Z důvodu studia build.xml se autor snaží build.xml redukovat na minimum, tak aby projekt (s redukovaným build.xml) byl stále spustitelný. Redukce build.xml probíhá tím způsobem, že se daný soubor po řádcích opakovaně promazává. Script promazávající build.xml je napsán v jazyce Python. Test je prováděn na jednoduchém projektu generovaném pomocí IDE NetBeans. Test je prováděn na GlassFish UBuntu.

Redukce build.xml

Deploynutí projektu + Dokumentace testovacího scriptu v jazyce Python

Jonáš Bartoň

Redukce build.xml

1. Testovaný project.

1.1. Zdrojový kód projektu.

Jedná se tento jednoduchý zdrojový kód:

Zdrojový kód je uložen na adrese:

https://drive.google.com/drive/folders/19AKfuhjugehd9dM6ZLz3uhksmuKKzNAB

1.2. Spuštění projektu na Glassfish.

Pro spuštění projektu je potřeba jej deployout. Před deploynutím je potřeba provést následující kroky

1) Build pomocí Ant
2) Vytvoření archivu .war
3) Deploy na GlassFish

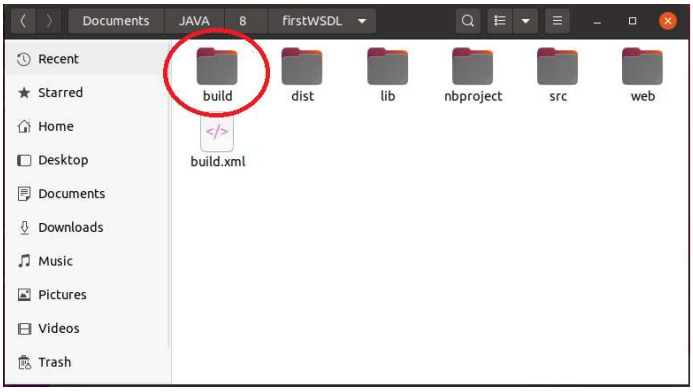
1.3. Build pomocí Ant.

Build pomocí ant spustíme z terminalu pomocí příkazu:

ant -f build.xml

```
root@jonas-VirtualBox:/home/jonas/Documents/JAVA/8/firstWSDL# ant -f build.xml
Buildfile: /home/jonas/Documents/JAVA/8/firstWSDL/build.xml

BUILD SUCCESSFUL
Total time: 1 second
```

Do složky s projektem nám přibyla složka build



1.4. Vytvoření archivu .war.

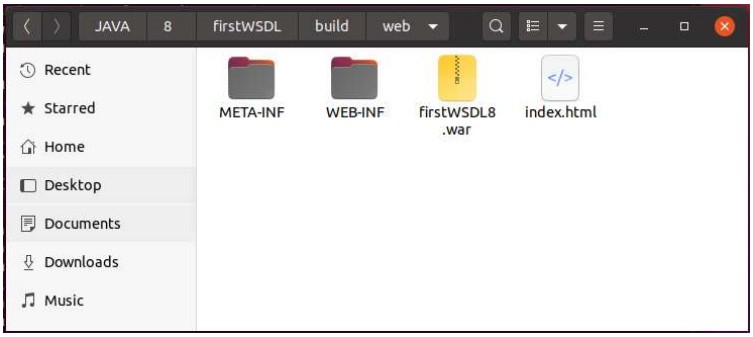
Archiv .war vytvoříme ve složce ../build/web/

Zapíšeme příkaz:

jar -cvf firstWSDL8.war .

```
root@jonas-VirtualBox:/home/jonas/Documents/JAVA/8/firstWSDL/build/web# jar -cvf firstWSDL8.war  .
added manifest
adding: WEB-INF/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/web.xml(in = 408) (out= 206)(deflated 49%)
adding: WEB-INF/classes/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/firstWSDLpackage/(in = 0) (out= 0)(stored 0%)
adding: WEB-INF/classes/firstWSDLpackage/firstWSDL.class(in = 856) (out= 480)(deflated 43%)
ignoring entry META-INF/
ignoring entry META-INF/MANIFEST.MF
adding: index.html(in = 446) (out= 256)(deflated 42%)
root@jonas-VirtualBox:/home/jonas/Documents/JAVA/8/firstWSDL/build/web#
```

