

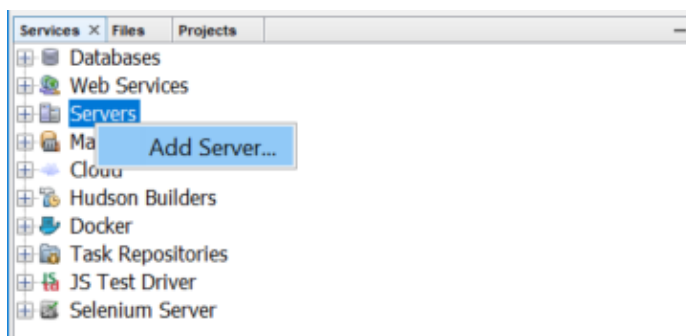
BI-TJV semestrálka

Pre Filipka od Ivetky a Tomáška

1. Stiahneš si Glassfish a pokiaľ chceš pracovať s Oraclovskou databázou tak aj ojdbc driver

Glassfish 5.0 si stiahneš na

<https://drive.google.com/file/d/1u9ScfARD0nIMl9QutLBOFUKVm3wAh1Bp/view> a pridáš si ho do Netbeansov – Services, Servers, pravým na Add Server a vyberieš zložku glassfish



Ovládač stiahneš na

<https://moodle.fit.cvut.cz/course/format/wiki/mediafile.php?id=61&path=%2ftutorials%2f06%2fojdbc7.jar.zip> a pridáš ho do zložky glassfish/lib (tam kde sú aj ostatné .jar súbory).

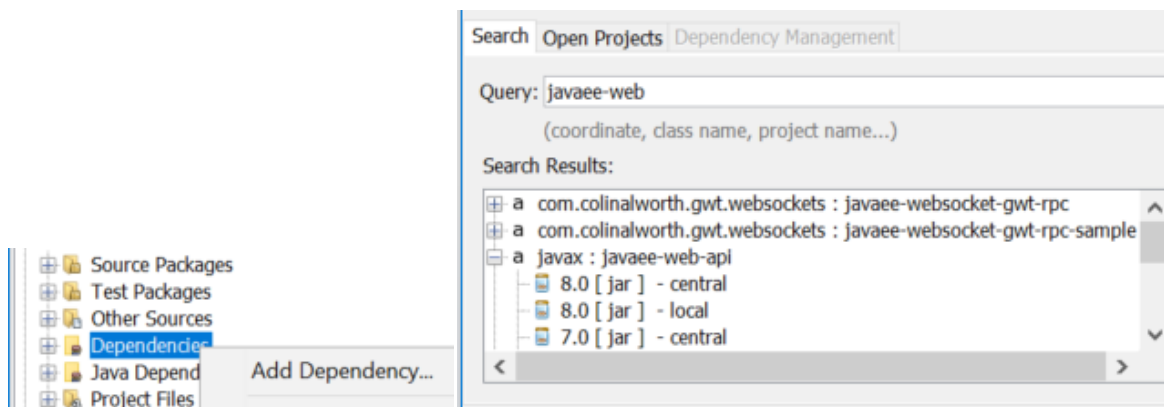
2. Vytvorím projekt s entitnými triedami

Založíš nový Maven Java Application projekt.

File -> New Project -> Maven -> Java Application

Do projektu pridáš novú dependency na javaee-web-8.0 (kvôli anotáciám a pod.).

Pravým na Dependencies -> Add Dependency -> vyhladáš javaee-web a zvolíš správnu verziu



Vytvoríš entitné triedy a pre každú napíšeš prázdny konštruktor.

V danej entitnej triede stlačíš Alt+Insert -> Generate Constructor -> nič nezaklikávaj a klikni na Generate

Pre každú entitu vytvoríš obalovú triedu. Tá bude obsahovať jednu premennú typu List<Entita>, k nej vygeneruješ getter a setter. Za názov obalovej triedy nezabudni dopísať implements Serializable.

```
public class ZakaznikBox implements Serializable {  
    private List<Zakaznik> zakaznici = new ArrayList();  
}
```

