OOIT.com

RemoteQuery

Rapid services for web and mobile

Toni Bünter, OOIT.com AG © März 2014

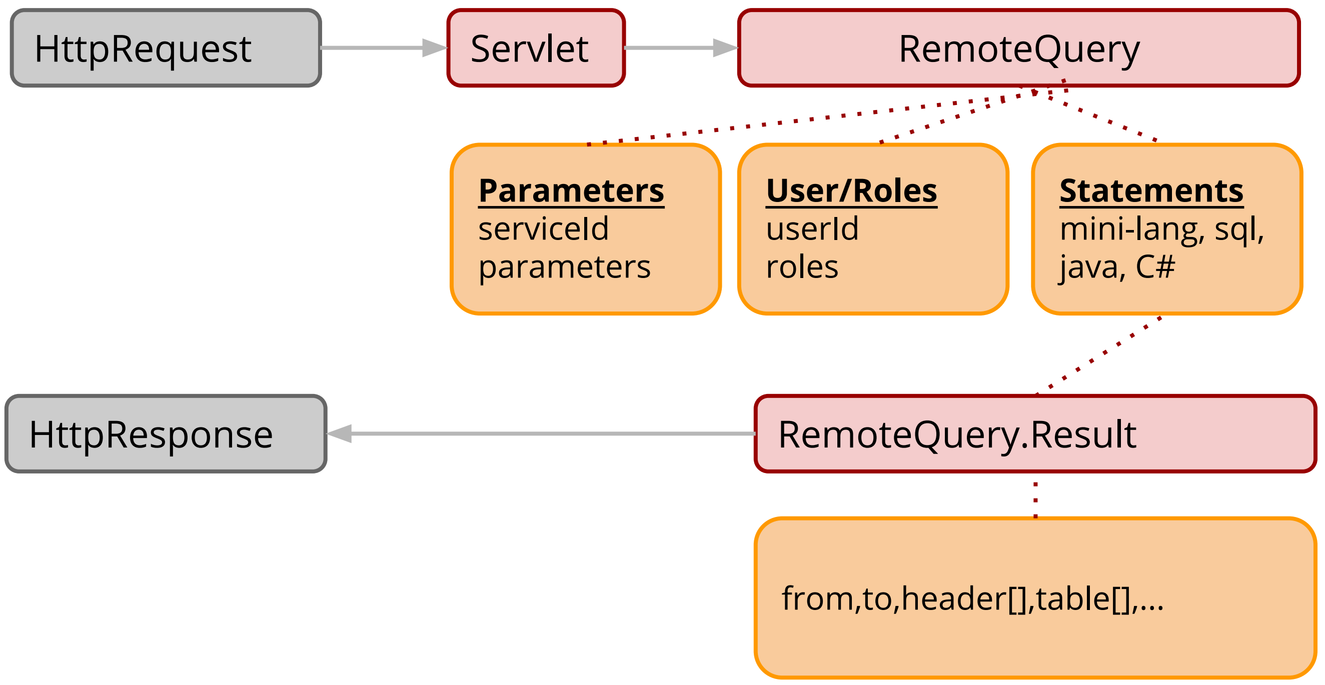
Goal of RemoteQuery

- rapid construction of services with
- best possible performance
- reduced technology for high productivity
- platforms: Java, ASP.NET/C# and SQL DBs
- core: one source file (like jQuery ;-)
- primary support for SQL and Java programs

RemoteQuery Technology Stack

- **Client**: Web (HTML5, ...), Mobile via HTTP
- **Middle** Tier: HttpHandler, **RemoteQuery** (java, csharp and PHP *)
- **Backend**: SQL scripts and programs (e.g. in Java)
- **Communication** : JSON over HTTP
- **Security** : Service access via Roles

RemoteQuery Web



Parameter Levels

Level	Description	
initial	Highest Level (0) of a parameter value. This is used for the parameter $USERID in a web application representing the authenticated user id.	select * from ADDRESS where ADDRESS_TID = bigint(:addressTid)
parameter	Level 1. Used for regular HTTP request paramters	java:address.InsertStreet
session	Level 2 In a web application this is used for String type entries in the HTTP session. Actually at start the paramters are read from the session after the request they are written back.	
application	Level 3. In a web application the entries in the ServletContext are read	select * from ADDRESS where ADDRESS_TID = bigint(:addressTid)
others 5-...		

Parameters Levels

RemoteQuery supports **parameter levels**. When injecting values into statements, values are taken from the parameters starting with the level 0 and working up to the last level. Values of a parameter level 0 have priority over a parameter of level 1. Primary usage is security such as the user principle. Parameters can be set in the statements and programmatically.

OOIT.com

Mini language for programs

sql-statement	A regular sql statements (select, update, delete, create …)	select * from ADDRESS where ADDRESS_TID = bigint(:addressTid)
java	Create a request and run the java class	java:address.InsertStreet
groovy	run a Groovy script	
sql	Runs an sql statement. Exactly like no prefix.	select * from ADDRESS where ADDRESS_TID = bigint(:addressTid)

OOIT.com

Mini language for statements (2)

include	includes the statement from another service referenced by the serviceId	include:InsertAddress;...
serviceId	runs a sub service referenced by the serviceId	serviceId:InsertAddress
java	Create a request and run the java class	java:address.InsertStreet
groovy	run a Groovy script	
add-role	add a role to the current roles	
remove-role	remove a role	
debug	write out debug level log	
tx-begint	start a transaction	
tx-end	end a transaction	

Mini language for parameters

set-X	Set a parameter value on a specific level. For X number or predefined names such as ‘initial’, ‘query’, ‘interquery’, ‘session’, ‘application’ are supported. Http request parameter are set as ‘query’ parameters.	set-initial:a=3,b=12 set-parameter:app=HR
set-if-empty-X	Set parameter only if there is no value set	set-if-emtpy:name=%;street=%
remove-X	removes a parameter	remove-param:a,b

Example: Address Search

Server Side: ServiceEntry

serviceId	searchAddress
statements	sql:select STREET, CITY ... from T_ADDRESS where STREET like :street or CITY like :city
access roles	sales, admin

Client-side Access via Ajax

Parameter	{street : 'Bahnhof%', city : 'Zuer%'}
Result	{ from: 0, to: 3, header : ['street', 'city', ...], table : [['Karl', 'Schmitt', 'Bahnhofstrasse 32', 'Zuerich'], ...] }

Features

Next to SQL scripts also Java Classes can be used

Multi-commands (separated with semicolons)

Special parameters : $USERID (principal name)

Mutli connection support

Transaction control

Appendix A: Persistence Tool

RemoteQuery.DataStore A simple persistency tool

```
DataStore<E>

    E newInstance(parameters)

    List<E> search(serviceId, parameters)

    E get(serviceId, paremeters)

    update(serviceId, E)
```