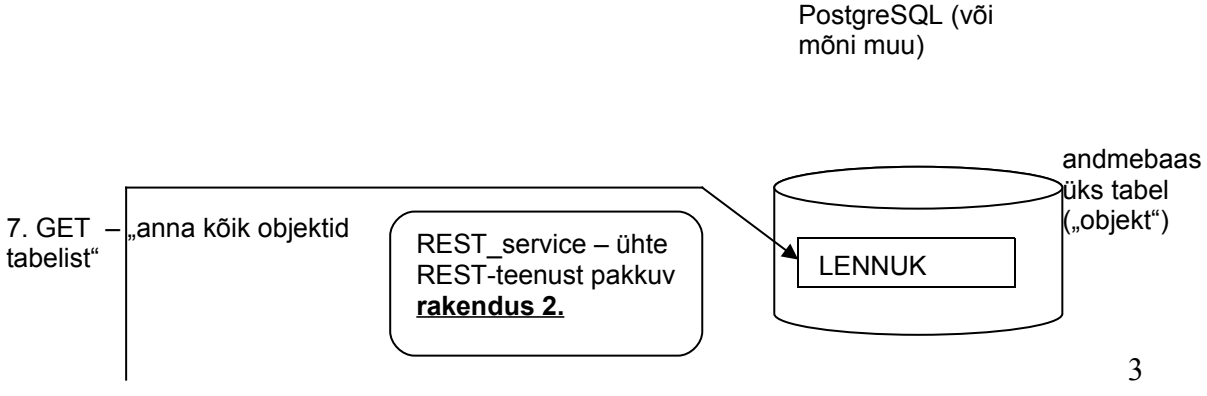
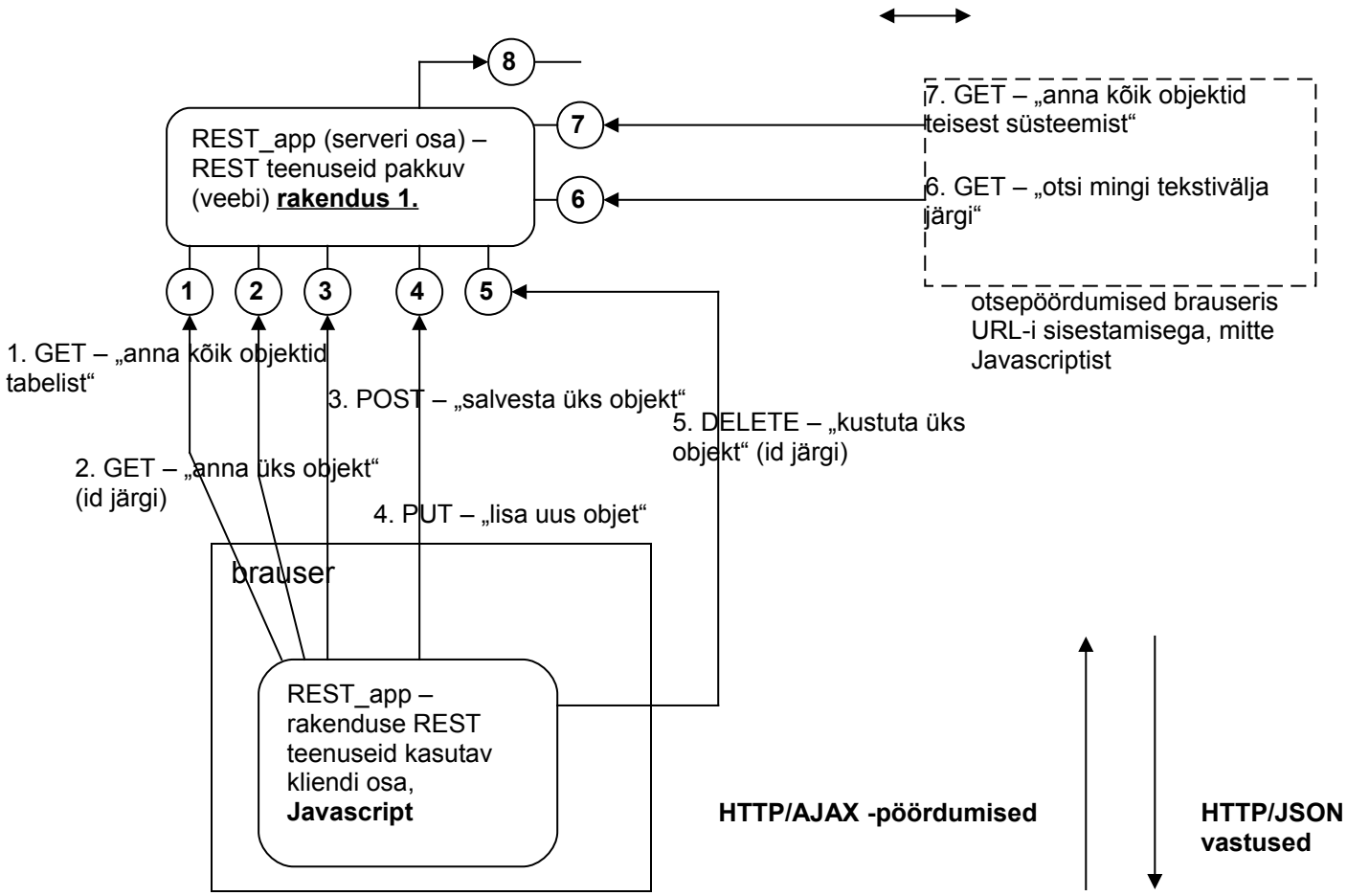
IDU0080
Ülesanne „REST teenused“
viimati muudetud: 15.02.2018