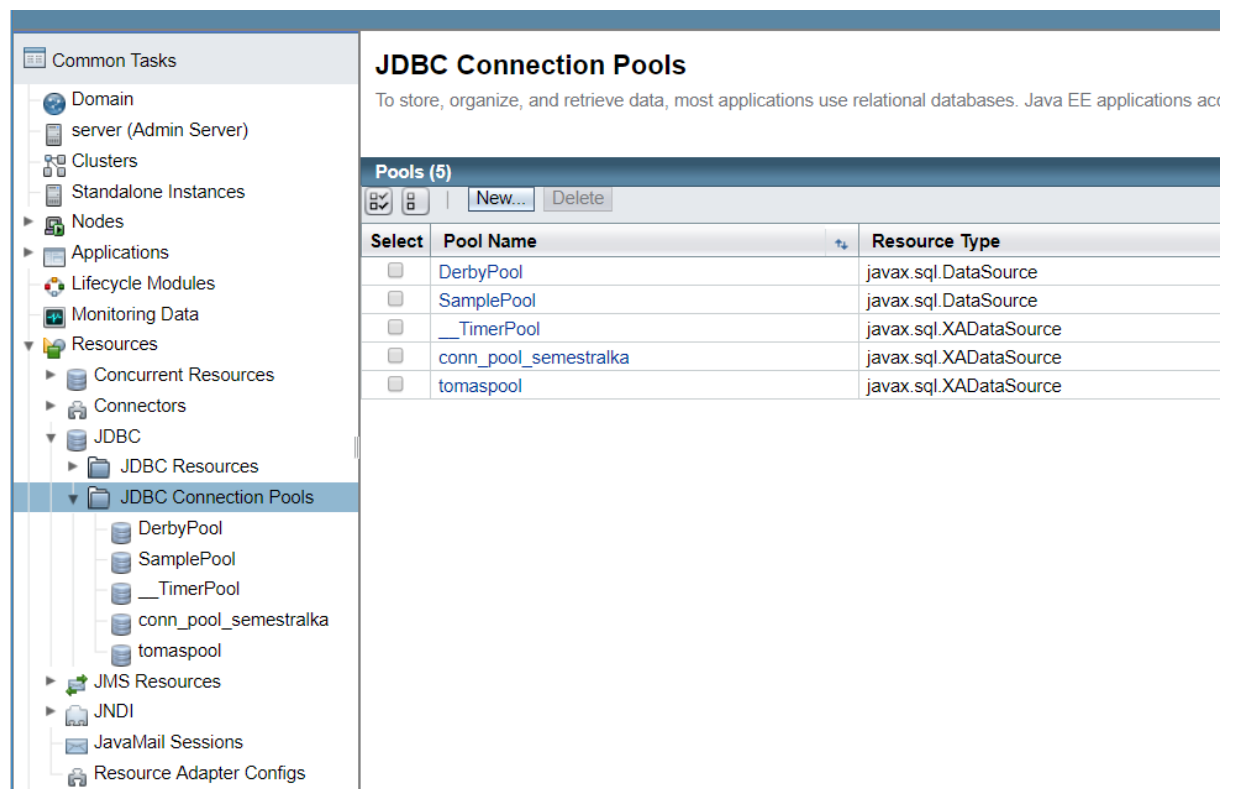
Pri komplikáciách skús Clean and build / vyresetovať Glassfish / zatvoriť Netbeans.

3. Vytvorím connection pool a resource

Naštartuješ Glassfish a v prehliadači otvoríš url localhost:4848.

Vytvoríš JDBC connection pool.

JDBC Connection pools -> New -> ako Resource Type zvolíme XADataSource a Database Driver Vendor vyberieme Oracle -> Next -> dôležité je nastaviť properties User, Password (nájdeš na <https://profile.fit.cvut.cz/cs/>) a URL (hodnotou je jdbc:oracle:thin:@oracle.fit.cvut.cz:1521:ORACLE)



JDBC Connection Pools

To store, organize, and retrieve data, most applications use relational databases. Java EE applications acc

Pools (5)

Select	Pool Name	Resource Type
<input type="checkbox"/>	DerbyPool	javax.sql.DataSource
<input type="checkbox"/>	SamplePool	javax.sql.DataSource
<input type="checkbox"/>	__TimerPool	javax.sql.XADataSource
<input type="checkbox"/>	conn_pool_semestralka	javax.sql.XADataSource
<input type="checkbox"/>	tomaspool	javax.sql.XADataSource