V příslušné složce přibyl .war archiv



1.5. Deploynutí projektu.

Při prvním deploynutí se musíme přihlásit na GlassFish server

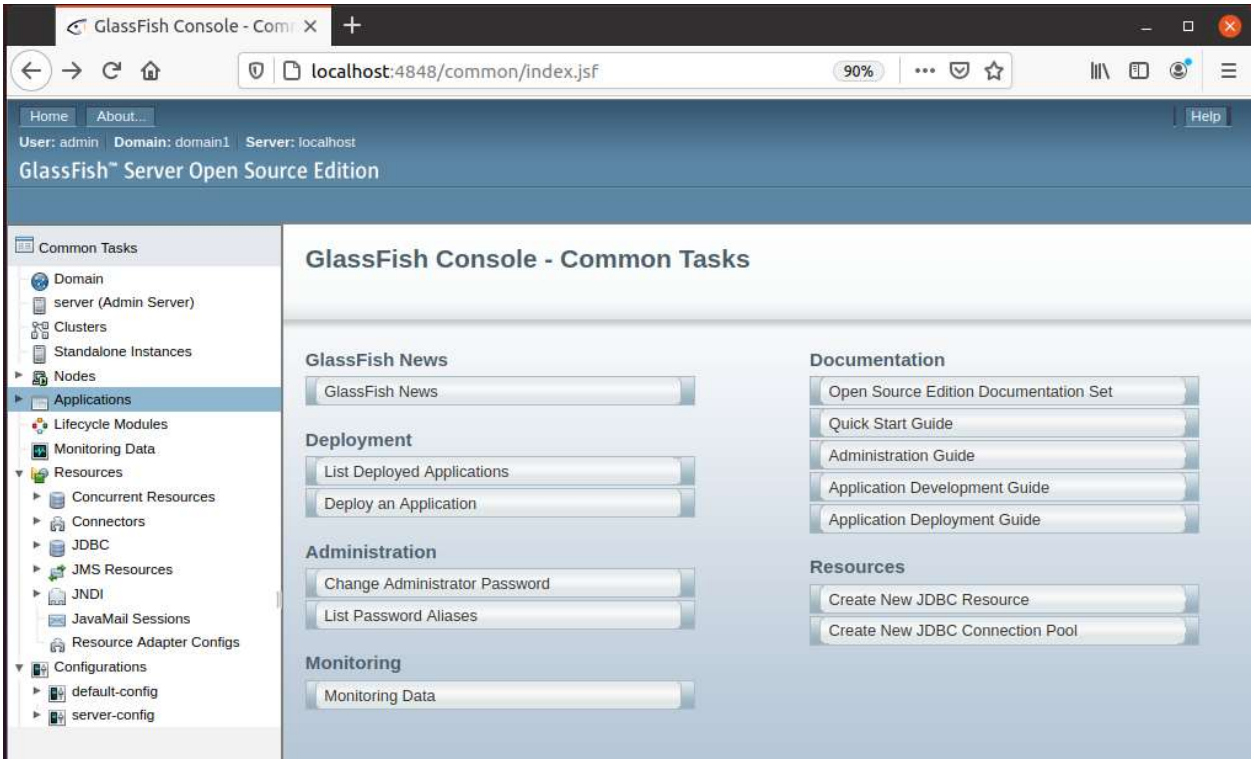
Ve složce umístění server glassfish5/bin spustíme script ./asadmin a následně start-domain

```
root@jonas-VirtualBox:/home/jonas/server/glassfish5/bin# ./asadmin
Use "exit" to exit and "help" for online help.
asadmin> start-domain
Waiting for domain1 to start .....
Successfully started the domain : domain1
domain  Location: /home/jonas/server/glassfish5/glassfish/domains/domain1
Log File: /home/jonas/server/glassfish5/glassfish/domains/domain1/logs/server.log
Admin Port: 4848
Command start-domain executed successfully.
asadmin> deploy /home/jonas/Documents/JAVA/8/firstWSDL/build/web/firstWSDL8.war
Application deployed with name firstWSDL8.
Command deploy executed successfully.
asadmin>
```

