

Restlet with Prototype and JSON

Introduction

I will start this tutorial with a sample "microblog", that's a text based blog demonstrating the usage of AJAX in Restlet.

Before we step in, we should review some knowledge if you never know or forget it:

- [What's RESTful^{1,2}](#)
- [What's Restlet?³](#)
- [How to use Restlet?⁴](#)
- [How to use db4o to simplify persistence?⁵](#)
- [How to use JSON in Prototype.js?⁶](#)

Demo construction

- Web client: call background service via JSON protocol in RESTful way (GET/PUT/POST/DELETE).
- Server side: uses db4o to work as store service provider, and expose data in RESTful way.
- Server handle process: [Application⁷](#) dispatches request to [Router⁸](#), Router finds corresponding resource, [Resource⁹](#) handles request and returns representation.

DB4OSimpler.Class

It's very clean from its name that it works as db4o function simpler. Its generalOperate method handles general operation with db4o:

NOTE

Note: This class works as only non-concurrent model, because it doesn't work as [client/server model¹⁰](#). If you have requirement, you should modify it in concurrent(client/server) model by using [Db4o.openServer method¹¹](#).

```
package com.bjinfotech.util;

import com.db4o.Db4o;
import com.db4o.ObjectContainer;
import com.db4o.query.Query;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import org.apache.commons.beanutils.*;
/**
 * It's very clean from its name that it works as db4o function simpler.
 * Its generalOperate method handles general operation with db4o.
 * @author cleverpig
 *
 */
public class DB4OSimpler {
```

10. <http://developer.db4o.com/resources/api/db4o-java/com/db4o/Db4o.html>

11. [http://developer.db4o.com/resources/api/db4o-java/com/db4o/Db4o.html#openServer\(java.lang.String,%20int\)](http://developer.db4o.com/resources/api/db4o-java/com/db4o/Db4o.html#openServer(java.lang.String,%20int))

```

//operation constants that will be used as generalOperate method's param
public static final int OPERATION_SAVE=0;
public static final int OPERATION_LOAD=1;
public static final int OPERATION_UPDATE=2;
public static final int OPERATION_DELETE=3;
public static final int OPERATION_QUERY=4;
public static final int OPERATION_LIST=5;
public static final int OPERATION_CLEAR=6;
/**
 * perform general Operation
 * @param fileName
 * @param op_code corresponding integer type constant
 * @param example object which is needed in operation
 * @param keyFieldName object's key field name
 * @return
 */
public static Object generalOperate(
    String fileName,
    int op_code,
    Object example,
    String keyFieldName){
    Object ret=null;
    //open db4o file to get ObjectContainer
    ObjectContainer db=Db4o.openFile(fileName);
    Iterator iter=null;
    List list=null;
    Query query=null;
    try{
        //perform operation according to op_code param value
        switch(op_code){
            case OPERATION_SAVE:
                //just set!It's very simple!
                db.set(example);
                ret=example;
                break;
            case OPERATION_LOAD:
                //just get!
                list=db.get(example);
                if (list!=null && list.size(>0)){
                    ret=list.get(0);
                }
                break;
            case OPERATION_UPDATE:
                //at first,I find objects which will be updated with its key
                query=db.query();
                query.constrain(example.getClass());
                query.descend(keyFieldName)
                    .constrain(BeanUtils.getProperty(example, keyFieldName));
                iter=query.execute().listIterator();
                //and then delete all of them
                while(iter.hasNext()){
                    db.delete(iter.next());
                }
                //set new one,now!
                db.set(example);
                ret=example;
                break;
            case OPERATION_DELETE:
                //just like update process:find firstly,and then delete them
                query=db.query();
                query.constrain(example.getClass());
                query.descend(keyFieldName)
                    .constrain(BeanUtils.getProperty(example, keyFieldName));
                iter=query.execute().listIterator();
                if (iter.hasNext()){
                    while(iter.hasNext()){
                        db.delete(iter.next());
                    }
                }
        }
    }
}