Additional Properties (19)		
<input checked="" type="checkbox"/> <input type="checkbox"/> <input type="button" value="Add Property"/> <input type="button" value="Delete Properties"/>		
Select	Name	Value
<input type="checkbox"/>	TNSEntryName	
<input type="checkbox"/>	Description	
<input type="checkbox"/>	User	
<input type="checkbox"/>	MaxStatements	0
<input type="checkbox"/>	DatabaseName	
<input type="checkbox"/>	NativeXA	false
<input type="checkbox"/>	ImplicitCachingEnabled	false
<input type="checkbox"/>	NetworkProtocol	tcp
<input type="checkbox"/>	URL	
<input type="checkbox"/>	ConnectionCacheName	
<input type="checkbox"/>	DataSourceName	OracleXADatasource
<input type="checkbox"/>	LoginTimeout	0
<input type="checkbox"/>	ServiceName	
<input type="checkbox"/>	ServerName	
<input type="checkbox"/>	ONSConfiguration	
<input type="checkbox"/>	DriverType	
<input type="checkbox"/>	PortNumber	0
<input type="checkbox"/>	ExplicitCachingEnabled	false
<input type="checkbox"/>	Password	

Po vytvorení skúsiš ping (musí byť zelené ping succeeded).

Vytvoríš resource.

JDBC Resources -> New -> pre dodržanie konvencie by názov mal začínať na "jdbc/", ako Pool Name vyberieš connection pool z predošlého kroku

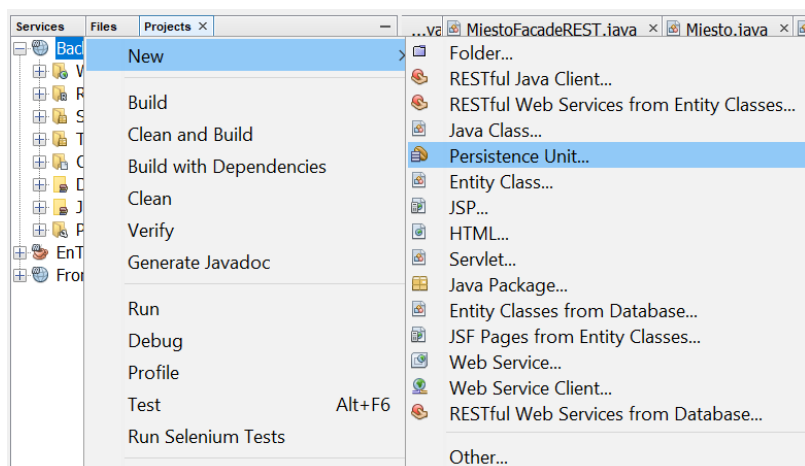
4. Vytvorenie projektu s RESTovou službou

Založíš nový Maven Web Application projekt.

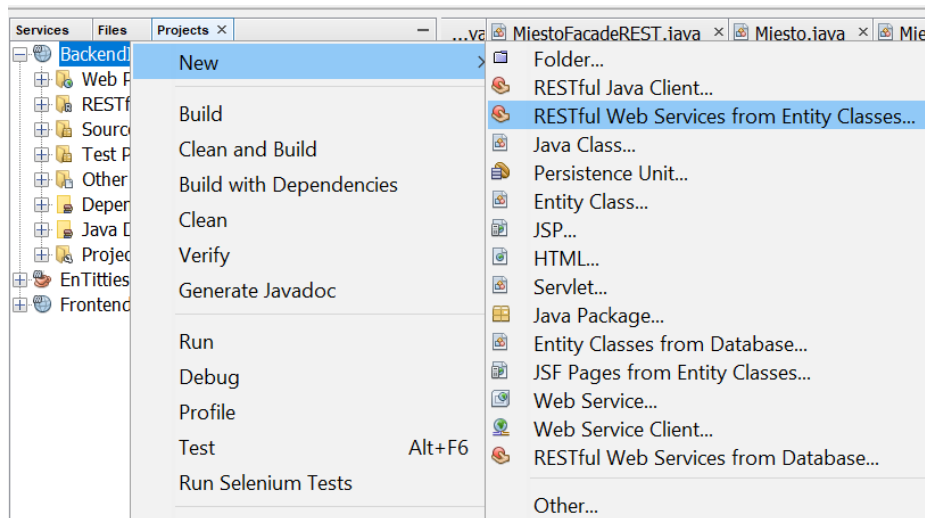
File -> New Project -> Maven -> Web Application

