How to include the CERIF packages in your JAVA project

There are two distinct JAVA projects, the *cerif-jpa-model* which represents the CERIF data model and the *cerif-jpa-services* which can be used to accommodate queries and custom functions and provides a mechanism for the programmer to access them. Each inner package in *cerif-jpa-services* provides a transfer object class for multilingual fields.

The *cerif-jpa-model* package doesn't need any editing or preparation. It can be used as is.

The package *cerif-jpa-services* depends on the model package through a pom dependency entry (already present in its pom.xml).

The app-context-cerif-jpa-services.xml file in *cerif-jpa-services* is used to initialize the db connection either with a dataSource bean or with a jee:jndi-lookup bean.

Any client project, e.g. a UI app, must include as a pom dependency the *cerif-jpa-services* package.

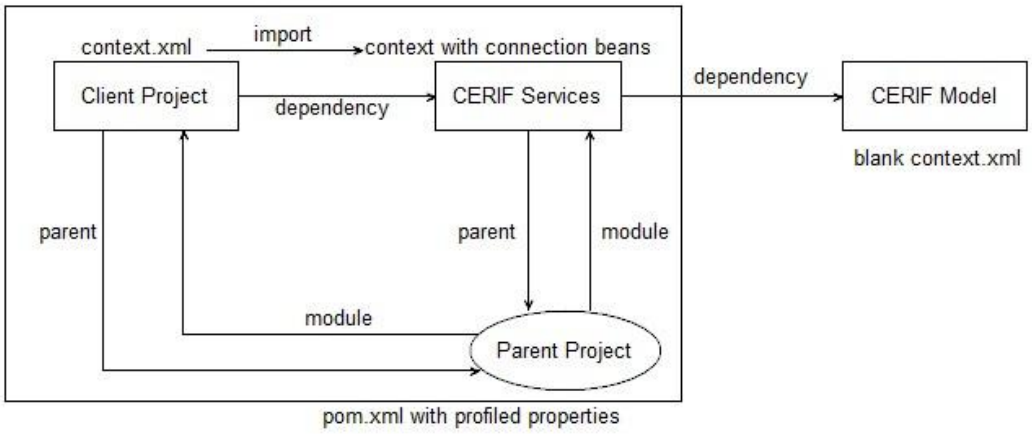
```
<dependency>
      <groupId>gr.ekt</groupId>
      <artifactId>cerif-jpa-services</artifactId>
      <version>0.1</version>
      <type>jar</type>
</dependency>
```

The needed queries and/or tranfer object classes can be added in the *cerif-jpa-services* package.

The context xml of the client project must include an import command for the context xml file of the *cerif-jpa-services* package e.g.

```
<beans:import resource="classpath:/META-INF/spring/app-context-cerif-jpa-services.xml"/>
```

Example project



The project relationships between individual packages can be structured as in the image above. A parent project can contain the client project and the services package. The parameters for the application can be stored in the parent's project pom.xml file inside pom profiles, e.g.

```
<profiles>
	<profile>
		<id>profile_id1</id>
		<activation>
			<activeByDefault>false</activeByDefault>
		</activation>
		<properties>
			<target.version>0.1</target.version>
			<db.driver>com.mysql.jdbc.Driver</db.driver>
			<db.url>jdbc:mysql://localhost:3306/db?characterEncoding=utf8</db.ur
		l>
			 <db.username>root</db.username>
			 <db.password>pass</db.password>

			 <!-- validate | update | create | create-drop -->
			<hibernate.hbm2ddl.auto>create</hibernate.hbm2ddl.auto>
			<hibernate.dialect>org.hibernate.dialect.MySQL5InnoDBDialect</hibern
			ate.dialect>
			 <hibernate.show_sql>true</hibernate.show_sql>
		</properties>
	</profile>
<profiles>
```

The parameters can then be passed in the app-context-cerif-jpa-services.xml as *${}* beans parameters.

The pom.xml of the parent project must contain the following

```
<modules>
	<module>cerif-jpa-services</module>
	<module>client-project-artifactId</module>
</modules>
```

The services' package pom.xml file and the client's project pom.xml must include the following:

```
<parent>
	<groupId>parent's version groupId</groupId>
	<artifactId>parent's version artifactId</artifactId>
	<version>parent's version</version>
</parent>
```

This example is a suggestion on how a project using the packages can be set. Surely it's not obligatory to follow this approach.

Example code

```
//create a project with a project title

@Autowired
private PersistenceService service;


//in a function

Project proj = new Project();
proj.setAcronym("ACRO");
service.save(proj);
ProjectTitle projt = new ProjectTitle(proj, Language.ENGLISH, Translation.HUMAN,
"Title");
service.save(projt);



//Find an Organisation Unit and edit it

//inside OrganisationUnitCrudRepository interface in services package

@Query(value= "select o from gr.ekt.cerif.entities.base.OrganisationUnit o where o.id =
?1")
OrganisationUnit findById(Long id);

//inside OrganisationUnitRepository interface in services package

OrganisationUnit findById(Long id);

//inside OrganisationUnitRepositoryImpl class in services package

@Autowired
private OrganisationUnitCrudRepository organisationUnitCrudRepository;

public OrganisationUnit findById(Long id) {
	return organisationUnitCrudRepository.findById(id);
}



//in a function

OrganisationUnit org =
service.getBaseService().getOrganisationUnitRepository().findById((long)100);
if (org!=null) {
	org.setAcronym("ACRO");
	service.save(org);
}
```