1. Ülesande olemus, nõutud funktsionaalsus





Nõutud funktsionaalsus:

1) Tuleks teha andmebaasi serverisse üks andmebaas milles on üks tabel. Tabelis peab olema vähemal 4 välja (primaarvõtmeväljaks unikaalne id; kaks tekstitüüpi andmevälja, üks numbri tüüpi andmeväli). Tabeli nimi ja väljade nimed ei tohi kokku langeda selle ülesandega seotud näite andmebaasi tabeli nime ja väljade nimedega. Andmebaasi serverina tuleb kasutada kas Eclipse'i enda JavaDB serverit, või seadistada oma arvutisse MySQL või PostgreSQL või mingi muu andmebaasi server millele on ligipääs jdbc kaudu.

2) Tuleks teha **ühte REST-teenust pakkuv rakendus** mis on ühendatud 1. punktis tehtud andmebaasi serveriga ja annab vastuseks kõik andmebaasi objektid JSON formaadis, selle teenuse poole tuleb pöörduda HTTP GET-päringuga.

Teenus peab olema testitav/käivitatav brauseriga.

Näiteks nii:

http://localhost:8080/REST_service/service/cars

3) Tuleb teha teine rakendus mis sisaldab ühte veebilehte , veebilehel käivituvat Javascripti koodi ja 6 veebteenust serveril. Neid veeb(REST)teenuseid kutsutakse välja Javascripti poolelt ja need peavad olema järgmised.

