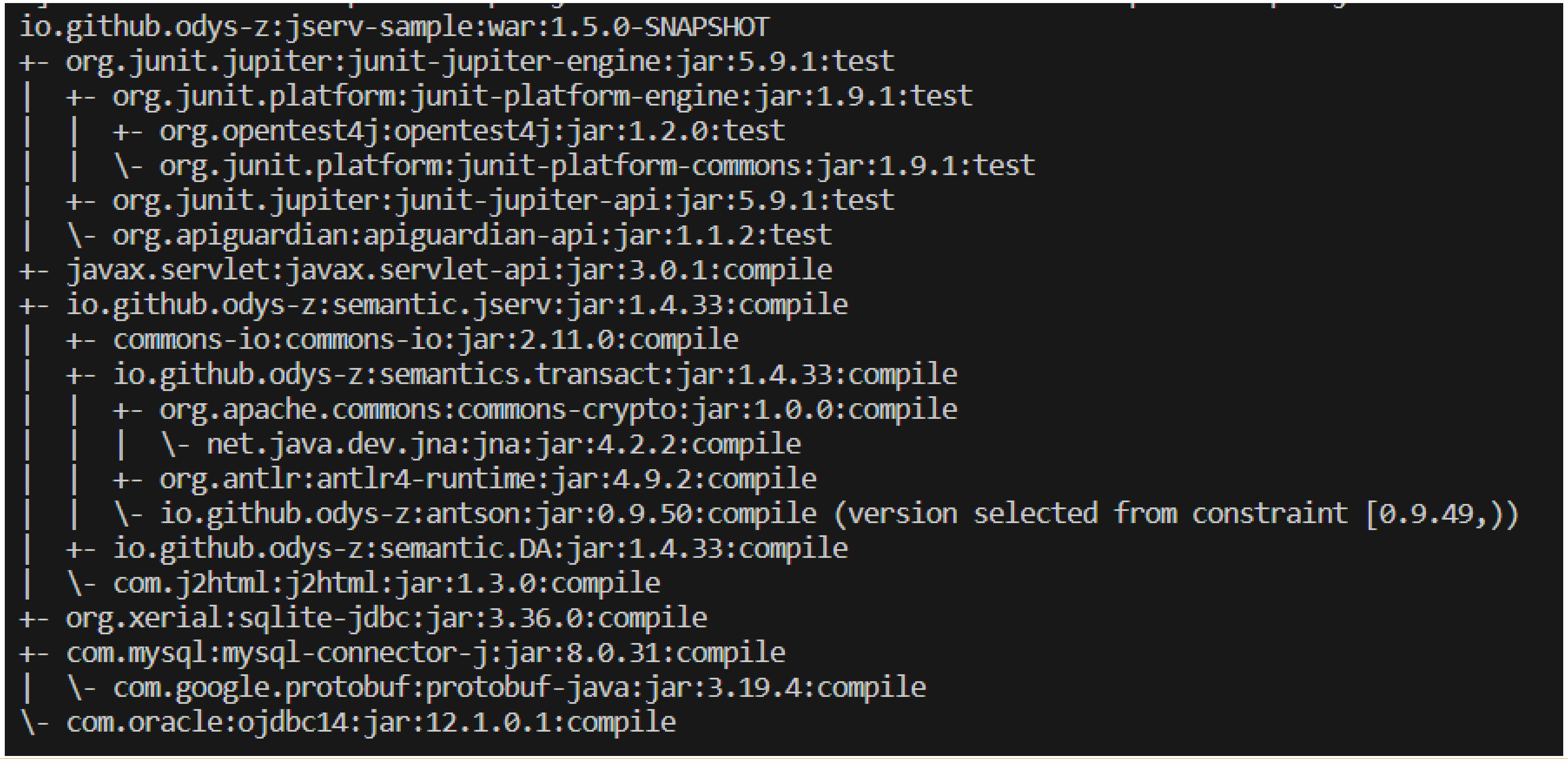
Semantic-* Verified App Architecture

SOA & OOP BEST PRACTICE

https://github.com/odys-z

CRUD Peer to Peer

Jserv-sample$ mvn clean dependency:tree



```
io.github.odys-z:jserv-sample:war:1.5.0-SNAPSHOT
+- org.junit.jupiter:junit-jupiter-engine:jar:5.9.1:test
|  +- org.junit.platform:junit-platform-engine:jar:1.9.1:test
|  |  +- org.opentest4j:opentest4j:jar:1.2.0:test
|  |  \- org.junit.platform:junit-platform-commons:jar:1.9.1:test
|  +- org.junit.jupiter:junit-jupiter-api:jar:5.9.1:test
|  \- org.apiguardian:apiguardian-api:jar:1.1.2:test
+- javax.servlet:javax.servlet-api:jar:3.0.1:compile
+- io.github.odys-z:semantic.jserv:jar:1.4.33:compile
|  +- commons-io:commons-io:jar:2.11.0:compile
|  +- io.github.odys-z:semantics.transact:jar:1.4.33:compile
|  |  +- org.apache.commons:commons-crypto:jar:1.0.0:compile
|  |  |  \- net.java.dev.jna:jna:jar:4.2.2:compile
|  |  +- org.antlr:antlr4-runtime:jar:4.9.2:compile
|  |  \- io.github.odys-z:antson:jar:0.9.50:compile (version selected from constraint [0.9.49,))
|  +- io.github.odys-z:semantic.DA:jar:1.4.33:compile
|  \- com.j2html:j2html:jar:1.3.0:compile
+- org.xerial:sqlite-jdbc:jar:3.36.0:compile
+- com.mysql:mysql-connector-j:jar:8.0.31:compile
|  \- com.google.protobuf:protobuf-java:jar:3.19.4:compile
\- com.oracle:ojdbc14:jar:12.1.0.1:compile
```

