Integraciją darome su Apache Camel: <http://camel.apache.org/>

Projektų buildinimo sistema yra maven: <http://maven.apache.org/>

Kodą laikome git repositorijose: <http://git-scm.com/>

Reikės Java 7 (Oracle arba OpenJDK)

Prie šio laiško yra prisegta "pasižaidimų" aplinka, kuri pakelia Camel, Jetty, Spring, palaiko CXF, ActiveMQ. Prisegtame archyve yra tik išeities tekstai. Visų pirma reiktų kodą susiimportuoti į git repozitoriją ir padaryti iš jo pirmą commit'ą. Kiekvieną užduotį reiktų padaryti atskiroje git repozitorijos atšakoje (branch'e) (nors viena nuo kitos jos gali priklausyti, nes daugumoje užduotys iteracinės), arba kiekvieną užduoties veikiantį variantą užtag'inti (kad būtų aišku, kur yra konkreti užduotis).

Camel direktyvas reikia rašyti su Java DSL kad ir pvz.: **src/main/java/lt/inventi/playground/PlaygroundRouteBuilder.java** klasėje.

Klasė, kurią reikia paleisti, yra **lt.inventi.jetty.JettyLauncher** (test dalyje).

**Užduotys:**

**1.** Panaudoti Timer (http://camel.apache.org/timer.html) komponentą taip, kad kas 2 sekundes į konsolę (stdout) būtų išvedama "Hello world!" (išvedimui naudoti Processor arba Bean)

**2.** Pagal Timer'į generuojamas atsitiktinis skaičius (bent 3 variantus, pvz. daugiau už X, daugiau už Y, kitaip) ir priklausomai nuo jo reikšmės, veiksmas perduodamas į kitus "route'us" (direct). Juose išvedamas kažkoks (kiekvienam atvejui skirtingas) tekstas į konsolę bei pati atsitiktinio skaičiaus reikšmė. Iš Java kodo gali būti išvedamas tik tekstas bei skaičius, o sąlyga turėtų būti tikrinama su Camel DSL direktyvomis (reikia susirasti, kokiomis).

**3.** Panaudoti Camel'io komponentą, kuris nuskaito ir rašo į failus. Jeigu failas buvo tuščias, nieko nedaroma. Kitaip reikia failo turinį paversti mažosiomis raidėmis, pridėti savo kažkokią signatūrą ir viską įrašyti į kitą failą su "-result" priedėliu prie failo vardo (prieš failo išplėtimą).

Užduočiai atlikti reikia kaip galima labiau naudoti Camel priemones, kur tik įmanoma (ir pavyks rasti). Privaloma panaudoti "simple" (ir "file") kalbą.

<http://camel.apache.org/simple.html>

<https://camel.apache.org/maven/current/camel-core/apidocs/org/apache/camel/builder/ValueBuilder.html>

(tai ką grąžina pvz. body() ir kt.)

**4.** Paimti duomenis iš failo, prie kiekvienos jo eilutės teksto prirašyti eilutės numerį (skaičiavimas prasideda nuo 1) ir galutinį rezultatą įrašyti į "result" failą (vardas kaip ir 3-ioje užduotyje). Viską padaryti Camel DSL priemonėmis be Processor'iaus ar Bean'o. Reikės panaudoti bent <http://camel.apache.org/splitter.html>

**Web servisai (CXF ir Soap UI):**

**5.** Reiktų susikurti paprastą Ping Pong web servisą (serverio pusę). Jo WSDL'ą galima rasti čia:

<https://github.com/apache/ode/blob/master/axis2-war/src/test/resources/TestClusteredOutgoingTraffic/PingPong.wsdl>

Jį reikia įsidėti į src/main/resources direktoriją, application context'e apsirašyti CXF endpoint'ą, pvz:

<cxf:cxfEndpoint id="pingPongEndpoint"

address="/Ping"

wsdlURL="PingPong.wsdl" endpointName="s:PingPong"

serviceName="s:PingPongService"

xmlns:s="http://pingpong.bpel.tps">

<cxf:inInterceptors>

<bean class="org.apache.cxf.transport.common.gzip.GZIPInInterceptor"/>

</cxf:inInterceptors>

<cxf:properties>

<entry key="dataFormat" value="PAYLOAD"/>

</cxf:properties>

</cxf:cxfEndpoint>

Ir atitinkamai apsibrėžti Camel route'ą: <http://camel.apache.org/cxf.html>.

Čia užtenka, kad iškviestas servisas grąžintų tą, ką gavo (nors tai nebus validus atsakymas).

Vietoje Spring XML'o galima naudoti Spring JavaConfig.

Servisą reikia išsikviesti su Soap UI: <http://www.soapui.org/>.

**6.** Reikia padaryti, kad servisas grąžintų normalų validų atsakymą, t.y. pong elemente tą patį tekstą, kuris buvo ping elemente. Atsakymą formuoti ir/arba ping tekstą išsitraukti dviem būdais:

**6a)** Su XPath:

<http://camel.apache.org/xpath.html>

**6b)** Su JAXB:

<http://cxf.apache.org/docs/maven-cxf-codegen-plugin-wsdl-to-java.html>

<http://camel.apache.org/jaxb.html>

**7.** Soap UI sukurti ir sukonfigūoti "SOAP Mock Service" taip, kad jis atliktų tą patį, kaip ir "Playground" servisas 6-oje užduotyje, tik prie "ping" teksto "pong'e" prirašyti tekstą "(from Soap UI)".

**8.** Įgyvendinti tokią grandinę:

\* Iš Soap UI kviečiamas "Playground" Ping Pong servisas.

\* Playground (CXF komponentu) kviečia Soap UI PingPong mock servisą.

\* Playground grąžiną tą patį, ką grąžina Soap UI mock servisas.

\* Taip atsakymas grįžta Soap UI request'ui.

**9.** Parašyti Soap UI testą, kuris patikrintų, kad tekstas, kurį jis nusiunčia "ping'e", jis gauna atgal "pong'e". Testas kviestų Playground PingPong servisą.

**Eilės (advanced):**

**10.** Tas pats, kas ir 4 užduotyje, tik pirma rezultatų failas turi atsigulti kažkokioje ActiveMQ eilėje, o iš jos kitas "route'as" turi turinį įrašyti į "result" failą (pavadinimo sudarymo taisyklės išlieka).

<http://camel.apache.org/jms.html>

<http://camel.apache.org/activemq.html>