```

field value

```

        ret=true;
    }
    else{
        ret=false;
    }
    break;
case OPERATION_QUERY:
    //just like update process:find firstly,and then return them
    query=db.query();
    query.constrain(example.getClass());
    query.descend(keyFieldName)
        .constrain(BeanUtils.getProperty(example, keyFieldName));
    iter=query.execute().listIterator();
    list=new ArrayList();
    while(iter.hasNext()){
        list.add(iter.next());
    }
    if (list.size(>0)
        ret=list;
    break;
case OPERATION_LIST:
    //return list of object which class is example.class.
    list=new ArrayList();
    iter=db.query(example.getClass()).listIterator();
    while(iter.hasNext()){
        list.add(iter.next());
    }
    if (list.size(>0)
        ret=list;
    break;
case OPERATION_CLEAR:
    //delete anything which class is example.class.
    iter=db.query(example.getClass()).listIterator();
    int deleteCount=0;
    while(iter.hasNext()){
        db.delete(iter.next());
        deleteCount++;
    };
    ret=true;
    break;
}
//commit,finally
db.commit();
}
catch(Exception ex){
    db.rollback();
    ex.printStackTrace();
}
finally{
    db.close();
}
return ret;
}
}

```

MicroblogApplication.Class

It's a restful Micoblog Server which serves static files and resource(MicroblogResource) and exposes some services(static html and Microblog):

NOTE

Note:I used TunnelService replacing custom finder,'cause I think TunnelService is easy for using.But I'd discuss how to using custom finder to implement same function.

```

package com.bjinfotech.restlet.practice.demo.microblog;

import org.restlet.Application;
import org.restlet.Component;
import org.restlet.Directory;
import org.restlet.Restlet;
import org.restlet.Router;
import org.restlet.data.Protocol;

/**
 * restful server
 * it serves static files and resource
 * @author cleverpig
 *
 */
public class MicroblogApplication {
    public static void main(String[] argv) throws Exception{
        Component component=new Component();
        //add http protocol
        component.getServers().add(Protocol.HTTP,8182);
        //add file protocol for accessing static web files in some directories
        component.getClients().add(Protocol.FILE);

        Application application=new Application(component.getContext()){
            @Override
            public Restlet createRoot(){
                //directory where static web files live
                final String DIR_ROOT_URI="file:///E:/eclipse3.1RC3/workspace/
RestletPractice/static_files/";
                //create router
                Router router=new Router(getContext());
                //attach static web files to "www" folder
                Directory dir=new Directory(getContext(),DIR_ROOT_URI);
                dir.setListingAllowed(true);
                dir.setDeeplyAccessible(true);
                dir.setNegotiateContent(true);
                router.attach("/www/",dir);
                //attach resource class:MicroblogResource to "/restful/blog" as
web service URI
                router.attach("/restful/blog",MicroblogResource.class);
                return router;
            }
        };
        //use TunnelService to simplify request's dispatching
        application.getTunnelService().setEnabled(true);
        application.getTunnelService().setMethodTunnel(true);
        application.getTunnelService().setMethodParameter("method");
        //attach application
        component.getDefaultHost().attach(application);
        component.start();
    }
}

```

microblogAppInterface.js

This is a JavaScript file used in microblog.html file,it call functions which was exposed in server side:

```

var SAVE_MODEL=1;
var UPDATE_MODEL=2;

function switchEditorModel(model){
    switch(model){
        case SAVE_MODEL:
            Element.show('save_button');
            Element.hide('update_button');
            Element.hide('remove_button');

```

```

        Element.hide('new_button');
        break;
    case UPDATE_MODEL:
        Element.hide('save_button');
        Element.show('update_button');
        Element.show('remove_button');
        Element.show('new_button');
        break;
    }
}

...

Event.observe(
    'save_button',
    'click',
    function(){
        var formObj=Form.serialize('edit_form',true);
        var xmlhttp = new Ajax.Request(
            "/restful/blog?method=PUT",
            {
                method: 'post',
                parameters: 'json='+encodeURIComponent(Object.toJSON(formObj)),
                onComplete: function(transport){
                    var retObj=transport.responseText.evalJSON();
                    if (retObj.subject){
                        alert('ok,"'+retObj.subject+" was saved!');
                        refreshBloglist();
                        switchEditorModel(UPDATE_MODEL);
                    }
                }
            }
        );
    },
    false
);

...

function refreshBloglist(){
    var xmlhttp = new Ajax.Request(
        "/restful/blog",
        {
            method: 'get',
            parameters: '',
            onComplete: function(transport){
                var retObjs=transport.responseText.evalJSON();
                if (retObjs.length && retObjs.length>0){
                    var listRepr='<ul>\n';
                    retObjs.each(function(obj,index){
                        if (index<retobjs.length-1 ) listrepr+="
<li>"><a href="javascript:load('\'+obj.subject+'\');">'+obj.subject+'</a>\n';
                    });
                    listRepr+="<ul>\n";
                    $('blogList').innerHTML=listRepr;
                }
                else{
                    $('blogList').innerHTML='Here is empty';
                }
            }
        }
    );
}

function load(subject){
    var xmlhttp = new Ajax.Request(
        "/restful/blog",
        {
            method: 'get',
            parameters: 'subject='+encodeURIComponent(subject),