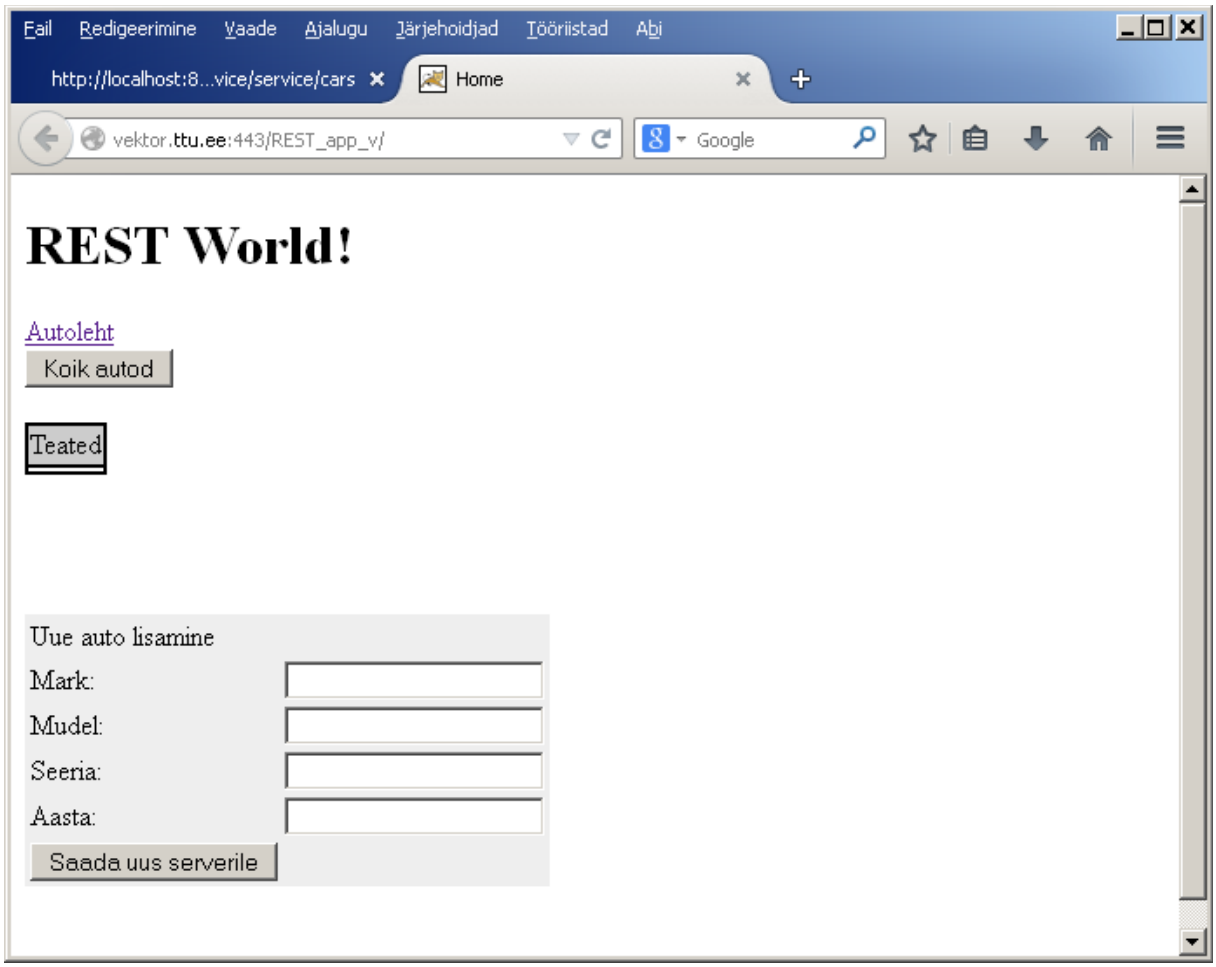
REST teenuste aadressid (URL-id) ei ole ette kirjutatud, need peab ise paika panema.

HTTP meetod/teenuse lühinimetus	mida teeb
1. GET / „anna kõik“	annab kõik objektid andmebaasi tabelist
2. GET / “anna üks“	annab ühe objekti , id järgi (id – kirje andmebaasi id)
3. POST / „salvesta muutused“	salvestab objekti muudatused (objekti andmeid muudab kasutaja ekraanivormil, nende salevstamsieks tuleb vääja kutsuda see veebteenus)
4. PUT / „lisa uus“	lisab uue objekti andmebaasi tabelisse
5. DELETE /“kustuta objekt“	kustutab objekti (kirje) andebaasi tabelist, etteantud id järgi
6. GET / „otsi mingi tekstivälja järgi“	otsib objektide hulgast mingi tekstivälja alguse järgi, LIKE ’ab%’ tüüpi päring.
7. GET / „anna kõik välisest süsteemist“	annab kõik objektid , ei võta objekte otse andmebaasist vaid pöördub **rakendus 2.** REST-teenuse poole mis võtab andmed andmebaasist. **rakendus 2** on siin nagu vahelüli **veebirakenduse** ja andmebaasi vahel

