RESTful(ish) APIs

Representational

State

Transfer

RESTful(ish) APIs

- kein Protokoll im engeren Sinne
- Übertragung über HTTP
- jede Ressource ist unter bestimmter URI erreichbar

z.B.: http://example.com/api/articles/23

RESTful(ish) APIs

Auszug aus der RFC 2616 (HTTP/1.1):

```
9   Method Definitions ..................
9.1   Safe and Idempotent Methods ......
9.1.1    Safe Methods ...................
9.1.2    Idempotent Methods .............
9.2   OPTIONS ..........................
9.3   GET ..............................
9.4   HEAD .............................
9.5   POST .............................
9.6   PUT ..............................
9.7   DELETE ...........................
9.8   TRACE ............................
9.9   CONNECT ..........................
```

http://www.ietf.org/rfc/rfc2616.txt

RESTful(ish) APIs

- verschiedene Repräsentationen
  - plain text
  - JSON
  - XML
  - ...
- stateless – zustandslos
- keine Beschreibungssprache (wie WSDL)
- stattdessen lockere Vereinbarungen

RESTful(ish) APIs

Best Practices:

- Ressourcen sollten immer Nomen sein
- Versionen/Repräsentationen nicht mit in URI

http://example.com/v2/articles/3.json

- beides lieber in den Accept-Header:

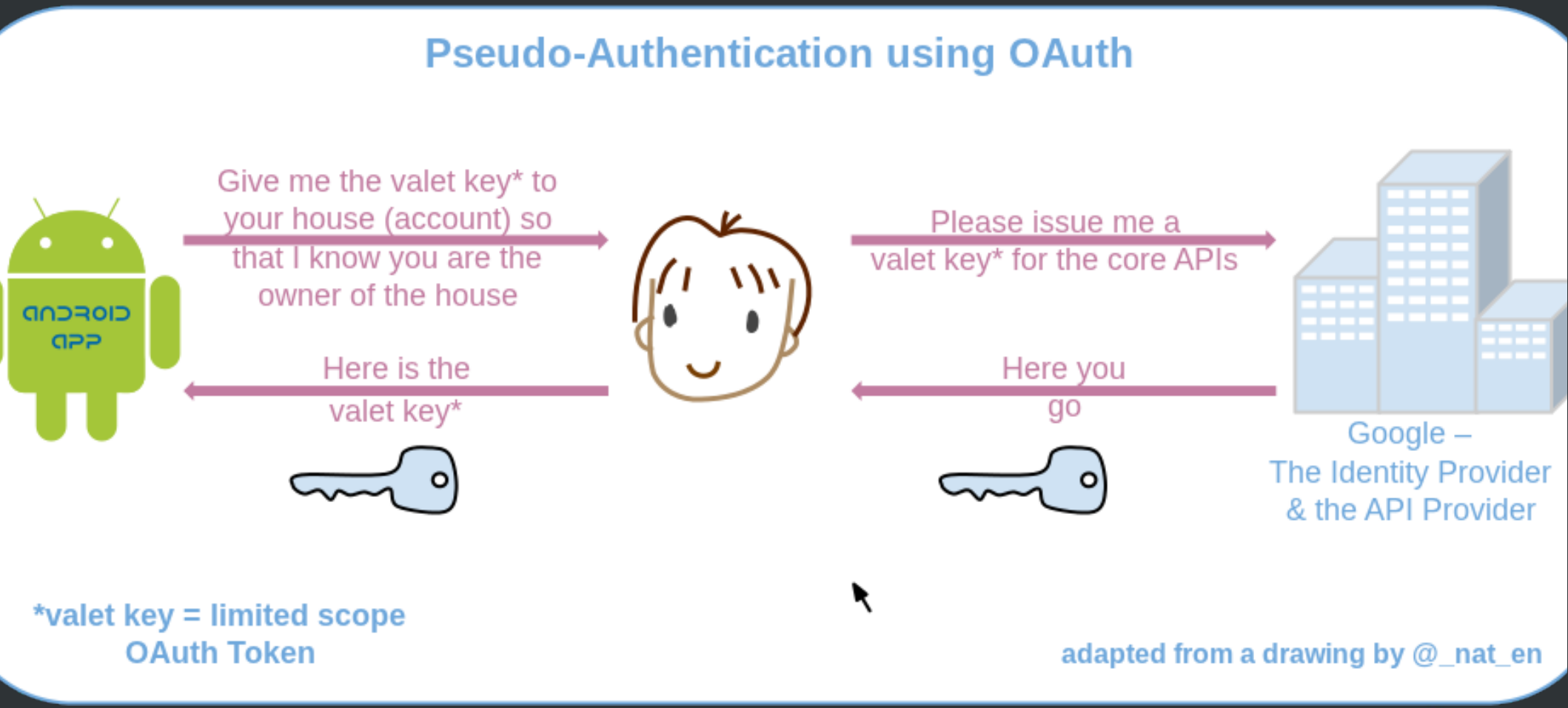
Accept: application/vnd.hickerspace-v2+json

