-Bibliothek aus abstrakter Beschreibung eines RESTful Web Service erzeugen.

## Warum?

- ▶ Vereinheitlichung bestehender Implementierungen
- ▶ Nutzung der API für externe Entwickler erleichtern (z.B. Authentifizierung kapseln)

# Anforderungen

- ▶ Austauschbarkeit der Zielsprache
- ▶ einfache Bedienbarkeit der Bibliothek
- ▶ gute Lesbarkeit des erzeugten Codes
- ▶ größtmögliche Typsicherheit des erzeugten Codes
- ▶ hohe Testabdeckung
- ▶ vollständige Generierung der Methoden aus der API-Beschreibung

# Spreadshirt

- ▶ führendes Unternehmen für *personalisierte Bekleidung*
- ▶ *Social-Commerce* (Interaktion mit Kunden steht im Vordergrund)
- ▶ Standorte in Europa & Nordamerika, HQ in Leipzig
- ▶  $\approx 450$  Mitarbeiter, 50 in der IT
- ▶  $4 * 10^5$  Spreadshirt-Shops mit  $33 * 10^6$  Produkten

- ▶ Online-Plattform um Kleidungsstücke und Accessoire mehr selbst zu:
  - ▶ gestalten
  - ▶ kaufen
  - ▶ zum Verkauf anbieten (eigene Designs als Motiv oder Produkt)

## Spreadshirt-API

- ▶ API erlaubt Entwicklern die Nutzung eines großen Teils der Funktionen der Online-Plattform in eigenen Applikationen
- ▶ u.a. Produkt Erstellung, Design Upload & Warenkorbverwaltung
- ▶ Erstellen eigener Shops und kundenspezifischer Anwendungen

### Zufallsshirt – Das nächste Shirt ist immer das beste

S	M	L
XL	XXL	3XL
4XL	5XL	

MOTIV  
(25 x 14 cm)  
[Screenshots](#)

Alles erklärt Deutsch

Neues Zufallsshirt!

Dieses Shirt jetzt kaufen für nur 20,40 €!  
Kommt nie wieder!

[Hilfreiche Bestellhinweise](#)

Schon die alten Kelten haben dieses bequeme Kleidungsstück geschätzt. Ideal fürs Anziehen im Morgengrauen - beim Kombinieren kann man einfach nichts falsch machen. Etwas für die Seele.

FRAUEN MÄNNER KINDER ETC

[Facebook Fan werden](#) (bedeutsame Nachrichten von neuen Motivsorten)  
Oder [Twitter Follower](#) (schöne Wörter)

[Impressum](#)

Abbildung: Beispiel für kundenspezifische Anwendung: zufallsshirt.de

# RESTful Web Service

## REST

- ▶ *Representational State Transfer* (Gegenständlicher Zustandstransfer)
- ▶ Softwarearchitekturstil für Webanwendungen
- ▶ Anwendungen bestehen aus *Ressourcen* mit eindeutigem Bezeichner (Abbildung 2)
- ▶ Zustand einer Ressource ist eine *Repräsentation*
- ▶ Aktionen mit einer REST-API über den Austausch von Repräsentationen



*http://api.spreadshirt.net/api/v1/*  
*Basis-URL*

*baskets/84/item/42*  
*Warenkorb* *Artikel*  
*Ressource*

**Abbildung:** Beispiel-URI, um den Artikel 42 aus dem Warenkorb 84 anzusprechen

*RESTful Web Service* ist eine Webanwendung die den REST Prinzipien entspricht

# Dokumentbeschreibungssprachen

## WADL

maschinenlesbare Beschreibung einer HTTP-basierten Webanwendung

## XSD

Dokumentbeschreibungssprache zur Definition von Datentypen

Gemeinsamkeiten:

- ▶ XML-Syntax, selbst wieder gültige XML-Dokumente
- ▶ Baumgrammatiken

# Codegenerierung

## Codegenerator

Programm welches aus einer *höhersprachigen Spezifikation*, einer Software oder eines Teilaspektes, die *Implementierung erzeugt*

## Vorteile

- ▶ Produktivitätssteigerung
- ▶ hohe Konsistenz des Generats
- ▶ zentrale Stelle für Änderungen (Eingabemodell)

# Generatorformen

## Klassifikation nach Generierungsmenge

- ▶ teilweise
  - ▶ Inline-Code Expander
  - ▶ Mixed-Code Generator
  - ▶ Partial-Class Generator
- ▶ vollständig (Tier<sup>1</sup>-Generator)
- ▶ mehrfach (n-Tier Generator)

---

<sup>1</sup>Stufen

# Datenmodelle

## Eingabe des Generators

- ▶ Applikationsmodell:
  - ▶ WADL → REST-Modell
  - ▶ XSD → Schema-Modell
- ▶ Sprachenmodell
  - ▶ kapselt Zielsprache
  - ▶ enthält Semantik
  - ! Syntax in Ausgabemodul (LanguageVisitor, siehe Abbildung 3)