```

```

        onComplete: function(transport){
            var retObj=transport.responseText.evalJSON();
            if (retObj.subject){
                $('subject').value=retObj.subject;
                $('content').value=retObj.content;
                $('tags').value=retObj.tags;
                alert('ok,'" +retObj.subject+" was loaded!');
                switchEditorModel(UPDATE_MODEL);
            }
        }
    );
}
}

```

MicroblogResource.Class

```

package com.bjinfotech.restlet.practice.demo.microblog;

import java.util.List;
import java.util.logging.Logger;

import org.restlet.Context;
import org.restlet.data.CharacterSet;
import org.restlet.data.Form;
import org.restlet.data.Language;
import org.restlet.data.MediaType;
import org.restlet.data.Request;
import org.restlet.data.Response;
import org.restlet.resource.Representation;
import org.restlet.resource.Resource;
import org.restlet.resource.ResourceException;
import org.restlet.resource.StringRepresentation;
import org.restlet.resource.Variant;
import org.restlet.data.Status;
import com.bjinfotech.util.JSONSimpler;
import com.bjinfotech.util.Utils;

public class MicroblogResource extends Resource {
    Logger log=Logger.getLogger(MicroblogResource.class.getSimpleName());
    //MicroblogPersistenceManager
    MicroblogPersistenceManager micoblogPM=new MicroblogPersistenceManager();
    //StringRepresentation constant
    final StringRepresentation NO_FOUND_REPR=new StringRepresentation("No
found!");
    final StringRepresentation ERR_REPR=new StringRepresentation("something
wrong!");
    //json param name in request
    final String JSON_PARAM="json";

    public MicroblogResource(
        Context context,
        Request request,
        Response response) {
        super(context, request, response);
        this.getVariants().add(new Variant(MediaType.TEXT_PLAIN));
        //it's important,please don't forget it.
        this.setAvailable(true);
        this.setModifiable(true);
        this.setNegotiateContent(true);
    }
    @Override
    /**
     * representing after calling default get handle
     * @param variant
     */
    public Representation represent(Variant variant) throws ResourceException{

```

```

        log.info("representing after calling default get handle...");
        Representation result = null;
        if (variant.getMediaType().equals(MediaType.TEXT_PLAIN)) {
            //find "subject" param in request,"subject" param is tranformed from
            web client.it means load one blog with special subject
            String
            subject=getRequest().getResourceRef().getQueryAsForm().getFirstValue("subject");
            log.info("subject:"+subject);
            //handle query
            if (subject!=null && subject.length()>0){
                //find blog with special subject
                Microblog example=new Microblog();
                example.setSubject(subject);
                List queryRet=micoblogPM.query(example);
                //return result in JSON format StringRepresentation
                if (queryRet!=null && queryRet.size()>0){
                    result=new StringRepresentation(
                        JSONSimpler.serializeFromBean(queryRet.get(0)),
                        MediaType.APPLICATION_JSON,
                        Language.ALL,
                        CharacterSet.UTF_8
                    );
                }
                else{
                    result=NO_FOUND_REPR;
                }
            }
            else{
                //return blog list in JSON format StringRepresentation
                result=new StringRepresentation(
                    JSONSimpler.serializeFromBeanList(micoblogPM.list()),
                    MediaType.APPLICATION_JSON,
                    Language.ALL,
                    CharacterSet.UTF_8
                );
            }
        }
        return result;
    }
    /**
     * call MicroblogPersistenceManager's method excluding query and list,just
     save/update/delete.
     * @param method method name
     * @param jsonParamVal json param value coming from request
     * @return
     */
    protected StringRepresentation callMethod(String method,String jsonParamVal){
        //transform json format string to Microblog object
        Microblog
        microblog=(Microblog)JSONSimpler.deserializeToBean(jsonParamVal,Microblog.class);
        if (microblog!=null){
            //call method and gain json format string as responseText
            String responseText=Utils.callMethodAndGainResponseJSONStr(
                micoblogPM,
                method,
                jsonParamVal,
                Microblog.class);
            log.info("response:"+responseText);
            //return json StringRepresentation
            return new StringRepresentation(
                responseText,
                MediaType.APPLICATION_JSON,
                Language.ALL,
                CharacterSet.UTF_8
            );
        }
        else{
            return ERR_REPR;
        }
    }

