



REST a Java

aneb



Jersey: referenční implementace JAX-RS (JSR 311)

Jakub.Podlesak@Sun.Com

O co jde?

Umožnit jednoduchý vývoj RESTových webových služeb v Javě

Obsah prezentace

- **Co je REST? Proč ho používat?**
- **API & Implementace**
- **Reference**

Roy Fielding a jecho disertace

UNIVERSITY OF CALIFORNIA,
IRVINE

Architectural Styles and the Design of Network-based Software Architectures

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

Roy Thomas Fielding

Dissertation Committee:
Professor Richard N. Taylor, Chair
Professor Mark S. Ackerman
Professor David S. Rosenblum

Přečtěte si tuhle knihu!



REpresentational State Transfer

- REST je architektonický styl
- Sada omezení
- Aplikovaná na architekturu distribuovaného systému

Klíčové koncepty

- Adresy
- Jednotné rozhraní
- Přenos hypermédií

Proč používat REST? (1/2)

- Funguje to!
- Jednoduchost
- Slabé vazby

Proč používat REST? (2/2)

- škálovatelnost
- kompozice & orchestrace



Obsah prezentace

- Co je REST? Proč ho používat?
- **API & Implementace**
- Reference

API: JAX-RS

- Standardizace v JCP: JSR 311
- Serverově orientované
- Bude součástí Java EE 6

JAX-RS: Hlavní cíle

- Základem je POJO a anotace
- HTTP-centrické API
- Nezávislost na běhovém kontejneru a datovém formátu

Varování

- JAX-RS je stále ve vývoji!
- **Může se měnit!**

Vysvětlení na příkladu

- Resource-Oriented Architecture (ROA)
 - > V knize RESTful Web services
 - > REST styl na Webu
- JAX-RS API v kontextu ROA
-

Resource-Oriented Architecture

- Adresovatelnost
- Jednotné rozhraní
- Bezestavovost
- “Prolinkovanost”

Adresovatelnost

- Webová služba se skládá z tzv. “resources”
- Každý “Resource” ma přiřazeno svoje URI
- Motto: Když to nemá URI, tak to není na Webu!

Adresovatelnost v JAX-RS

- Resource je POJO opatřené anotací **@Path**
- **@Path** obsahuje šablonu URI
- **“Root Resource Class”**

```
@Path("/collection")  
class CollectionResource {  
    ...  
}
```

```
@Path("/collection/{eid}")  
class EntryResource {  
    ...  
}
```

Jednotné rozhraní

- “Co se bude dělat” je dáno HTTP metodou
- Každý předem ví, co bude daná metoda dělat nezávisle na konkrétním “resource”

Jednotné rozhraní v JAX-RS

- HTTP metody jako Java metody s anotacemi:

@HEAD

@GET

@PUT

@DELETE

@POST

```
@Path("/collection")
class CollectionResource {
    @GET <type> get() { ... }
    @POST <type> create(<type>) { ... }
}
```

```
@Path("/collection/{eid}")
class EntryResource {
    @GET <type> get(...) { ... }
    @PUT void update(...) { ... }
    @DELETE void delete(...) { ... }
}
```

Ukázka!

- Jak to tedy funguje
- Jak dostat data ven (z Javy)
- Jak dostat data dovnitř (do Javy)

Co když nevracíme “jen” data? (1/2)

```

@Path("/collection")
@ConsumesMime("application/xml")
@ProducesMime("application/xml")
class CollectionResource {
    @GET Col get() { ... }

    @POST Response create(Ent e) {
        // create and persist the new entry
        // create entry resource URI
        URI u = ...
        // build response and return
        return Response.created(u).build();
    }
}

```

Co když nevracíme “jen” data? (2/2)

C: POST /collection HTTP/1.1

C: Host: cols.com

C: Content-Type: application/xml

C: Content-Length: 35

C:

C: <entry><name>myEntry</name></entry>

S: HTTP/1.1 201 Created

S: Location: <http://cols.com/collection/12345>

S: Content-Length: 0

Bezestavovost

- HTTP protokol je bezestavový
- Stav webové služby je dán stavem “resources”
- Za aplikační stav (stav klienta) má zodpovědnost klient

```
@Path("/{collection}/{eid}")
@ConsumesMime("application/xml")
@ProducesMime("application/xml")
class EntryResource {
    String eid;
    EntryResource(@UriParam("eid") String eid)
    { this.eid = eid; }

    @GET Ent get() { ... }

    @PUT void update(Ent e) { ... }

    @DELETE void delete() { ... }
}
```

```
@Path("/{collection}/{eid}")
@ConsumesMime("application/xml")
@ProducesMime("application/xml")
class EntryResource {
    String eid;
    EntryResource(@UriParam("eid") String eid) {
        this.eid = eid;
        if ("eid does not exist")
            // Not found
            throw new WebApplicationException(404);
    }
}
```

“Prolinkovanost”

- Reprezentace “resource” obsahují odkazy na další “resources”
- Hypermédia jsou prostředkem pro změnu stavu aplikace (HTML, XML, JSON mohou obsahovat odkazy)

```
class CollectionResource {  
  
    @POST Response create(  
        @HttpContext UriInfo ui, Ent e) {  
        // get the id of the created entry  
        String eid = ...  
        // build URI  
        URI u = ui.getAbsolutePathBuilder().  
            path(eid).build();  
        // build response and return  
        return Response.created(u).build();  
    }  
  
}
```

Projekt Jersey

- Open source
- Referenční implementace JAX-RS
- Dostupný na java.net

Obsah prezentace

- Co je REST? Proč ho používat?
- API & Implementace
- **Reference**

API: JAX-RS

- <http://jsr311.dev.java.net>
- **Diskuse** expertní skupiny je v režimu pouze pro čtení **dostupná i nečlenům**
 - > Přihlaste se na: dev@jsr311.dev.java.net

Implementace: Jersey

- Open Source
 - > <http://jersey.dev.java.net>
- Dev and user e-mailové skupiny
 - > dev@jersey.dev.java.net, users@jersey.dev.java.net
- Early access downloads
 - > <https://jersey.dev.java.net/servlets/ProjectDocumentList>
 - > Poslední a stabilní buildy

Další informace

- Blogy
 - > <http://blogs.sun.com/sandoz/>
 - > <http://blogs.sun.com/japod/>
 - > <http://weblogs.java.net/blog/mhadley/>
 - > <http://blogs.sun.com/theaquarium/>
- Jersey příklady
 - > Podívejte se do examples adresáře v distribuci
- Otázky?
 - > Ptejte se na users@jersey.dev.java.net

Q&A

- <http://blogs.sun.com/japod>

Děkuji vám za pozornost!