

# Multimedia im Netz

Wintersemester 2013/14

Übung 08 (Hauptfach)

# Anmeldung zur Klausur

- Die Anmeldung zur Klausur ist ab sofort über Uniworx möglich.
- Anmeldung bis: **12.02.2014 (23:00 Uhr)**
- Abmeldung bis: **17.02.2014 (12:00 Uhr)**
  
- Nur Leute die angemeldet sind, können an der Klausur teilnehmen.
- Sollte man nicht an der Klausur teilnehmen (obwohl man sich angemeldet hat) muss sich wieder von der Klausur abmelden!

# Mashups

- Zusammenstellung multimedialer Inhalte zu einem bestimmten Thema
- Bereitstellung der Inhalte durch Webanwendungen über Web-APIs
- Parsen der zurückgelieferten XML-Dateien
- Angepasste Anzeige der Informationen durch HTML
- Beispiele für Mashups:  
<http://www.programmableweb.com/mashups>

# Mögliche Webdienste



- Weitere Beispiele:

<http://www.programmableweb.com/apis/directory/1?sort=mashups>

# API-Schlüssel

- Um die APIs mancher Dienste in Anspruch zu nehmen, werden sogenannte API-Schlüssel benötigt (z.B. Flickr)

<p>Wählen Sie <b>nicht-kommerziell</b> aus, wenn:</p> <ul style="list-style-type: none"><li>• Mit Ihrer Anwendung kein Geld verdient wird.</li><li>• Ihre Anwendung bringt zwar Geld ein, Sie haben jedoch ein familiengeführtes, kleines oder unabhängiges Unternehmen.</li><li>• Sie entwickeln ein derzeit nicht kommerzielles Produkt, dies könnte sich jedoch in Zukunft ändern.</li><li>• Sie erstellen eine private Website oder einen Blog, für die Sie nur Ihre eigenen Bilder verwenden.</li></ul> <p><b>NICHT KOMMERZIELLEN SCHLÜSSEL BEANTRAGEN</b></p>	oder	<p>Wählen Sie <b>kommerziell</b> aus, wenn:</p> <ul style="list-style-type: none"><li>• Sie oder Ihr Büro arbeiten für eine bekannte Marke.</li></ul> <p>UND einer der folgenden Punkte ist zutreffend:</p> <ul style="list-style-type: none"><li>• Sie möchten Gewinne erzielen.</li><li>• Sie erheben eine Gebühr für Ihr Produkt oder Ihre Dienstleistungen.</li><li>• Sie beziehen Flickr Inhalte in Ihr Produkt ein und möchten diese Dienstleistungen verkaufen.</li></ul> <p><b>KOMMERZIELLEN SCHLÜSSEL BEANTRAGEN</b></p>
---	------	---

- Der Schlüssel muss dann bei jeder Anfrage mit angegeben werden
- Flickr: [http://www.flickr.com/services/api/misc.api\\_keys.html](http://www.flickr.com/services/api/misc.api_keys.html)

# OAuth (<http://oauth.net/>)

- Protokoll, das eine standardisierte API-Autorisierung ermöglicht
- Stellt zum Beispiel sicher, dass eine Anwendung nur im Namen des Nutzers interagieren darf, wenn diese vom Nutzer genehmigt worden ist.
- Beispiel: Twitter (siehe nächste Folie)

# Twitter und OAuth

- Es gibt zwei Authentifizierungsvarianten in Twitter
  - Application-user Authentifizierung
    - Die Anwendung agiert im Namen eines Nutzers
    - Es kann überprüft werden, ob die Anwendung die Erlaubnis hat, im Namen des Nutzers zu agieren.
  - Application-only Authentifizierung
    - Die Anwendung hat keinen Nutzer-kontext
    - Wird zum Beispiel verwendet, um auf öffentliche Informationen in Twitter zuzugreifen
- Es gibt verschiedene Bibliotheken, welche diese Funktionalitäten unterstützen, z.B.:
  - <https://github.com/abraham/twitteroauth> für PHP

# Beispiel: Twitter und Application-user Authentifizierung (1)

## OAuth settings

Your application's OAuth settings. Keep the "Consumer secret" a secret. This key should never be human-readable in your application.

Access level	Read-only <a href="#">About the application permission model</a>
Consumer key	11111111111111111111111111111111
Consumer secret	22222222222222222222222222222222

Request token URL

Authorize URL

Access token URL

Callback URL

Sign in with Twitter  No

## Your access token

Use the access token string as your "oauth\_token" and the access token secret as your "oauth\_token\_secret" to sign requests with your own Twitter account. Do not share your `oauth_token_secret` with anyone.

Access token	33333333333333333333333333333333
Access token secret	44444444444444444444444444444444

Access level  Read-only

# Beispiel: Twitter und Application-user Authentifizierung (2)

```
<?php
require 'twitteroauth.php';
$consumerkey = "111111111111111111111111";
$consumersecret = "222222222222222222222222";
$accesstoken = "333333333333333333333333";
$accesstokensecret = "444444444444444444444444";

$twitter = new TwitterOAuth($consumerkey,
                             $consumersecret,
                             $accesstoken,
                             $accesstokensecret);

$value = $_GET["q"];

$tweets = $twitter->get('https://api.twitter.com/
                        1.1/statuses/
                        user_timeline.json?
                        count=10&screen_name=' . $value);

echo json_encode($tweets);
?>
```

# Exploring the Twitter API

The screenshot shows the Twitter API console interface. At the top, the 'Service' is set to 'https://api.twitter.com/1.1' and 'Authentication' is set to 'No Auth'. The 'Request URL' is 'https://api.twitter.com/1.1/statuses/user\_timeline.json'. Below this, there are tabs for 'Query\*', 'Template', and 'Headers'. The 'Query' tab is active, showing a table with columns 'Parameter', 'Value', and 'Description'. A single parameter 'count' is listed with an empty input field for its value. The description for 'count' states: 'Specifies the number of tweets to try and retrieve, up to a maximum of 200. The value of count is best thought of as a limit to the number of tweets to return because suspended or deleted content is removed after the count has been applied. We include retweets in the count, even if include\_rts is not supplied. It is recommended you always send'. Below the table, there are sections for 'Request' and 'Response'. The 'Response' section contains the text 'Send this request when you're ready'.