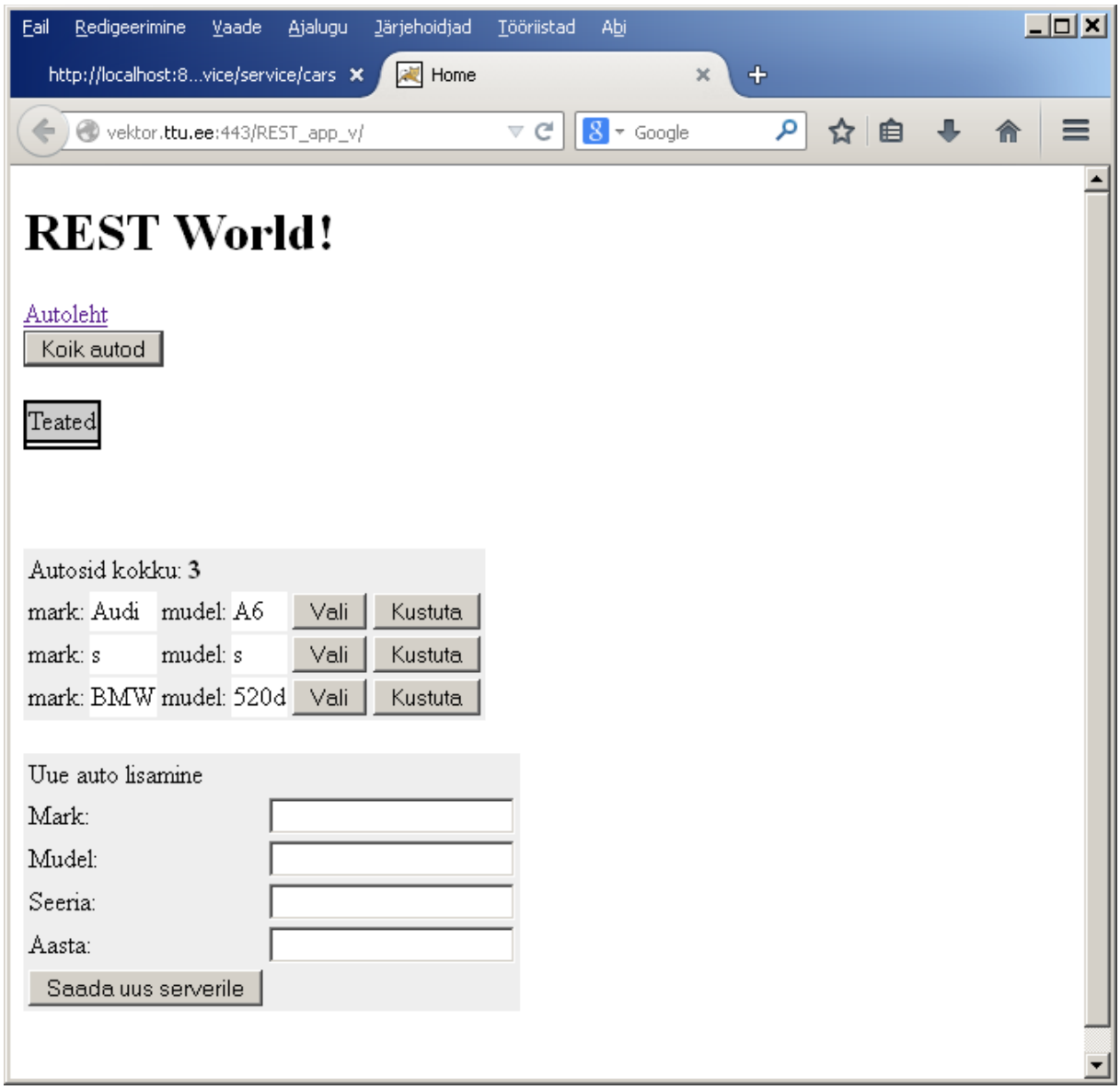
Ülalnimetatud meetodeid (välja arvatud viimast 6. GET „anna kõik välisest süsteemis“) tuleb kasutada Javascriptis kirjutatud kasutajaliideselt.

