Vesys Pascal Lüscher

Inhalt

[Socket 2](#_Toc455761570)

[Client 2](#_Toc455761571)

[Server 3](#_Toc455761572)

[Soap 4](#_Toc455761573)

[Client 4](#_Toc455761574)

[Server 5](#_Toc455761575)

[REST 6](#_Toc455761576)

[Date 6](#_Toc455761577)

[Resource 7](#_Toc455761578)

[Server 10](#_Toc455761579)

[Client 11](#_Toc455761580)

# Socket

## Client

**import** java.io.BufferedReader;

**import** java.io.InputStreamReader;

**import** java.io.PrintWriter;

**import** java.net.Socket;

**public** **class** EchoClient {

**public** **static** **void** main(String[] args) **throws** Exception {

String host = "localhost";

**int** port = 1234;

**if** (args.length > 0) { host = args[0]; }

**if** (args.length > 1) { port = Integer.*parseInt*(args[1]); }

System.*out*.println("connecting to " + host + ":" + port);

Socket s = **new** Socket(host, port, **null**, 0);

// null means any network interface

// 0 means any port

System.*out*.println("connected to " + s.getRemoteSocketAddress());

PrintWriter out = **new** PrintWriter(s.getOutputStream());

BufferedReader in = **new** BufferedReader(**new** InputStreamReader(s.getInputStream()));

BufferedReader stdin = **new** BufferedReader(**new** InputStreamReader(System.*in*));

String input = stdin.readLine();

**while** (input != **null** && !input.equals("")) {

out.println(input);

out.flush();

System.*out*.println("Echo: " + in.readLine());

input = stdin.readLine();

}

s.close();

System.*out*.println("disconnected.");

}

}

## Server

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.PrintWriter;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**public** **class** EchoServer2 {

**public** **static** **void** main(String args[]) **throws** IOException {

**int** port = 1234;

**try** (ServerSocket server = **new** ServerSocket(port)) {

System.*out*.println("Startet Echo Server on port " + port);

**while** (**true**) {

Socket s = server.accept();

Thread t = **new** Thread(**new** EchoHandler(s));

t.start();

}

}

}

**private** **static** **class** EchoHandler **implements** Runnable {

**private** **final** Socket s;

**private** EchoHandler(Socket s) {

**this**.s = s;

}

**public** **void** run() {

System.*out*.println("connection from " + s);

**try** {

BufferedReader in = **new** BufferedReader(**new** InputStreamReader(s

.getInputStream()));

PrintWriter out = **new** PrintWriter(s.getOutputStream(), **true**);

String input = in.readLine();

**while** (input != **null** && !"".equals(input)) {

out.println(input);

input = in.readLine();

}

System.*out*.println("done serving " + s);

} **catch** (IOException e) {

System.*err*.println(e);

**throw** **new** RuntimeException(e);

} **finally** {

**try** {

s.close();

} **catch** (IOException e) {

System.*err*.println(e);

}

}

}

}

}

# Soap

## Client

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.xml.ws.handler.Handler;

**import** javax.xml.ws.handler.HandlerResolver;

**import** javax.xml.ws.handler.PortInfo;

**import** ch.fhnw.ds.echo.client.jaxws.Echo;

**import** ch.fhnw.ds.echo.client.jaxws.EchoService;

**import** ch.fhnw.ds.echo.handlers.LogHandler;

**import** ch.fhnw.ds.echo.handlers.TestLogicalHandler;

**import** ch.fhnw.ds.echo.handlers.TestSoapHandler;

**public** **class** Client {

**public** **static** **void** main(String[] args) **throws** Exception {

EchoService service = **new** EchoService();

service.setHandlerResolver(**new** ClientHandlerResolver());

Echo port = service.getEchoPort();

String result = port.sayHello("Dominik");

System.*out*.println(result);

}

}

**class** ClientHandlerResolver **implements** HandlerResolver {

@SuppressWarnings("rawtypes")