Parameter	Value	Description	* Required
count	<input type="text"/>	Specifies the number of tweets to try and retrieve, up to a maximum of 200. The value of count is best thought of as a limit to the number of tweets to return because suspended or deleted content is removed after the count has been applied. We include retweets in the count, even if include_rts is not supplied. It is recommended you always send	

<https://dev.twitter.com/console>

# Beispiel: Spotify

## Web API

The Web API may be used to explore Spotify's music catalogue. Please refer to the following instructions.

## Services

These are the available services.

- [lookup](#)
- [search](#)

## Requests

You may use ordinary HTTP GET messages. The base URL for each API method looks like the following

```
http://ws.spotify.com/service/version/method[.format]?parameters
```

<https://developer.spotify.com/technologies/web-api/>

# Beispiel: Spotify Search

## Beispielanfragen:

- <http://ws.spotify.com/search/1/album?q=foo>
- <http://ws.spotify.com/search/1/album?q=foo&page=2>
- <http://ws.spotify.com/search/1/album.json?q=foo>
- <http://ws.spotify.com/search/1/album.json?q=foo&page=2>

## Response:

- XML
- JSON

<https://developer.spotify.com/technologies/web-api/search/>

# Wiederholung: JSON

- JavaScript Object Notation
- Einfaches Format zum Datenaustausch
- Basiert auf Name-Wert Paaren

- **Beispiel:**

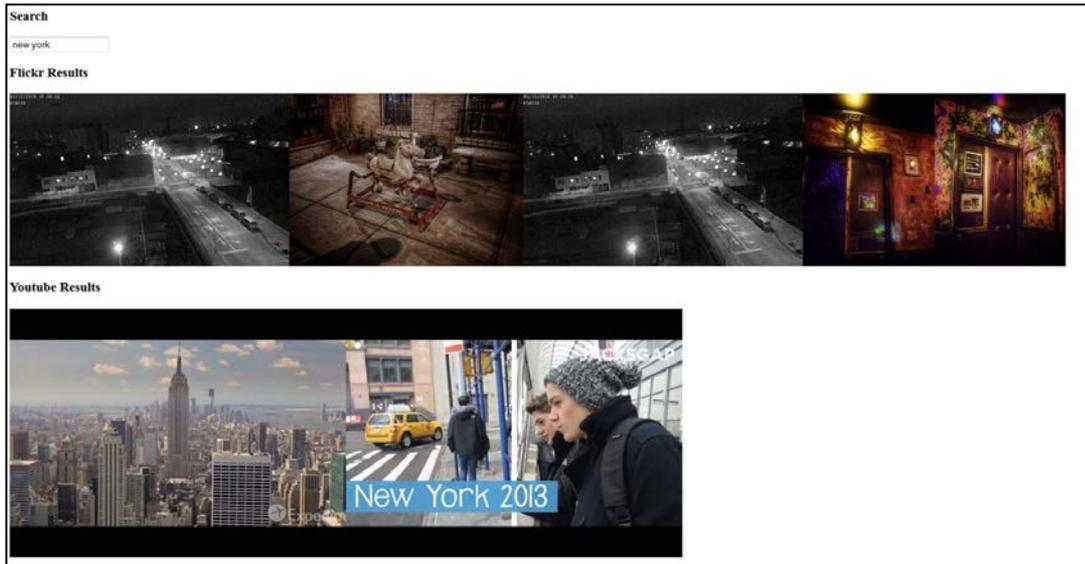
```
{
  "vorname": "Max",
  "nachname": "Mustermann",
  "telefon": [{
    "type": "Home",
    "nummer": "5648978965"
  }, {
    "type": "Mobil",
    "nummer": "6458979878"
  }]
};
```

# Beispiel: Spotify Response

```
{
  - info: {
    num_results: 1,
    limit: 100,
    offset: 0,
    query: "bonjovi",
    type: "album",
    page: 1
  },
  - albums: [
    - {
      name: "Karaoke: BonJovi",
      popularity: "0.28",
      - external-ids: [
        - {
          type: "upc",
          id: "058622069028"
        }
      ],
      href: "spotify:album:0ENiUtq1Hn9aJY2SXIBE3A",
      - artists: [
        - {
          href: "spotify:artist:66BZN51PJ5k6bam5jpNCTF",
          name: "Starlite Karaoke"
        }
      ],
      - availability: {
        territories: "AD AR AT AU BE BG BO BR CA CH CL CO CR CY CZ DE DK DO"
      }
    }
  ]
}
```

# Übungsblatt 8

- **Thema: Mashups mit AJAX**
- **Bearbeitungszeit: 1 Woche**
- **Abgabe: 22.01.2014**



**Danke! Fragen?**