CRUD: Userst Peer to Peer

Typescript + ReactJS (Client)

Semantic Protocol

```
export class UserstReq extends UserReq {
   static __type__ = 'io.oz.jsample.semantier.UserstReq';
   static A = {... }
   pk: any;
   userId: string;
   userName: string;
   orgId: string;
   roleId: string;
   hasTodos: boolean;
   record: Tierec;
   relations: DbRelations;
   deletings: string[];
   page: PageInf;
}
```

Java + Servlet (JSON Service)

```
io.oz.jsample.samantier.UserstReq implements Anson{
		PageInf page;
		String userId;
		String userName;
		String orgId;
		String roleId;
		boolean hasTodos;

		public HashMap<String, Object> record;
		public ArrayList<Relations> relations;
		String pk;
}
```

```
Protocol.registerBody(UserstReq.__type__,  (jsonBd) => { return new UserstReq(uri, jsonBd); });
```

```
@WebServlet(description = "Semantic tier: users", urlPatterns = { "/users.tier" })
public class UsersTier extends ServPort<UserstReq> {
    @Override
    protected void onPost(AnsonMsg<UserstReq> jmsg, HttpServletResponse resp)
            throws ServletException, IOException, AnsonException, SemanticException {
        resp.setCharacterEncoding("UTF-8");
        try {
            IUser usr = JSingleton.getSessionVerifier().verify(jmsg.header());
            UserstReq jreq = jmsg.body(0);

            AnsonResp rsp = null;
            if (A.records.equals(jreq.a()))
                rsp = records(jreq, usr);
            …
            else throw new SemanticException(String.format(
                "request.body.a can not handled: %s\\n Only a = [%s, %s, %s, %s, %s] are supported.",
                jreq.a(), A.records, A.rec, A.insert, A.update, A.del));
            write(resp, ok(rsp));
        } catch (Exception e) { write(resp, err(MsgCode.exSession, e.getMessage()));
        } finally { resp.flushBuffer(); }
    }
```

```
export class UsersTier extends Semantier {
  port = 'userstier';

  records(conds: PageInf, onLoad: OnLoadOk<Tierec>) {
    let client = this.client;
    let that = this;

    let req = client.userReq(this.uri, this.port,
                             new UserstReq( this.uri, conds )
        .A(UserstReq.A.records) );

    client.commit(req, (resp) => {
        let {cols, rows} = AnsonResp.rs2arr(resp.Body().Rs());
        that.rows = rows;
        conds.total = resp.Body()?.Rs()?.total || 0;
        onLoad(cols, rows as Tierec[]);
    },
    this.errCtx);
}
```

```
public class AnsonMsg <T extends AnsonBody>
extends Anson {
  public static enum Port implements IPort {
    ...
    userstier("users.tier"),
  }
}
```

Typscript App & React Context

Export App

```
render() {
  let that = this;
  return (
    <MuiThemeProvider theme={JsampleTheme}>
      <AnContext.Provider value={{
          ssInf: undefined,
          pageOrigin: window ? window.origin : 'localhost',
          servId: this.state.servId,
          servs: this.props.servs,
          anClient: this.anClient,
          uiHelper: this.anReact,
          hasError: this.state.hasError,
          iparent: this.props.iparent,
          ihome: this.props.iportal || 'portal.html',
          error: this.errorCtx,
      }} > …
```

```
render() {
  let that = this;
  return (
    <MuiThemeProvider theme={JsampleTheme}>
      <AnContext.Provider value={{
          …
       }} >
        <Sys menu='sys.menu.jsample'
          sys={L('AnReact')} menuTitle={L('Sys Menu')}
          myInfo={myInfoPanels}
          onLogout={this.goPortal} />
        {this.errorMsgbox}
      </AnContext.Provider>
    </MuiThemeProvider>);
```