**public** List<Handler> getHandlerChain(PortInfo portInfo) {

List<Handler> chain = **new** ArrayList<Handler>();

chain.add(**new** LogHandler()); // SOAP Handler

chain.add(**new** TestLogicalHandler()); // Logical Handler

chain.add(**new** TestSoapHandler()); // SOAP Handler

**return** chain;

}

}

## Server

**import** java.util.Date;

**import** javax.jws.HandlerChain;

**import** javax.jws.WebParam;

**import** javax.jws.WebService;

@WebService

**public** **class** Echo {

**public** String sayHello(@WebParam(name = "name") String name) {

**if** ("ex".equals(name)) **throw** **new** IllegalStateException("ex not expected");

**return** "Hello " + name + " from SOAP at " + **new** Date();

}

}

**import** javax.xml.ws.Endpoint;

**public** **class** EchoPublisher {

**public** **static** **void** main(String[] args){

String url = "http://127.0.0.1:9898/echo";

Endpoint.*publish*(url, **new** Echo());

System.*out*.println("service published");

System.*out*.println("WSDL available at "+url+"?wsdl");

}

}

# REST

## Date

**import** java.util.Date;

**import** javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement

**public** **class** Msg {

**private** String text;

**private** Date date;

**public** Msg(){}

**public** Msg(String text){

**this**(text, **new** Date());

}

**public** Msg(String text, Date date) {

**this**.text = text;

**this**.date = date;

}

**public** String getText() {

**return** text;

}

**public** **void** setText(String text) {

**this**.text = text;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

}

## Resource

**import** java.beans.XMLEncoder;

**import** java.io.ByteArrayOutputStream;

**import** java.net.URI;

**import** javax.inject.Singleton;

**import** javax.ws.rs.Consumes;

**import** javax.ws.rs.DELETE;

**import** javax.ws.rs.FormParam;

**import** javax.ws.rs.GET;

**import** javax.ws.rs.POST;

**import** javax.ws.rs.PUT;

**import** javax.ws.rs.Path;

**import** javax.ws.rs.PathParam;

**import** javax.ws.rs.Produces;

**import** javax.ws.rs.core.CacheControl;

**import** javax.ws.rs.core.Context;

**import** javax.ws.rs.core.HttpHeaders;

**import** javax.ws.rs.core.MultivaluedMap;

**import** javax.ws.rs.core.Request;

**import** javax.ws.rs.core.Response;

**import** javax.ws.rs.core.Response.ResponseBuilder;

**import** javax.ws.rs.core.UriInfo;

**import** ch.fhnw.ds.rest.msg.data.Msg;

@Singleton

@Path("/msg")

**public** **class** MsgResource {

**private** String msg = "Hello, world!";

**public** MsgResource() {

System.*out*.println("MsgResource() called");

}

// GET on /msg

// ===========

@GET

@Produces("text/plain")

**public** String getPlain() {

**return** msg + "\n";

}

@GET

@Produces("text/html")

**public** String getHtml() {

StringBuffer buf = **new** StringBuffer();

buf.append("<html><body><h1>Message Text</h1>" + msg + "<br>");

buf.append("<form method=\"POST\" action=\"/msg\">");

buf.append("<p>Text: <input name=\"msg\" type=\"text\" size=20/>");

buf.append("<input type=\"submit\" value=\"Submit\" />");

buf.append("</form>");

buf.append("</body></html>");

**return** buf.toString();

}

@GET

@Produces("text/xml")

**public** String getSimpleXml() {

**return** "<string>" + msg + "</string>";

}

@GET

@Produces( { "application/json", "application/xml" })

**public** Msg getXml() {

**return** **new** Msg(msg);

}

// PUT on /msg

// ===========

@PUT

@Consumes("text/plain")

**public** **void** setTextPlain(String new\_msg) {

msg = new\_msg;

}

@PUT

@Consumes( { "application/json", "application/xml" })

**public** **void** setTextXml(Msg message) {

msg = message.getText();

}

@PUT

@Consumes("application/x-www-form-urlencoded")

@Produces("text/xml")

**public** String setTextForm(@FormParam("msg") String new\_msg) {

msg = new\_msg;