(Stichwort: *Vendor MIME Type*)

RESTful(ish) APIs

- Authentifzierung
  - API-Key als Parameter (SSL!)
  - Basic Auth (SSL!)
  - OAuth

RESTful(ish) APIs



http://en.wikipedia.org/wiki/File:OpenIDvs.Pseudo-AuthenticationusingOAuth.svg

RESTful(ish) APIs

- Wo werden RESTful APIs benutzt?
  - OSM (Clients/Bots)
  - Twitter
  - Facebook
  - Wetter-Services
  - automatische Google-Suchen
  - eBay
  - Amazon Web Services
  - ...

RESTful(ish) APIs

- ..und bei uns?
  - Retweet-Bot (Twitter)
  - MensaHildesheim (Twitter)
  - Twitterdrucker

  - Raumstatus
  - LED-Ticker
  - Audio-Announces
  - Ampel
  - ...

RESTful(ish) APIs

”Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions.”

http://flask.pocoo.org/

RESTful(ish) APIs

Minimales Beispiel:

```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5
6  @app.route('/<name>')
7  def api_hello(name):
8     return 'Hallo %s!' % name
9
10
11 if __name__ == '__main__':
12    app.run(debug=True)
```

RESTful(ish) APIs

Minimales Beispiel:

```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5
6  @app.route('/<name>')
7  def api_hello(name):
8     return 'Hallo %s!' % name
9
10
11 if __name__ == '__main__':
12    app.run(debug=True)
```

RESTful(ish) APIs

Minimales Beispiel:

```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5
6  @app.route('/<name>')
7  def api_hello(name):
8      return 'Hallo %s!' % name
9
10
11 if __name__ == '__main__':
12     app.run(debug=True)
```

RESTful(ish) APIs

Minimales Beispiel:

```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5
6  @app.route('/<name>')
7  def api_hello(name):
8      return 'Hallo %s!' % name
9
10
11 if __name__ == '__main__':
12     app.run(debug=True)
```

RESTful(ish) APIs

Minimales Beispiel:

```
1  from flask import Flask
2
3  app = Flask(__name__)
4
5
6  @app.route('/<name>')
7  def api_hello(name):
8     return 'Hallo %s!' % name
9
10
11 if __name__ == '__main__':
12    app.run(debug=True)
```

RESTful(ish) APIs

```
1  from flask import Flask, request
2  app = Flask(__name__)
3
4  waren = ['Shirt', 'Schuhe', 'Hose']
5
6  @app.route('/waren', methods=['GET', 'POST'])
7  def api_waren():
8      if request.method == 'GET':
9          return "Waren: %s" % waren
10
11     elif request.method == 'POST':
12         neueWare = request.form['name']
13         waren.append(neueWare)
14         index = len(waren) - 1
15         return 'Abgelegt unter id %d.' % index
16
17 @app.route('/waren/<index>')
18 def api_waren_nummer(index):
19     return waren[int(index)]
20
21 if __name__ == '__main__':
22     app.run(debug=True)
```

