

Neue Features in JEE6: JAX-RS "Jersey"

(JSR-311, RESTful Services in Java)

Klassischer WS-Ansatz

- SOAP-Nachrichten via HTTP-GET
- Art der Methode in der Nachricht selbst
- Vielzahl an Methoden (Prozesssteuerung)
- Große Auswahl an WS-* Specs

Was ist REST?

- HTTP GET, POST, PUT, DELETE
- Ressourcen an URIs
- Nachrichtentyp (Anfrage und Rückgabe) frei wählbar (XML, atom, JSON, txt, HTML)
- Stateless, performant, basic

Einführung JAX-RS Jersey

- REST für Java
- JSR 311
- Referenzimplementierung Jersey
- JEE6 (Glassfish v3)
- Vollständig Annotationsbasiert
- Abbildung via Jersey-Servlet

Beispiel RESTful Service in JAX-RS

```
package de.resttest;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

@Path("/helloworld")
public class RestIt {

    @GET
    @Produces("text/plain")
    public String getStuff() {
        return "Hello World";
    }
}
```

Beispiel RESTful Service in JAX-RS

```
<servlet>
  <servlet-name>Jersey Web Application</servlet-name>
  <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
-  <init-param>
    <param-name>com.sun.jersey.config.property.packages</param-name>
    <param-value>de.resttest</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>Jersey Web Application</servlet-name>
  <url-pattern>/*</url-pattern>
</servlet-mapping>
```

RESTful Client mit Jersey

```
package de.cloutier.twitter;

import javax.ws.rs.core.MultivaluedMap;

import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.core.util.MultivaluedMapImpl;

public class JerseyTest {

    public static void main(String[] args) {
        Client client = Client.create();
        WebResource webResource = client.resource("http://search.twitter.com/search.json");
        MultivaluedMap<String, String> queryParams = new MultivaluedMapImpl();
        queryParams.add("q", "#java");
        webResource.queryParams(queryParams).get(String.class);
    }
}
```

Real World: Twitter API

- Abfragen via GET
- Neue Einträge via POST
- Löschungen via DELETE
- Status, Trends, Suchen, Messages, Account, etc.
- Formate: xml, json, rss, atom

RESTful WS vs. Klassische WS

- Einfach
 - Zustandslos
 - Performant
 - Nahe Integration an HTTP
 - Unabhängig von XML. Auch txt/HTML möglich
- SOAP
 - Frei wählbare Prozesse
 - WS-* möglich: Security, Contract, WSDL

fin