Pravým na Dependencies -> Add Dependency -> klikneš na Open Projects a zvolíš projekt s entitnými triedami

Pravým na projekt -> New -> New Persistence Unit -> zvolíš jdbc resource z predošlého kroku a zaklikneš Drop And Create strategy (ak máš v databáze pripravené dáta tak iba Create)



Pravým na projekt -> New -> zvolíš RESTful Web Services from Entity Classes (pre každú entitu vytvoríš rest service)



Vo vygenerovaných metódach, hlavne findAll(), možno budeš potrebovať použiť obalové triedy, ktoré si v predošlých krokoch vytvoril. Nezabudni odstrániť anotáciu @Override a zmeniť názov metódy. Môžeš sa inšpirovať týmto:

```
@GET
@Produces({MediaType.APPLICATION_XML, MediaType.APPLICATION_JSON})
public MiestoBox findAllMiesta() {
    MiestoBox miesta = new MiestoBox();
    miesta.setMiesta(super.findAll());
    return miesta;
}
```

Pri komplikáciách skús Clean and build / vyresetovať Glassfish / zatvoriť Netbeans.

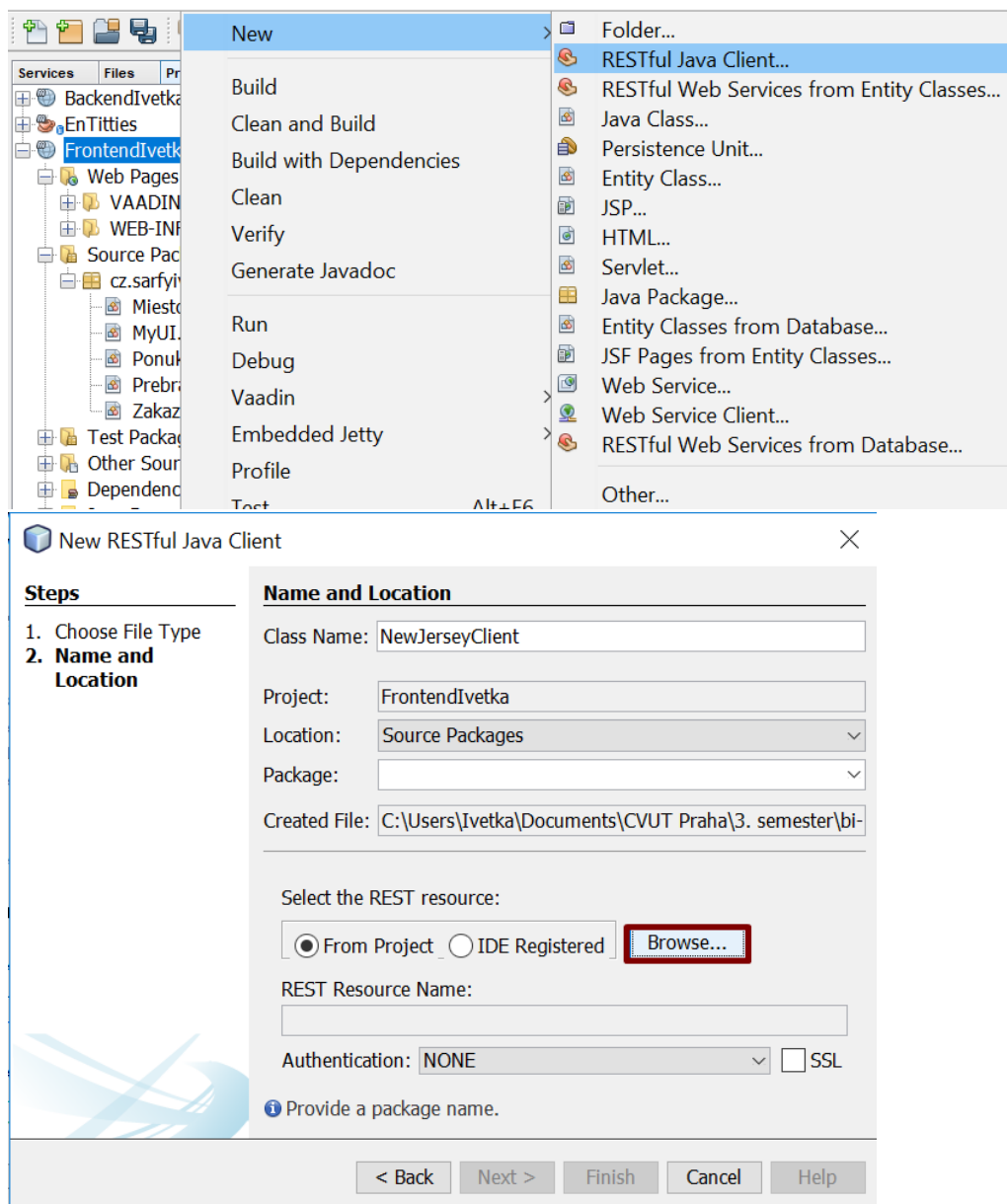
5. Vytvorenie projektu s Clientmi

Založíš nový Vaadin Web Application projekt.

File -> New Project -> Vaadin-> Vaadin Web Application Project

Vygeneruješ klientov.

Pravým na projekt -> New -> vygeneruješ RESTful Java Client pre každú restovú službu, ktorú si v predošlom kroku vytvoril (v sekcii Select the REST resource klikneš na Browse..)



Do MyUI súboru umiestniš komponenty na vytvorenie užívateľského rozhrania.

Vo vygenerovanom MyUI sa už nachádza vopred pripravený layout, do ktorého môžeš umiestňovať komponenty.

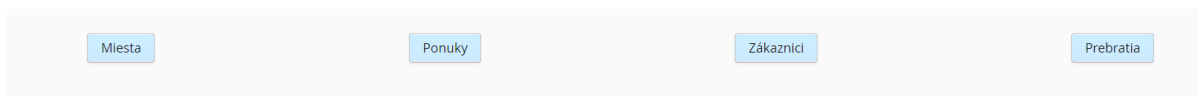
```
@Override
protected void init(VaadinRequest vaadinRequest) {

    final VerticalLayout layout = new VerticalLayout();
```

Vytvoríš ďalšie layouts, podľa toho koľko máš entitných tried.

```
GridLayout headingPonuka = new GridLayout(3, 1);
headingPonuka.setSizeFull();
headingPonuka.setVisible(false);