```

```

    }
    @Override
    /**
     * handling post in high level
     * @param entity
     */
    public void acceptRepresentation(Representation entity) throws
ResourceException{
        log.info("handling post in high level...");
        super.acceptRepresentation(entity);
        getResponse().setStatus(Status.SUCCESS_OK);
        Form f = new Form(entity);
        String jsonParamVal=f.getValues(JSON_PARAM);
        log.info("json param:"+jsonParamVal);
        //call update and set response
        getResponse().setEntity(callMethod("update", jsonParamVal));
    }
    @Override
    /**
     * handling put in high level
     * @param entity
     */
    public void storeRepresentation(Representation entity) throws
ResourceException{
        log.info("handling put in high level...");
        super.storeRepresentation(entity);
        getResponse().setStatus(Status.SUCCESS_CREATED);
        Form f = new Form(entity);
        String jsonParamVal=f.getValues(JSON_PARAM);
        log.info("json param:"+jsonParamVal);
        //call save and set response
        getResponse().setEntity(callMethod("save", jsonParamVal));
    }
    @Override
    /**
     * handling delete in high level
     * @param entity
     */
    public void removeRepresentations() throws ResourceException{
        log.info("handling delete in high level...");
        super.removeRepresentations();
        getResponse().setStatus(Status.SUCCESS_OK);
        Form f = getRequest().getEntityAsForm();
        String jsonParamVal=f.getValues(JSON_PARAM);
        log.info("json param:"+jsonParamVal);
        //call delete and set response
        getResponse().setEntity(callMethod("delete", jsonParamVal));
    }
}
}

```

Microblog.Class

```
package com.bjinfotech.restlet.practice.demo.microblog;
```

```

public class Microblog {
    private String subject;
    private String content;
    private String tags;

    public String getSubject() {
        return subject;
    }
    ...
}

```

}

Running Application

Running MicroblogApplication, and visit <http://localhost:8182/www/microblog.html>.



Checkout Full Code

[microblog_sourcecode](#)¹²

How to custom Finder to replace TunnelService

Sure, you can custom a finder to do what tunnelService do.

You can visit <http://dobrzanski.net/2007/04/22/using-put-and-delete-methods-in-ajax-requesta-with-prototypejs/> to get detail about how to using restful way in prototype.

NOTE

12. daisy:54-restlet (microblog_sourcecode)

A simpler way to do this is to customize the TunnelService.

`getApplication().getTunnelService().setMethodName("_method")`. That's all!

In Application:

```
...
Router router = new Router(getContext());
//It's very easy!
router.setFinderClass(PrototypeFinder.class);
...
```

Custom Finder:

```
public class PrototypeFinder extends Finder {
    public PrototypeFinder(Context context, Class
targetClass) {
        super(context, targetClass);
    }

    public void handle(Request request, Response response) {
        //get "_method" param value
        Parameter p = request.getEntityAsForm().getFirst("_method");
        //reset request method according to "_method" param value
        request.setMethod(null != p ? Method.valueOf(p.getValue()) :
request.getMethod());
        super.handle(request, response);
    }
}
```

javascript snippet in web page:

```
...

function callJSON() {
    new Ajax.Request('/ajax', {
        parameters: 'name=PUT', method: 'put', putBody: "PUT BODY",
        onComplete: function (transport) {
            alert(transport.responseText);
        }
    });
    new Ajax.Request('/ajax', {
        parameters: 'name=POST', method: 'post',
        onComplete: function (transport) {
            alert(transport.responseText);
        }
    });
    new Ajax.Request('/ajax', {
        parameters: 'name=DELETE', method: 'delete',
        onComplete: function (transport) {
            alert(transport.responseText);
        }
    });
}

...
```

Thanks

Evgeny Shepelyuk:this guy gave me a lot of good advice!

Links

Router

<http://www.restlet.org/documentation/1.1/api/org/restlet/Router.html>

Application

<http://www.restlet.org/documentation/1.1/api/org/restlet/Application.html>

Resource

<http://www.restlet.org/documentation/1.1/api/org/restlet/resource/Resource.html>

db4o

<http://www.db4o.com>