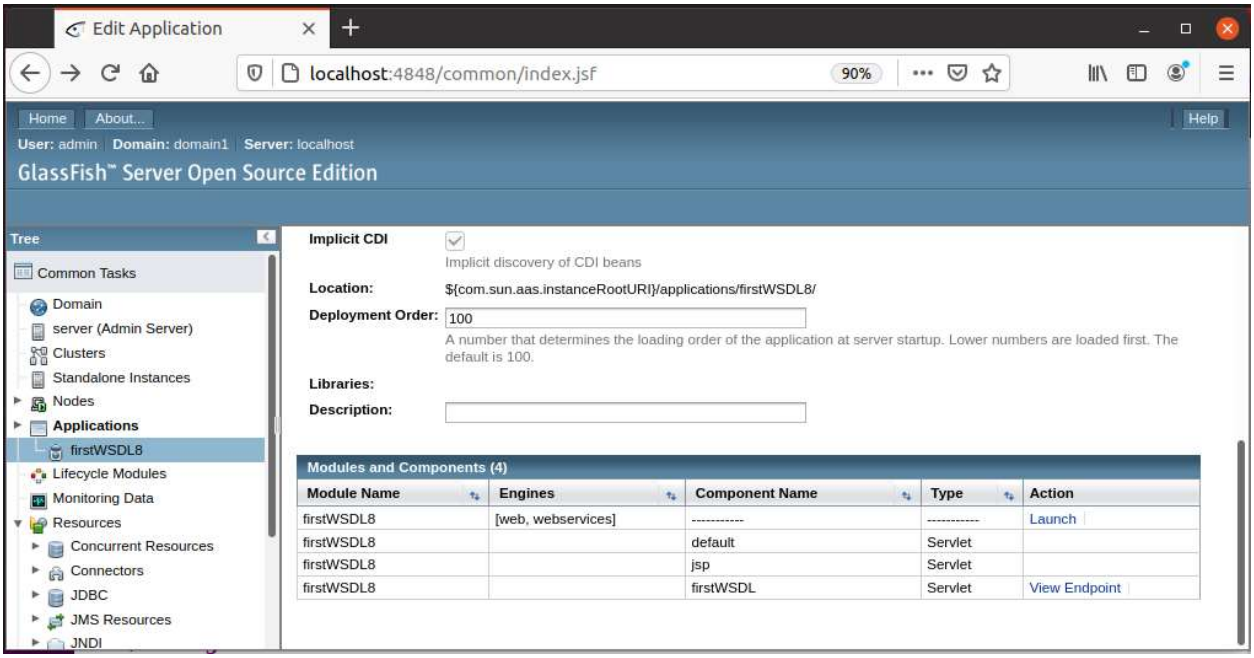
1.6. Otestování projektu.

Zadáme do webového prohlížeče url:

