

Java client implementation guide

We will use Maven and Spring WS to create simple Web Service client.

- Configure Maven dependencies:

```

                                pom.xml
<project...>
  <dependencies>
    <!-- JAXB dependencies -->
    <dependency>
      <groupId>javax.xml.bind</groupId>
      <artifactId>jaxb-api</artifactId>
      <version>2.2.2</version>
    </dependency>
    <dependency>
      <groupId>com.sun.xml.bind</groupId>
      <artifactId>jaxb-impl</artifactId>
      <version>2.2.2</version>
    </dependency>

    <!-- Spring WS dependencies -->
    <dependency>
      <groupId>org.springframework.ws</groupId>
      <artifactId>spring-ws-core</artifactId>
      <version>2.0.1.RELEASE</version>
    </dependency>
    <dependency>
      <groupId>org.springframework.ws</groupId>
      <artifactId>spring-ws-security</artifactId>
      <version>2.0.1.RELEASE</version>
    </dependency>
    <dependency>
      <groupId>org.apache.ws.commons.schema</groupId>
      <artifactId>XmlSchema</artifactId>
      <version>1.4.7</version>
    </dependency>

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-oxm</artifactId>
      <version>${spring.version}</version>
    </dependency>

  </dependencies>
</project>
```

- Download WS XSD schemas to project directory `src/main/resources/schema`
 - <http://solargis.info/schema/common-types.xsd>
 - <http://solargis.info/schema/common-geo.xsd>
 - <http://solargis.info/schema/common-pv.xsd>
 - <http://solargis.info/schema/ws-pvplanner.xsd>
 - <http://solargis.info/schema/ws-data.xsd>

- <http://solargis.info/schema/data-request.xsd>
- Configure JAXB to generate classes:

pom.xml

```
<project...>
  <build>
    <plugins>
      <plugin>
        <groupId>org.jvnet.jaxb2.maven2</groupId>
        <artifactId>maven-jaxb2-plugin</artifactId>
        <version>0.7.5</version>
        <executions>
          <execution>
            <id>jaxb-ws-pvplanner</id>
            <goals>
              <goal>generate</goal>
            </goals>
          </execution>
        </executions>
        <configuration>
          <schemaDirectory>src/main/resources/schema<
/schemaDirectory>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

- Define Spring WebServiceTemplate context:

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:oxm="http://www.springframework.org/schema/oxm"
       xmlns:p="http://www.springframework.org/schema/p"
       xsi:schemaLocation="http://www.springframework.org/schema/oxm
http://www.springframework.org/schema/oxm/spring-oxm.xsd
       http://www.springframework.org/schema/beans http://www.
springframework.org/schema/beans/spring-beans.xsd">

  <!-- contextPath - package where to find request and response
classes to marshall / unmarshal
       if request and response are in different packages, declare
marshaller and unmarshaller separately -->
  <oxm:jaxb2-marshaller id="jaxb2marshaller" contextPath="eu.geomodel.
schema.ws.pvplanner"/>

  <bean id="pvPlannerTemplate" class="org.springframework.ws.client.
core.WebServiceTemplate"
       p:marshaller-ref="jaxb2marshaller" p:unmarshaller-ref="
jaxb2marshaller"
       p:defaultUri="https://solargis.info/ws/soap/pvPlanner"> <!--
web service endpoint uri -->
  <property name="interceptors">
    <list>
      <bean class="org.springframework.ws.soap.security.wss4j.
Wss4jSecurityInterceptor"
           p:securementActions="Timestamp UsernameToken"
           p:securementUsername="demo"
           p:securementPassword="demo"/>
    </list>
  </property>
</bean>
</beans>

```

- Define client Spring bean:

```

// imports omitted
public class PvPlannerWsClient {

    @Autowired
    @Qualifier("pvPlannerTemplate")
    WebServiceTemplate template;

    public CalculateResponse callPvPlanner(/*some args*/) {
        return (CalculateResponse) template.marshalSendAndReceive
(prepareRequest(/*some args*/));
    }

    private CalculateRequest prepareRequest(/*some args*/) {
        // just sample request
        CalculateRequest request = new CalculateRequest();
        Location location = new Location();
        location.setLat(48.612590); // location of demo site
        location.setLng(20.827079);
        request.setLocation(location);
        PvSystem system = new PvSystem();
        system.setInstalledPower(1);
        system.setModuleType(ModuleTypeEnum.C_SI);
        Settings settings = new Settings();
        settings.setInverterEfficiency(97.5);
        settings.setDcLosses(5.5);
        settings.setAcLosses(1.5);
        settings.setAvailability(99);
        system.setSettings(settings);
        MountingFixedRoofMounted mounting = new
MountingFixedRoofMounted();
        mounting.setAzimuth(175);
        mounting.setInclination(45D);
        system.setMounting(mounting);
        request.setSystem(system);
        return request;
    }
}

```