ByteArrayOutputStream stream = **new** ByteArrayOutputStream();

XMLEncoder enc = **new** XMLEncoder(stream);

enc.writeObject(new\_msg);

enc.close();

**return** **new** String(stream.toByteArray()) + "\n";

}

// POST on /msg (used for forms)

// =============================

@POST

@Consumes("application/x-www-form-urlencoded")

@Produces("text/html")

**public** String doPost(@FormParam("msg") String new\_msg) {

msg = new\_msg;

**return** getHtml();

}

@POST

@Consumes( { "application/xml", "application/json" })

**public** Response createNewMessage(@Context UriInfo uriInfo, Msg message) {

URI location = uriInfo.getAbsolutePathBuilder().path(message.getText())

.build();

**return** Response.*created*(location).build();

}

// DELETE on /msg

// ==============

@DELETE

@Produces("text/plain")

**public** String onDelete() {

msg = **null**;

**return** "Message deleted.\n";

}

// GET on /msg/cc

// ==============

@GET

@Path("cc")

@Produces("text/plain")

**public** Response getPlain2() {

ResponseBuilder builder = Response.*ok*(msg + "\n");

CacheControl cc = **new** CacheControl();

cc.setMaxAge(1000); // HTTP max-age field, in seconds

cc.setNoTransform(**true**);

cc.setPrivate(**true**);

builder.cacheControl(cc);

**return** builder.build();

}

// GET on /msg/{id} and on /msg/{id}/headers

// =========================================

@GET

@Produces("text/plain")

@Path("{id}")

**public** String readDetailsInfo(@PathParam("id") String path, @Context Request r) {

**return** msg + ": " + path + "\n";

}

@GET

@Produces("text/plain")

@Path("{id}/headers")

**public** String readDetailHeaders(@PathParam("id") String path,

@Context HttpHeaders headers) {

StringBuffer buf = **new** StringBuffer();

buf.append("Headers of request " + path + "\n\n");

MultivaluedMap<String, String> map = headers.getRequestHeaders();

**for** (String key : map.keySet()) {

buf.append(key + ": " + map.getFirst(key) + "\n");

}

**return** buf.toString();

}

}

## Server

**import** java.io.IOException;

**import** java.net.URI;

**import** java.util.HashMap;

**import** java.util.Map;

**import** org.glassfish.grizzly.http.server.HttpServer;

**import** org.glassfish.jersey.grizzly2.httpserver.GrizzlyHttpServerFactory;

**import** org.glassfish.jersey.server.ResourceConfig;

**public** **class** Server2 {

**public** **static** **void** main(String[] args) **throws** IOException {

**final** String baseUri = "http://localhost:9998/";

**final** Map<String, Object> config = **new** HashMap<>();

config.put("jersey.config.server.provider.packages", "ch.fhnw.ds.rest.msg.resources"); // package with resource classes

// @Singleton annotations will be respected

**final** ResourceConfig rc = **new** ResourceConfig().setProperties(config);

System.*out*.println("Starting grizzly...");

HttpServer httpServer = GrizzlyHttpServerFactory.*createHttpServer*(URI.*create*(baseUri), rc);

System.*out*.println(String.*format*("Jersey app started with WADL available at "

+ "%sapplication.wadl\nTry out %smsg\nHit enter to stop it...",

baseUri, baseUri));

System.*in*.read();

httpServer.shutdown();

}

}

## Client

**import** javax.ws.rs.client.Client;

**import** javax.ws.rs.client.ClientBuilder;

**import** javax.ws.rs.client.Entity;

**import** javax.ws.rs.client.WebTarget;

**import** javax.ws.rs.core.Form;

**import** javax.ws.rs.core.MediaType;

**public** **class** Client3 {

**public** **static** **void** main(String[] args) {

Client c = ClientBuilder.*newClient*();

WebTarget r = c.target("http://localhost:9998/msg");

Form f = **new** Form();

f.param("msg", "Hello from Client3");

String result = r.request().accept(MediaType.*TEXT\_HTML*).post(Entity.*form*(f)).toString();

System.*out*.println(result);

}

}