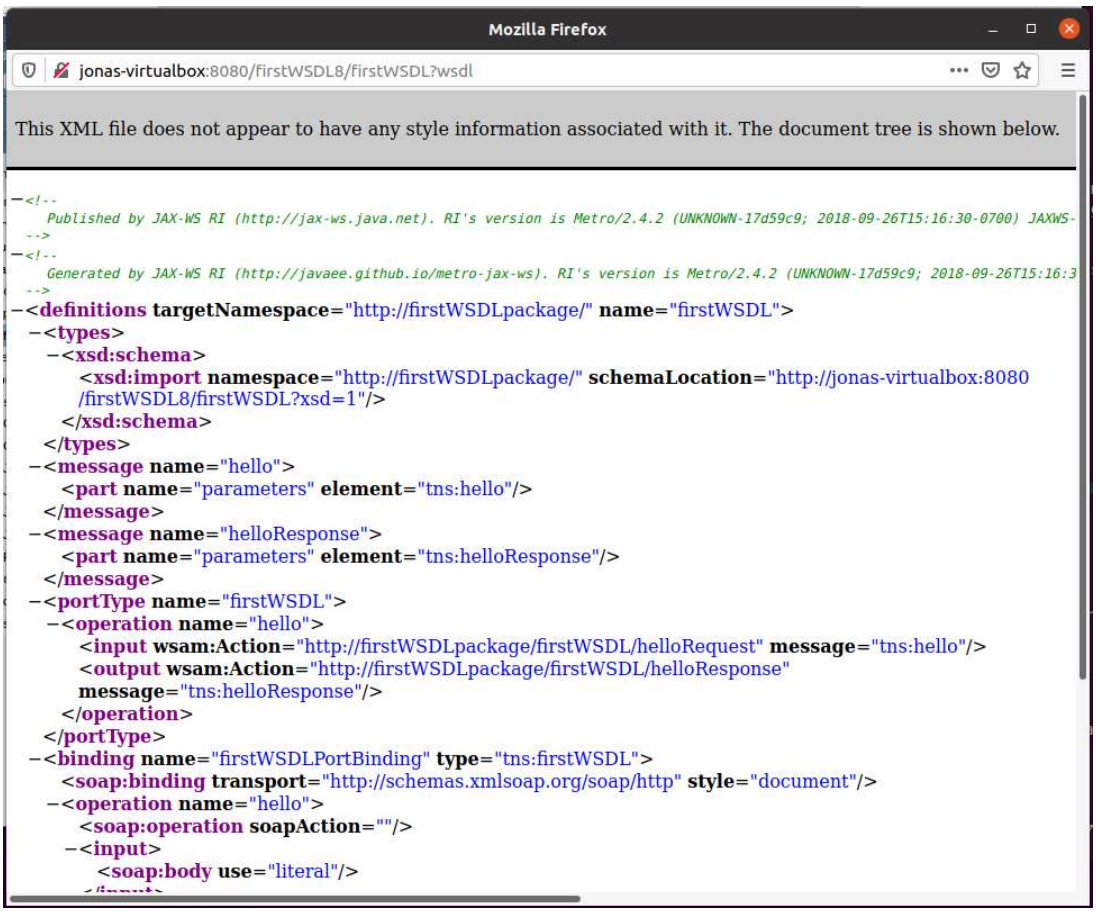
Localhost:4848



Následně rozklikneme firstWSDL8



Klikneme na View EndPoint a následně na WSDL, výsledek je následující.



1.7. Účel scriptu na promazávání build.xml.

Script promazává build.xml, tak aby projekt byl stále spustitelný. Níže zobrazujeme ukázku build.xml

```
19      </target>
20      <target depends="-pre-init,-init-private,-init-libraries,-init-user" name="-init-project">
21          <property file="nbproject/project.properties"/>
22      </target>
23      <target depends="-pre-init,-init-private,-init-libraries,-init-user,-init-project,-init-macr
24          <condition property="have.sources">
25              <or>
26                  <available file="${src.dir}"/>
27              </or>
28          </condition>
29          <available file="${conf.dir}/MANIFEST.MF" property="has.custom.manifest"/>
30          <property name="build.meta.inf.dir" value="${build.web.dir}/META-INF"/>
31      </target>
32      <target depends="init" name="-init-cos" unless="deploy.on.save">
33      </target>
34      <target name="-post-init">
35      </target>
36      <target depends="-pre-init,-init-private,-init-libraries,-init-user,-init-project,-do-init"
37      </target>
38      <target name="-init-macrodef-property">
39      </target>
40      <target depends="-init-ap-cmdline-properties" if="ap.supported.internal" name="-init-macrode
41          <macrodef name="javac" uri="http://www.netbeans.org/ns/web-project/2">
42              <attribute default="${src.dir}" name="srcdir"/>
43              <attribute default="${build.classes.dir}" name="destdir"/>
44              <attribute default="${build.generated.sources.dir}/ap-source-output" name="apgenerat
45              <attribute default="${includes}" name="includes"/>
46              <attribute default="${empty.dir}" name="gensrcdir"/>
47              <sequential>
48                  <property location="${build.dir}/empty" name="empty.dir"/>
49                  <mkdir dir="${empty.dir}"/>
50                  <mkdir dir="@{apgeneratedsrcdir}"/>
51                  <javac debug="@{debug}" deprecation="${javac.deprecation}" destdir="@{destdir}"
52                      <compilerarg value="-processorpath"/>
```

Po spuštění scriptu, pak build.xml vypadá následovně

```
20  <target depends="-pre-init,-init-private,-init-libraries,-init-user" name="-init-project">
21  </target>
22  <target depends="-pre-init,-init-private,-init-libraries,-init-user,-init-project,-init-macrodef-property" name="-do-init">
23  </target>
24  <target depends="init" name="-init-cos" unless="deploy.on.save">
25  </target>
26  <target name="-post-init">
27  </target>
28  <target depends="-pre-init,-init-private,-init-libraries,-init-user,-init-project,-do-init" name="-init-check">
29  </target>
30  <target name="-init-macrodef-property">
31  </target>
32  <target depends="-init-ap-cmdline-properties" if="ap.supported.internal" name="-init-macrodef-javac-with-processors">
33  </target>
34  <target depends="-init-ap-cmdline-properties" name="-init-macrodef-javac-without-processors" unless="ap.supported.internal">
35  </target>
36  <target depends="-init-macrodef-javac-with-processors,-init-macrodef-javac-without-processors" name="-init-macrodef-javac">
37  </target>
38  <target if="${junit.available}" name="-init-macrodef-junit-init">
39  </tagget>
40  <target name="-init-test-properties">
41  </target>
42  <target if="${nb.junit.single}" name="-init-macrodef-junit-single" unless="${nb.junit.batch}">
```

Podotkněme, že projekt je stále spustitelný.