Typscript App & React Context

Export App

```
render() {
 return (
  <MuiThemeProvider theme={JsampleTheme}>
    <AnContext.Provider value={{
      ssInf: undefined,
      pageOrigin: window ? window.origin : 'localhost',
      servId: this.state.servId,
      servs: this.props.servs,
      anClient: this.anClient,
      uiHelper: this.anReact,
      hasError: this.state.hasError,
      iparent: this.props.iparent,
      ihome: this.props.iportal || 'portal.html',
      error: this.errorCtx,
    }} > …
```

```
render() {
 let that = this;
 return (
  <MuiThemeProvider theme={JsampleTheme}>
    <AnContext.Provider value={{
      …
     }} >
      <Sys menu='sys.menu.jsample'
        sys={L('AnReact')} menuTitle={L('Sys Menu')}
        onLogout={this.goPortal} />
      {this.errorMsgbox}
    </AnContext.Provider>
  </MuiThemeProvider>);
```

Typscript App & React Context

Exception handling

```
this.errorCtx = {onError: this.onError, msg: ''};

onError(c: string, r: AnsonMsg<AnsonResp>) {
    this.errorCtx.msg = r.Body()?.msg();
    this.errorMsgbox = <AnError
            onClose={() => this.onErrorClose(c)} fullScreen={false}
            title={L('Error’)}
            msg={this.errorCtx.msg as string} />

    this.setState({
        hasError: !!c,
        nextAction: c === MsgCode.exSession ? 're-login' : 'ignore'});
}
```

```
export class UsersTier extends Semantier {
    port = 'userstier’;

    records(conds: PageInf, onLoad:
        OnLoadOk<Tierec>) {
            …
            client.commit(req, (resp) => {
                …
            },
        this.errCtx);
}
```

Typscript App & React Context

Exception handling

```
this.errorCtx = {onError: this.onError, msg: ''};

onError(c: string, r: AnsonMsg<AnsonResp>) {
   this.errorCtx.msg = r.Body()?.msg();
   this.errorMsgbox = <AnError
         onClose={() => this.onErrorClose(c)} fullScreen={false}
         title={L('Error')}
         msg={this.errorCtx.msg as string} />

   this.setState({
      hasError: !!c,
      nextAction: c === MsgCode.exSession
                                 ? 're-login' : 'ignore'});
}
```

```
export class UsersTier extends Semantier {
   port = 'userstier';

   records(conds: PageInf, onLoad: OnLoadOk<Tierec>) {
         …
         client.commit(req, (resp) => {
            …
         },
      this.errCtx);
}

// @anclient/semantier
export interface ErrorCtx {
   msg?: string;
   onError: (code: string,
      resp: AnsonMsg<AnsonResp>) => void
}
```

Typscript App & React Context

```
static bindHtml(elem: string, opts: AnreactAppOptions) : void {
    let portal = opts.portal || 'index.html';
    try { Langstrs.load('/res-vol/lang.json'); } catch (e) {}
    AnReactExt.bindDom(elem, opts, onJsonServ);

    function onJsonServ(elem: string,
                opts: AnreactAppOptions, json: JsonServs) {
        let dom = document.getElementById(elem);
        ReactDOM.render(
            <App servs={json}
                    servId={opts.serv}
                    iportal={portal}
                    iwindow={window}/>,
          dom);
    }
}
```

```
static bindDom( elem: string, opts: AnreactAppOptions,
        onJsonServ: (elem: string,
            opts: AnreactAppOptions, json: JsonServs) => void) {

    if (!opts.serv) opts.serv = 'host';
    if (!opts.home) opts.home = 'main.html';

    if (typeof elem === 'string') {
        $.ajax({ url: 'private/host.json’ })
        .done( (json: JsonServs) => onJsonServ(elem, opts, json) )
        .fail( (e: any) => {
            $.ajax({url: 'github.json'})
            .done((json: JsonServs) => onJsonServ(elem, opts,
json))
            .fail( (e: { responseText: any; }) => { … ) } )
        } )
    }
}
```

Docker & Volume

No Nginx Reverse Proxy

```
docker build –t .
docker run --name jsample -v jsample.sqlite:/var/local/volume -p 8080:8080 -d --rm ###
```

Web Service

```
FROM nginx:stable-alpine
COPY dist/ /usr/share/nginx/html/
```

Jserv Sample

```
FROM tomcat:9.0
COPY target/jserv-sample.war $CATALINA_HOME/webapps
EXPOSE 8080
```

HTTP/HTTPS Stream

HTTP/HTTPS Response code 206

```
protected void doGet(HttpServletRequest req HttpServletResponse resp)
        throws ServletException, IOException {
    String range = req.getHeader("Range");
    if (!isblank(range)) {
        try {
            Docs206.get206(req, resp);
        } catch (SsException e) {
            write(resp, err(MsgCode.exSession, e.getMessage()));
            resp.sendError(HttpServletResponse.SC_UNAUTHORIZED);
        }
        return;
    }
    …
}
```

```
mime === 'video':
return (
<video key={i} controls
  onLoad={(e) => {
    console.log('--- video loaded ---’ }}
  onTouchStart={(e) => {
    // pause / unpause
    state.paused = !state.paused;
  }}
  onEnded={e => config.paused = true}
>
  <source src={src} type={mime}/>
</video>);
```

Problem

Application is Fragile

- Cons
- Pros

IDE Tool:

- Manage dataset
- (De)serialize Packages
- CRUD module