RESTful(ish) APIs

```
1  from flask import Flask, request
2  app = Flask(__name__)
3
4  waren = ['Shirt', 'Schuhe', 'Hose']
5
6  @app.route('/waren', methods=['GET', 'POST'])
7  def api_waren():
8      if request.method == 'GET':
9          return "Waren: %s" % waren
10
11     elif request.method == 'POST':
12         neueWare = request.form['name']
13         waren.append(neueWare)
14         index = len(waren) - 1
15         return 'Abgelegt unter id %d.' % index
16
17 @app.route('/waren/<index>')
18 def api_waren_nummer(index):
19     return waren[int(index)]
20
21 if __name__ == '__main__':
22     app.run(debug=True)
```

RESTful(ish) APIs

```
1  from flask import Flask, request
2  app = Flask(__name__)
3
4  waren = ['Shirt', 'Schuhe', 'Hose']
5
6  @app.route('/waren', methods=['GET', 'POST'])
7  def api_waren():
8      if request.method == 'GET':
9          return "Waren: %s" % waren
10
11     elif request.method == 'POST':
12         neueWare = request.form['name']
13         waren.append(neueWare)
14         index = len(waren) - 1
15         return 'Abgelegt unter id %d.' % index
16
17 @app.route('/waren/<index>')
18 def api_waren_nummer(index):
19     return waren[int(index)]
20
21 if __name__ == '__main__':
22     app.run(debug=True)
```

RESTful(ish) APIs

```
1  from flask import Flask, request
2  app = Flask(__name__)
3
4  waren = ['Shirt', 'Schuhe', 'Hose']
5
6  @app.route('/waren', methods=['GET', 'POST'])
7  def api_waren():
8      if request.method == 'GET':
9          return "Waren: %s" % waren
10
11     elif request.method == 'POST':
12         neueWare = request.form['name']
13         waren.append(neueWare)
14         index = len(waren) - 1
15         return 'Abgelegt unter id %d.' % index
16
17 @app.route('/waren/<index>')
18 def api_waren_nummer(index):
19     return waren[int(index)]
20
21 if __name__ == '__main__':
22     app.run(debug=True)
```

RESTful(ish) APIs

```
1  from flask import Flask, request
2  app = Flask(__name__)
3
4  waren = ['Shirt', 'Schuhe', 'Hose']
5
6  @app.route('/waren', methods=['GET', 'POST'])
7  def api_waren():
8      if request.method == 'GET':
9          return "Waren: %s" % waren
10
11     elif request.method == 'POST':
12         neueWare = request.form['name']
13         waren.append(neueWare)
14         index = len(waren) - 1
15         return 'Abgelegt unter id %d.' % index
16
17 @app.route('/waren/<index>')
18 def api_waren_nummer(index):
19     return waren[int(index)]
20
21 if __name__ == '__main__':
22     app.run(debug=True)
```

RESTful(ish) APIs

```
1  from flask import Flask, request
2  app = Flask(__name__)
3
4  waren = ['Shirt', 'Schuhe', 'Hose']
5
6  @app.route('/waren', methods=['GET', 'POST'])
7  def api_waren():
8      if request.method == 'GET':
9          return "Waren: %s" % waren
10
11     elif request.method == 'POST':
12         neueWare = request.form['name']
13         waren.append(neueWare)
14         index = len(waren) - 1
15         return 'Abgelegt unter id %d.' % index
16
17 @app.route('/waren/<index>')
18 def api_waren_nummer(index):
19     return waren[int(index)]
20
21 if __name__ == '__main__':
22     app.run(debug=True)
```

RESTful(ish) APIs

```
1  from flask import Flask, request
2  app = Flask(__name__)
3
4  waren = ['Shirt', 'Schuhe', 'Hose']
5
6  @app.route('/waren', methods=['GET', 'POST'])
7  def api_waren():
8      if request.method == 'GET':
9          return "Waren: %s" % waren
10
11     elif request.method == 'POST':
12         neueWare = request.form['name']
13         waren.append(neueWare)
14         index = len(waren) - 1
15         return 'Abgelegt unter id %d.' % index
16
17 @app.route('/waren/<index>')
18 def api_waren_nummer(index):
19     return waren[int(index)]
20
21 if __name__ == '__main__':
22     app.run(debug=True)
```