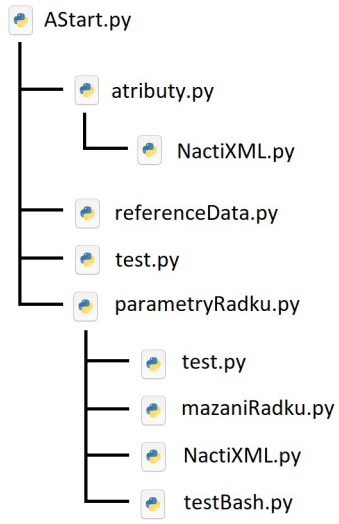
2. Dokumentace testovacího scriptu.

Zdrojový kód scriptu je zde:

https://drive.google.com/drive/folders/1fLFkYzVwU6Ju0swap-ndfeMf8Xjbdulr

2.1. Struktura kódu.

Kód je rozčleněn do následujících souborů:



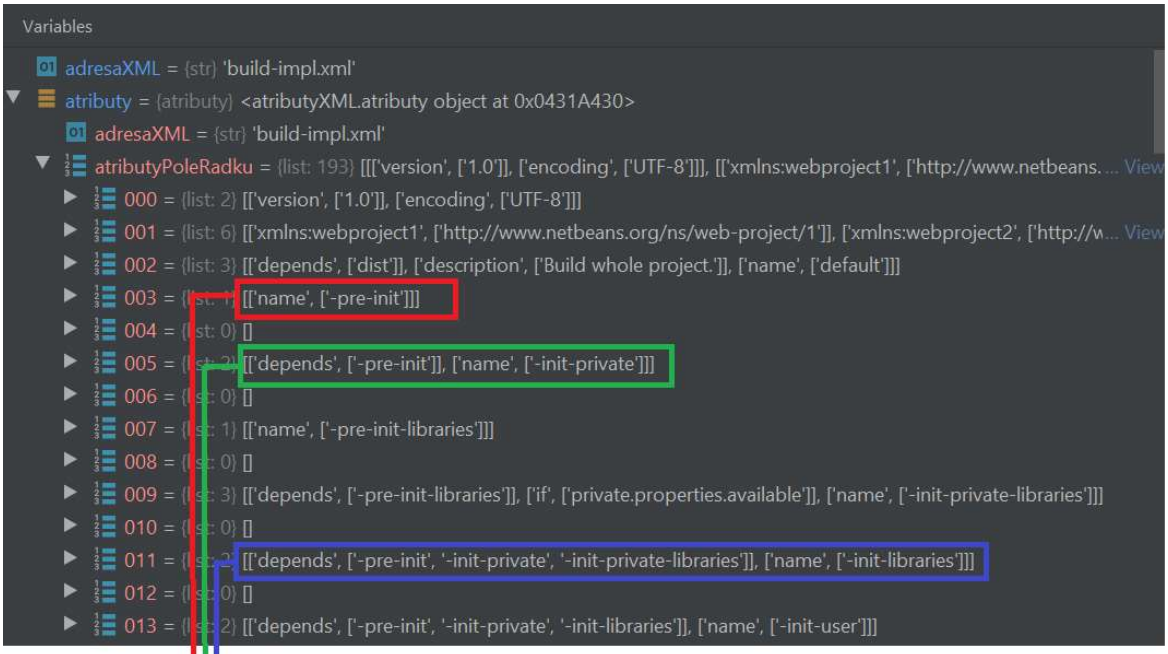
2.2. Základní data.