1.1. Kasutajaliidese toimimine.

Kasutajaliides peaks töötama üldiselt nii nagu pildil näha. Veebilehe disain on muidugi vaba, ei pea olema nii nagu allpool toodud, st. antud ülesande juures on rõhuasetus REST teenustel, kasutajaliides võimaldab REST teenuste toimimist kontrollida.



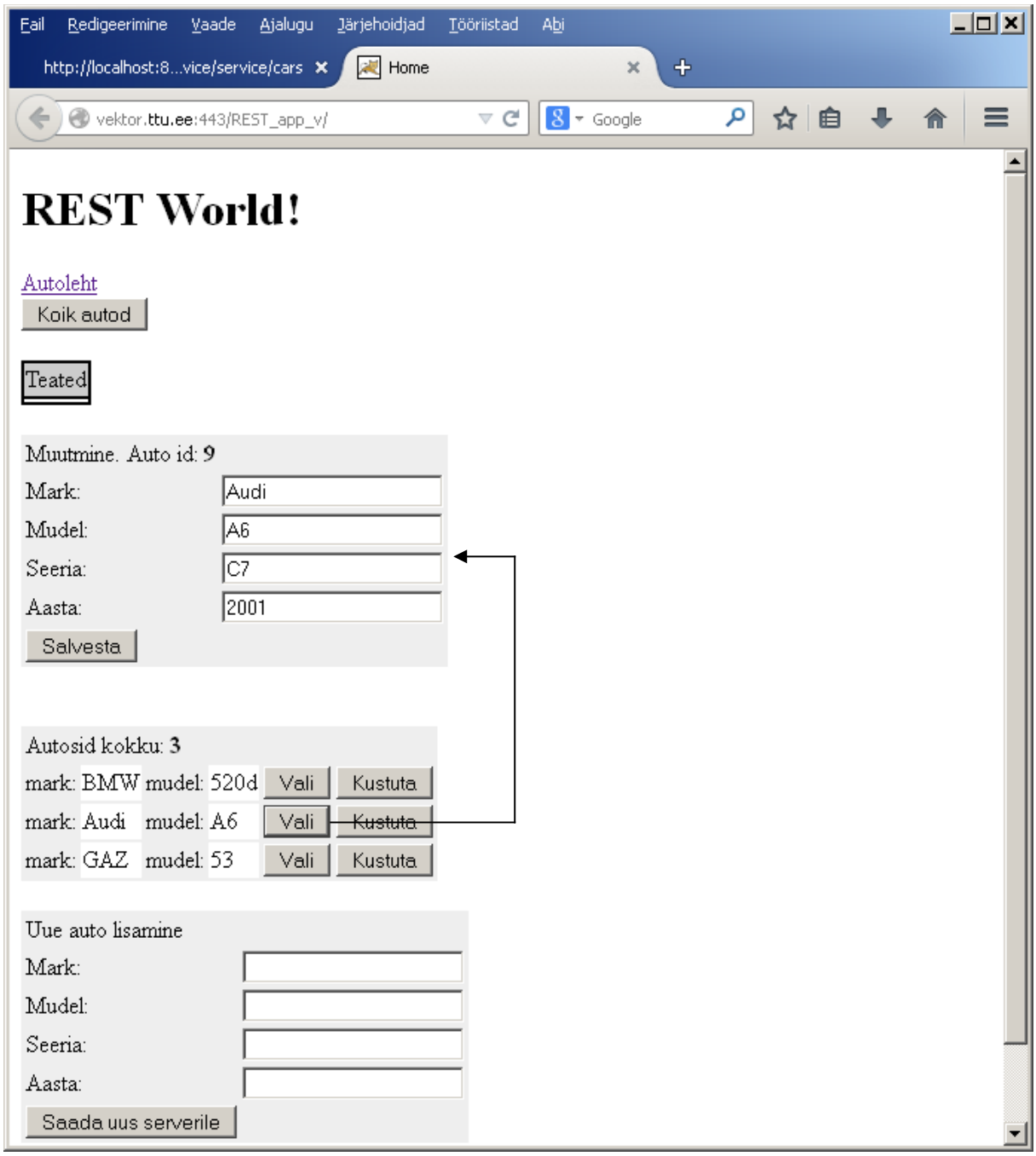
1. Nupp „Kõik <objektid>“ toob ekraanile (javascripti AJAX-väljakutse teenusele **1. GET / „anna kõik“**) kõik andmebaasi objektid.