RESTful(ish) APIs

```
1  from flask import Flask, request
2  app = Flask(__name__)
3
4  waren = ['Shirt', 'Schuhe', 'Hose']
5
6  @app.route('/waren', methods=['GET', 'POST'])
7  def api_waren():
8      if request.method == 'GET':
9          return "Waren: %s" % waren
10
11     elif request.method == 'POST':
12         neueWare = request.form['name']
13         waren.append(neueWare)
14         index = len(waren) - 1
15         return 'Abgelegt unter id %d.' % index
16
17 @app.route('/waren/<index>')
18 def api_waren_nummer(index):
19     return waren[int(index)]
20
21 if __name__ == '__main__':
22     app.run(debug=True)
```

RESTful(ish) APIs

Decorators

```
1 @app.route('/secret-page')
2 @requires_auth
3 def secret_page():
4     return 'Geheim!'
5
```

RESTful(ish) APIs

Decorator: @requires_auth

```
1  def requires_auth(f):
2     @wraps(f)
3     def decorated(*args, **kwargs):
4        auth = request.authorization
5        if not auth or not
6           check_auth(auth.user, auth.pw):
7
8           # returns 401 + text
9           return authenticate()
10
11       return f(*args, **kwargs)
12    return decorated
13
14
```

RESTful(ish) APIs

Decorator: @requires_auth

```
1  def requires_auth(f):
2      @wraps(f)
3      def decorated(*args, **kwargs):
4          auth = request.authorization
5          if not auth or not
6              check_auth(auth.user, auth.pw):
7
8              # returns 401 + text
9              return authenticate()
10
11         return f(*args, **kwargs)
12     return decorated
13
14
```

RESTful(ish) APIs

Decorator: @requires_auth

```
1  def requires_auth(f):
2      @wraps(f)
3      def decorated(*args, **kwargs):
4          auth = request.authorization
5          if not auth or not
6              check_auth(auth.user, auth.pw):
7
8              # returns 401 + text
9              return authenticate()
10
11         return f(*args, **kwargs)
12     return decorated
13
14
```

RESTful(ish) APIs

jsonify & Status Codes

```
1 from flask import Flask, jsonify
2
3 @app.route('/articles')
4 def api_articles():
5     resp = jsonify({'success': False,
6                     'status': 'Method not allowed.'})
7     resp.status_code = 405
8     return resp
```

RESTful(ish) APIs

zurück zur **Hickerspace-API**:

- Rewrite in Flask (Python)..
- ..mit gevent..
- ..läuft auf gunicorn

RESTful(ish) APIs

zurück zur **Hickerspace-API**:

/ledticker	(Schreiben*)	
/ampel	(Schreiben*, Lesen)	
/announce	(Schreiben*)	
/room	(Schreiben*, Lesen)	
/info	(Lesen)	[Space-API-konform]
/muc	(Schreiben*, Lesen)	
/mate-o-meter	(to do)	

* = Authentifizierung nötig

RESTful(ish) APIs

zurück zur **Hickerspace-API**:

/wiki/new

/wiki/userspace

/wiki/updated

/poll/announce (Polling nur authentifiziert)

/poll/ledticker (Polling nur authentifiziert)

/poll/ampel

RESTful(ish) APIs

SpaceAPI

- hackerspaces.nl schlugen Specs vor
- Hackerspaces können API implementieren und werden im SpaceStatusDirectory aufgenommen
- Anwendungen
  - Firefox-Extension
  - Smartphone App
  - Global Hackerspaces Status Wall

http://hackerspaces.me

RESTful(ish) APIs

- Los, schreibt coole APIs!

Weitermachen.

(Ende)