Script spustíme spuštěním souboru AStart.py

Ve scriptu umístíme breakpoint na řádek:
`atributyXMLSouboru = atributy.getAtributyPoleRadku()`

```
import atributyXML
import referenceData
import parametryRadku
import test


adresaXML = "build-impl.xml"
posledniIndexCyklu = 0

# v kazdem cyklu nacte a zapise XML
# pokazde nacita XML znovu, tak aby data byla aktualni
for cyklus in range(1, 10):

    # ziska atributy z xml
    atributy = atributyXML.atributy(adresaXML)
    atributyXMLSouboru = atributy.getAtributyPoleRadku()
    poleRadkuXML = atributy.getPoleRadkuXML()
```

a obdržíme data, obrázek níže porovnává data v debugu s načteným xml.




```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <project xmlns:webproject1="http://www.netbeans.org/ns/web-project/1" xmlns:webproject2="http://
4      <target depends="dist" description="Build whole project." name="default"/>
5      <!--
6                  INITIALIZATION SECTION
7          -->
8      <target name="-pre-init">
9      </target>
10     <target depends="-pre-init" name="-init-private">
11     </target>
12     <target name="-pre-init-libraries">
13     </target>
14     <target depends="-pre-init-libraries" if="private.properties.available" name="-init-private-
15     </target>
16     <target depends="-pre-init,-init-private,-init-private-libraries" name="-init-libraries">
17     </target>
18     <target depends="-pre-init,-init-private,-init-libraries" name="-init-user">
19     </target>
20     <target depends="-pre-init,-init-private,-init-libraries,-init-user" name="-init-project">
21         <property file="nbproject/project.properties"/>
22     </target>
23     <target depends="-pre-init,-init-private,-init-libraries,-init-user,-init-project,-init-macr
24         <condition property="have.sources">
25             <or>
26                 <available file="${src.dir}"/>
27             </or>
28         </condition>
29         <available file="${conf.dir}/MANIFEST.MF" property="has.custom.manifest"/>
30         <property name="build.meta.inf.dir" value="${build.web.dir}/META-INF"/>
31     </target>
```

V modulu atributyXML.py se tedy převede XML do objektové podoby, s kterou se pak následně dale pracuje.

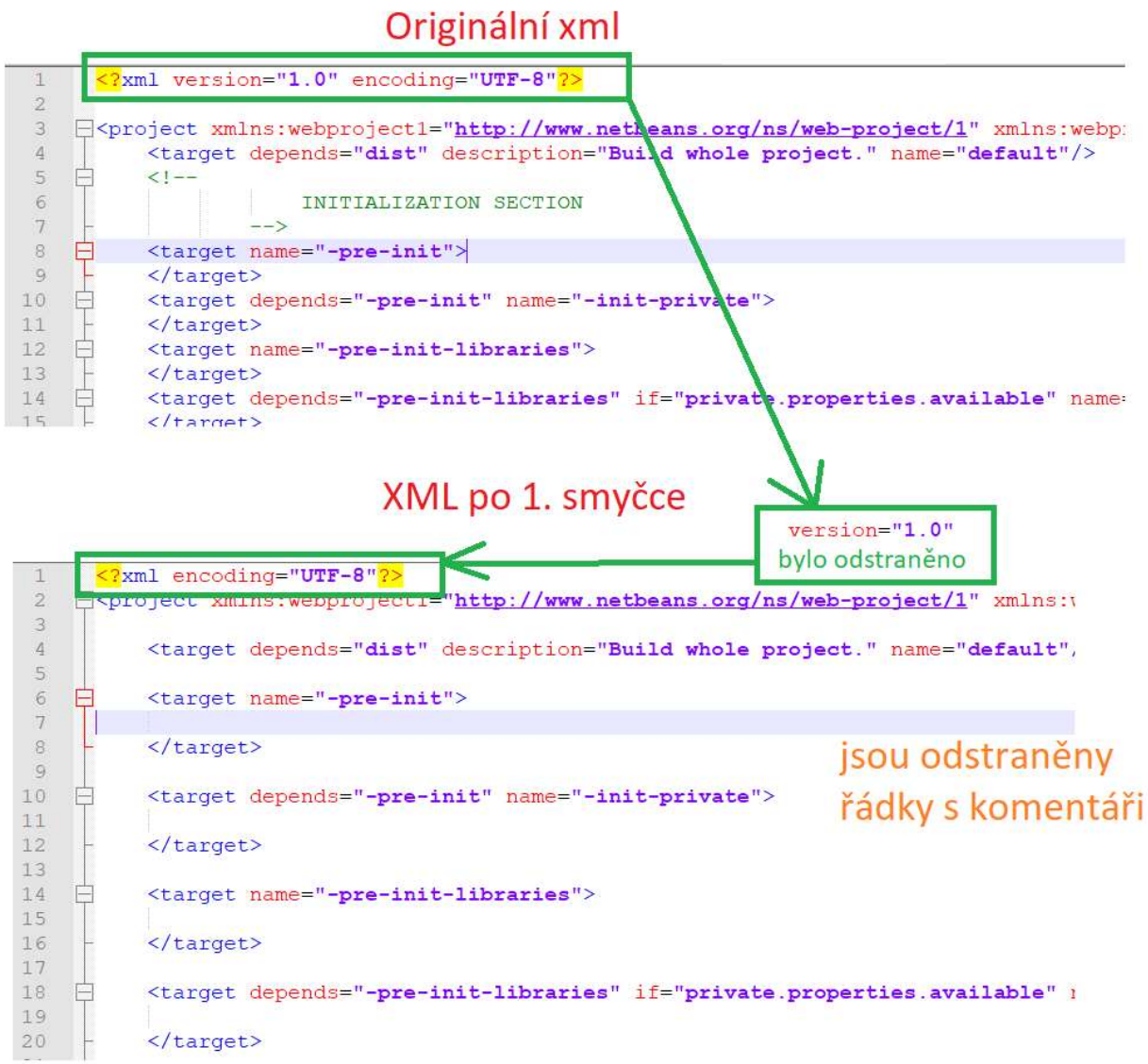
2.3. Postupné promazávání Build.xml.