2. Ekraanil tabelina (ridadena) näidatate objektide hulgast saab nüüd sobiva ära valida või kustutada.

Valik (nupp „Vali“) – pöördumine **2. GET / “anna üks“**

Tulemust näidatakse eraldi (alam-)vormis (kas sama lehe peal või teise lehe peal, see pole tähtis).



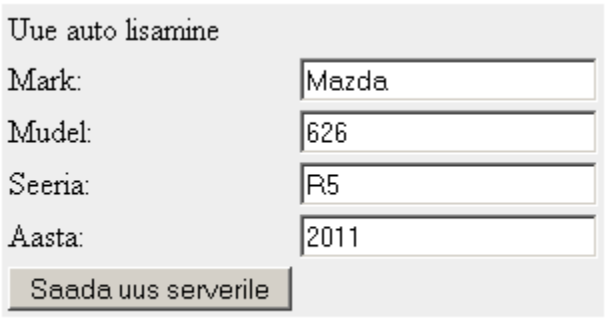
Nupp „Kustuta“ kustutab objekti andmebaasist – pöördumine **5. DELETE /“kustuta objekt“**

Väljavalitud objekti näitamise vormil peab olema nupp „Salvesta“. Sellele vajutamisel pöördumine **3. POST / „salvesta muutused“.**

Salvestamise tulemusena ei pea muutuma objekti andmed muutmisvormi all olevas tabelis, seda tabelit ekraanil ei ole salvestamisel vaja uuesti laadida/uuendada.

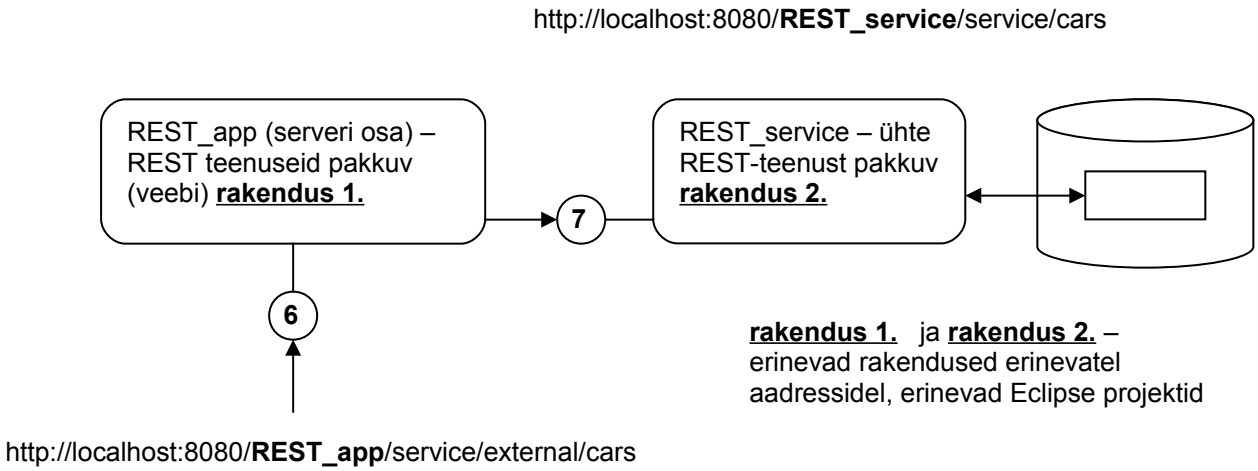
Kogu aeg võib ekraanil valmis olla uue objekti lisamise vorm (aga seda võib avada ka mõnele lingile või nupule vajutamisega kui nii tundub sobivam.