# Generator für die Spreadshirt-API

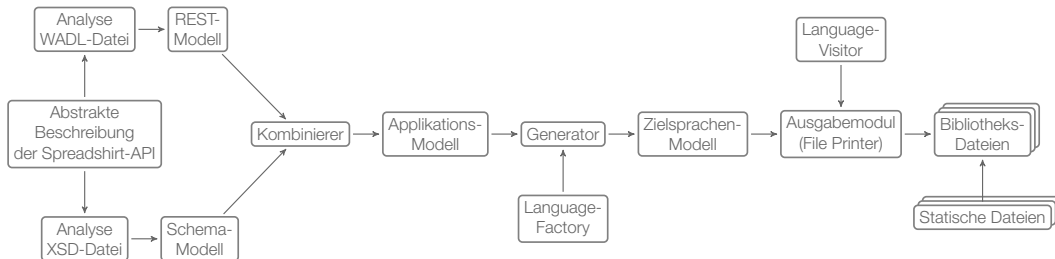


Abbildung: Sequenzdiagramm des Generators für die Spreadshirt-API

# Datenklassen

- ▶ zielsprachenabhängige Repräsentation der Typen aus der XML-Schema Beschreibung
- ▶ Variablen bilden die Attribute und Elemente aus dem Schematyp ab
- ▶ Getter- und Setter-Methoden für alle Variablen
- ▶ transportunabhängiger Datenaustausch mit API → Methoden zur Serialisierung und Deserialisierung

```
1  <?php
2      require_once( 'Unit.php' );
3
4      class Point
5      {
6          private $unit; // unit
7          private $y; // double
8          private $x; // double
9
10         function __construct(
11             /* double */ $y,
12             /* double */ $x
13         )
14         {
15             $this->y = $y;
16             $this->x = $x;
17         }
18
19         public function setUnit(
20             /* unit */ $unit
21         )
22         {
23             $this->unit = $unit;
24         }
25         ...
```



```

1      ...
2      public function toXML()
3      {
4          $xml = new SimpleXMLElement(/* Point */ '<Login xmlns="http://api.spreadshirt.net"/>');
5          $xml->addChild(/* string */ 'unit',/* unit */ $this->unit);
6          $xml->addChild(/* string */ 'y',/* double */ $this->y);
7          $xml->addChild(/* string */ 'x',/* double */ $this->x);
8          return $xml->asXML();
9      }
10     public static function fromXML(
11         /* SimpleXMLElement */ $xml
12     )
13     {
14         $unit = Unit::fromXML(/* SimpleXMLElement */ $xml->unit);
15         $y = $xml->y;
16         $x = $xml->x;
17         ...
18     }
19     ...
20     public function getX()
21     {
22         return $x = $this->x;
23     }
24 }
25 ?>
    
```

# Ressourcenklassen

- ▶ zielsprachenabhängige Abbildung der Ressourcenbeschreibungen aus WADL-Datei
- ▶ Methoden der Klassen entsprechen den Methoden der abgebildeten Ressource

```

1  <?php
2      require_once('Static/methods.php');
3      require_once('Static/apiUser.php');
4      /* Create or list products for user. */
5      class UsersUserIdProducts
6      {
7          private $baseUrl = 'http://192.168.13.10:8080/api/v1/'; // string
8          ...
9          /* */
10         public function POST(
11             /* array */ $parameters,
12             /* ApiUser */ $apiUser,
13             /* ProductDTO */ $productDTO
14         )
15         { ... }
16         ...
17         function __construct(
18             /* string */ $userId
19         )
20         {
21             $this->userId = $userId;
22             $this->resourceUrl = $this->baseUrl . 'users' . '/' . $userId . 'products';
23         }
24     }
25  ?>

```

# statische Klassen

- ▶ wurden manuell erstellt
- ▶ enthalten gemeinsam genutzten Code ohne variable Bestandteile
- ▶ Bibliothek enthält zwei dieser Klassen:
  - ▶ Kommunikation über HTTP-Methoden mit der API
  - ▶ Kapselung der Authentifizierung

# Zusammenfassung

- ▶ Entwicklung der Datenmodelle
- ▶ Überführung der Beschreibung in Eingabedatenmodelle des Generators
- ▶ Erstellung und Implementierung des Sprachenmodells für PHP
- ▶ Implementierung des Generators
- ▶ Generierung der Bibliothek

# Ausblick

- ▶ Parameterobjekte
- ▶ Fluent-Interface
- ▶ Java-Bibliothek (Sprachenmodell)
- ▶ Erzeugung von Dokumentation und Testdaten

# Diskussion

- ▶ XSD, WADL
- ▶ RESTful Web Service
- ▶ Datenmodelle für Web Service Beschreibung und Programmiersprache
- ▶ Stufen-Generator
- ▶ PHP Client-Bibliothek