Následně můžeme umístit breakpoint do modulu parametryRadku.py např. na poslední řádek smyčky:

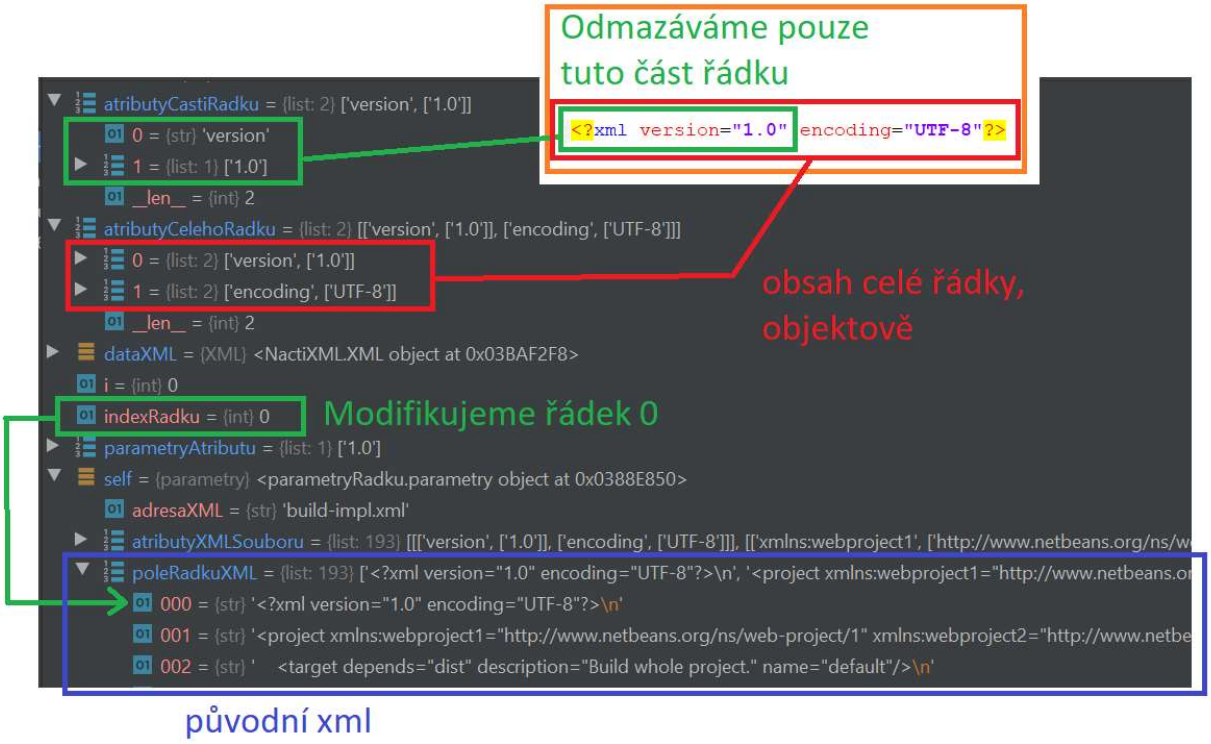
```
def prochazejParametryDanehoRadku(self, indexRadku):

    atributyCelehoRadku = self.atributyXMLSouboru[indexRadku]

    for i in range(0, len(atributyCelehoRadku)):
        atributyCastiRadku = atributyCelehoRadku[i]
        atribut = atributyCastiRadku[0]
        parametryAtributu = atributyCastiRadku[1]

        # nacte XML s radky znovu
        dataXML = NactiXML.XML(self.adresaXML)
        self.poleRadkuXML = dataXML.getPoleRadkuXML()

        # v metode opravi vsechny radky pro zadany parametr
        self.prochazejParametryDanehoAtributu(parametryAtributu, indexRadku)

        # zapise XML
        self.zapisXML(self.poleRadkuXMLNew)
        print()
```