Vajutades nupule „Saada uus serverile“ saadetakse täidetud vormist võetud andmed serverile tehes pöördumise **4. PUT / „lisa uus“.**



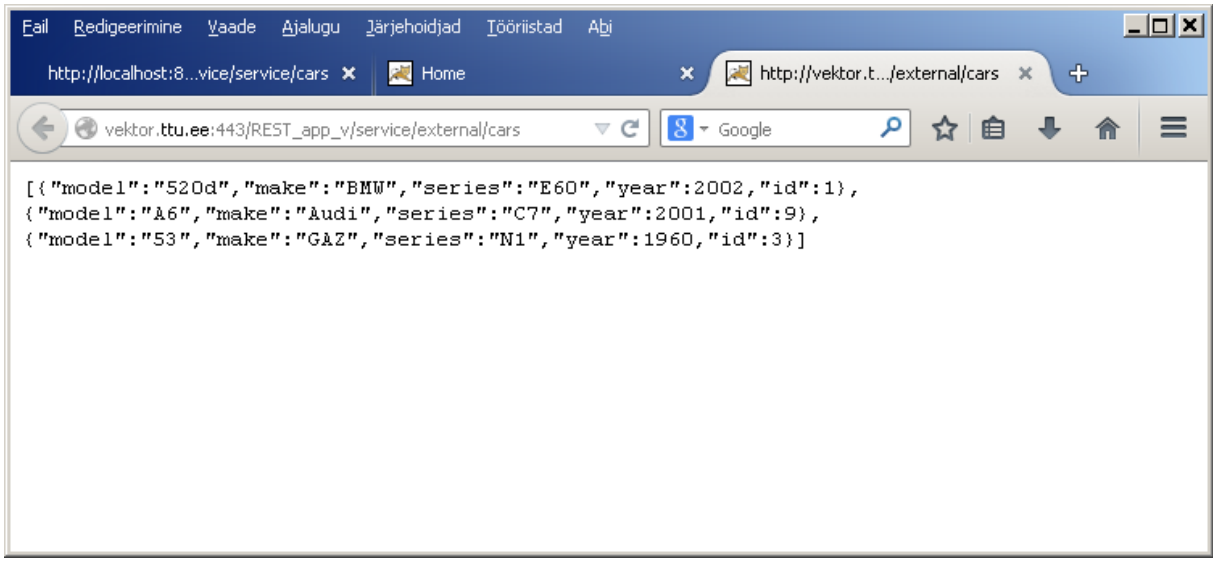
1.2. Andmed rakendusevälisest süsteemist.

Selleks et proovida kuidas käib andmevahetus erinevate infosüsteemide või infosüsteemi osade vahel (seal kasutatakse ka REST-teenuseid, see on REST-teenuste teine kasutamisevaldkond) tuleks pöördumine **6. GET / „anna kõik välisest süsteemist“** realiseerida mõnevõrra teisiti kui teised teenused.



Kui tuleb serverile pöördumine **6. GET / „anna kõik välisest süsteemist“** siis ei pöördu rakendus andmebaasi poole vaid pöördub rakenduse 2. poole mis sisaldab ainult ühte teenust (sama teenus mis on ka rakenduses1. - **1. GET / „anna kõik**“) ja saab objektid selle teenuse käest.Need „välisteenuse“ käest saadud andmed saadab teenus 6. siis kliendile (brauserile) tagasi.