```

Všetky vytvorené layouts budú defaultne skryté, spolu so všetkými komponentami v nich, a užívateľ si ich bude môcť zobrazovať pomocou buttonov, ktoré sa zobrazia po otvorení stránky.



Po kliknutí na určitý button by mal užívateľ vedieť na akej obrazovke sa práve nachádza. Toto sa dá jednoducho vyriešiť nadpisom.

```
final Label h3 = new Label("Zákazníci");
h3.addStyleName(ValoTheme.LABEL_H1);
headingZakaznik.addComponent(h3, 1, 0);
headingZakaznik.setComponentAlignment(h3, Alignment.MIDDLE_CENTER);
```

Užívateľ by mal byť schopný vkladať nové údaje do tabuľky, upravovať ich a mazať.

Mazať údaje nie je vždy možné (pokiaľ existuje závislosť v inej tabuľke), a preto je nutné užívateľa vždy informovať o tom, či daný údaj v tabuľke bol úspešne zmazaný alebo ho nie je možné odstrániť. Túto informáciu môžeme napríklad vyhodiť na obrazovku v novom okne zakaždým keď sa užívateľ pokúsi vymazať ľubovoľný údaj v tabuľke.

```
Label msg;
if ( numRowsOld == numRowsUpdated ) {
    msg = new Label("Zákazníka " + myDelZakaznik.getMeno() + " " + myDelZakaznik.getPriezvisko() + " nie je možné zmazať!");
    delContentZak.addStyleName("delNot");
} else {
    msg = new Label("Zákazník " + myDelZakaznik.getMeno() + " " + myDelZakaznik.getPriezvisko() + " bol úspešne zmazaný.");
    delContentZak.addStyleName("del");
}
```

Ďalej je potrebné pridať možnosť vyhľadávania minimálne v jednej tabuľke. Môžete to urobiť napríklad pomocou ComboBoxu (výber zo všetkých rôznych krstných mien, po vybratí nám ostane v tabuľke napr. všetky Jany) alebo klasického vyhľadávania.

A search interface consisting of a text input field containing the text 'v dome' and a blue button to its right labeled 'Filtruj podľa popisu'. Below this is a table with a single row containing the text 'Popis miesta' and another row containing the text 'v dome'.

Veľa šťastia!