A můžeme sledovat postupné promazávání build.xml:

V prvním cyklu dosáhneme změny:


Originální xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns:webproject1="http://www.netbeans.org/ns/web-project/1" xmlns:webp
<target depends="dist" description="Build whole project." name="default"/>
<!--
INITIALIZATION SECTION
-->
<target name="-pre-init">
</target>
<target depends="-pre-init" name="-init-private">
</target>
<target name="-pre-init-libraries">
</target>
<target depends="-pre-init-libraries" if="private.properties.available" name=
XML po 1. smyčce
version="1.0"
bylo odstraněno
<?xml encoding="UTF-8"?>
<project xmlns:webproject1="http://www.netbeans.org/ns/web-project/1" xmlns:
<target depends="dist" description="Build whole project." name="default",
<target name="-pre-init">
</target>
<target depends="-pre-init" name="-init-private">
</target>
<target name="-pre-init-libraries">
</target>
<target depends="-pre-init-libraries" if="private.properties.available"
</target>
jsou odstraněny
řádky s komentáři

Pak v debugu můžeme pozorovat data:



2.4. Vlastní redukce build.xml.

Redukce probíhá po jednotlivých řádcích. V Modulu ParametryRadku.py běží kód v jednotlivých smyčkách po řádcích a v každé smyčce vykonává akce:

1) Uloží si originální build.xml, pro případ, že test neproiběhne úspěšně, aby mohl vrátit data.
2) Redukuje build.xml – tzn. odmaže příslušný atrybut nebo parametr z xml.
3) Otestuje, zda se dá projekt sbuildovat, zda je spustitelný a zda vrací stejná data, jako v předchozím cyklu

4a) Pokud vysledekTestu = True, pak build.xml ponechá a žádné kroky již nevykonává.
4b) Pokud vysledekTestu = False, pak redukovaný xml přepíše originálním xml.

vysledekTestu je návratová hodnota z modulu testBash.py

2.5. Vlastní test nové verze xml.

Vlastní test probíhá v modulu testBash.py, při každém cyklu kód prochází:

```
def test(self):

    # odstrani slozku build, pokud existuje
    self.odstranBuild()

    #zavola jednotlive testy
    self.statusBuild = self.testBuild()

    # pokud jsou data stromu prazdna, pak je pouze ziska, jinak je kontroluje

    if(self.treeDefault == ""):
        self.statusTree = True
        # ziska data stromu
        self.treeDefault = self.vratTree()
        print(self.treeDefault)
    else:
        if(self.statusBuild == True):
            self.statusTree = self.testTree()

    if(self.statusTree == True):
         self.statusWar = self.testWar()

    if(self.statusWar == True):
         self.statusDeploy = self.testDeploy()

    # pokud jsou data WSDL prazdna, pak je pouze ziska, jinak je kontroluje

    if(self.clientDefault == ""):
        self.statusClient = True
        # ziska data stromu
        self.clientDefault = self.vratWSDL()
        #self.vytvorZalohu()
    else:
        if(self.statusDeploy == True):
            self.statusClient = self.testWSDL()
```

Ve výše uvedeném kódu test vykonává kroky:

1) Provede build projektu a detekuje zda build proběhne úspěšně -> v tom případě vrátí self.statusBuild = True
2) Otestuje zda stromová struktura projektu je konstantní jako v předchozím běhu (verzi) -> nastaví self.statusTree = True
3) Vytvoří archiv .war a otestuje zda byl vytvořen -> vrátí self.statusWar = True
4) Provede deploy na GlassFish a otestuje, zda byl projekt deploynut -> self.statusDeploy = True
5) Otestuje zda klient WSDL vrací stejná data, jako v předchozí verzi -> self.statusClient = True

Pokud alespoň jeden status je False, pak se test ukončí se statusem False

Pokud jsou všechny statusy True, pak test vrátí True. Status se vrátí do modulu ParametryRadku.py