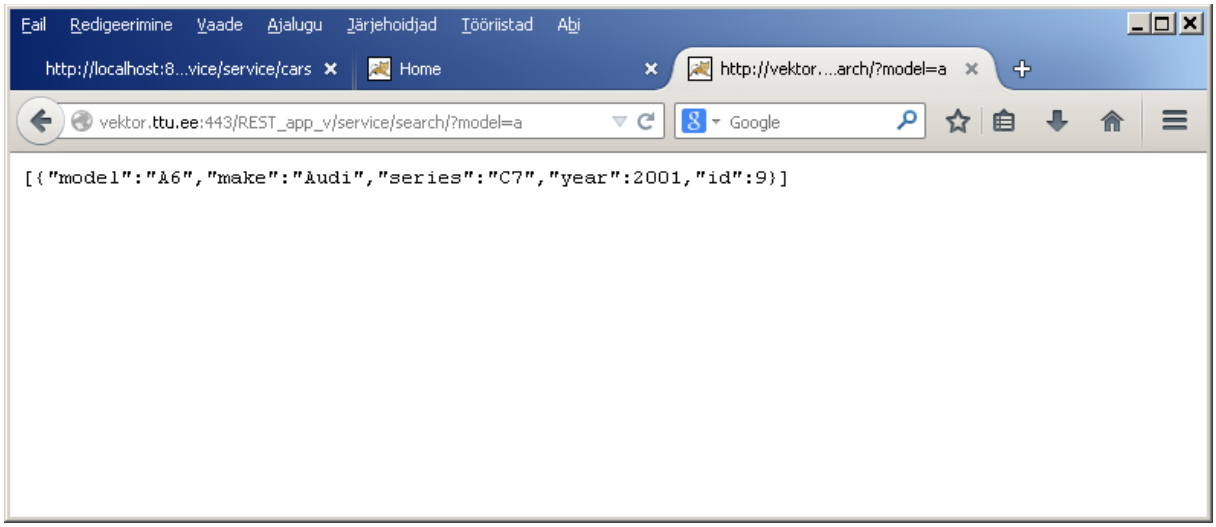
Seda meetodit (6.) ei pea kasutama veebiliideses, seda ei pea javascriptis AJAX-iga välja kutsuma. Kuna tegemist on GET-pöördumisega siis võib seda testida otse brauserist, nagu näiteks nii:



1.3. Otsingumeetod.

Otsingu REST-teenust ei ole samuti vaja kasutada javascriptist, kuna tegemist on ka GET meetodiga siis piisab kui seda saab testida brauseriga (näiteks nii):

http://vektor.ttu.ee:443/REST_app_v/service/search/?model=a



2. Töövahendite ja tehnoloogia valikust.

Need kes ei ole sarnaseid ülesandeid enne teinud on võiksid võtta aluseks selle ülesandega kaasneva näite ja seda näidet muuta ja täiendada et saada soovitud tulemus. Näide koosneb kahest Eclipse projektist „REST_service“ (rakendus2.) ja „REST_app“ (rakendus 1.) mis tuleks importida oma Eclipse alla ja seal siis edasi arendada. Näite kohta vaata lähemalt punktist 4.

Ei ole keelatud kasutada ka teisi platvorme ja programmeerimiskeeli ja andmebaasisüsteeme mis on näitest erinevad aga sellisel juhul leppige harjutustundi andva õppejõuga kokku.

NB! Kui kasutate oma valitud tehnoloogiat siis samuti nagu Eclipse kasutajad peate te näitama kaitsmisel et võtate need rakendused mida kaitsmisel näitate oma Git kontolt (ja näitate versiooni üleslaadimise ajalugu).

3. Ettevalmistused.

3.1. Andmebaasi tabel ja andmebaas.

Mõelge valmis (või valige olemasolevatest andmebaasidest) andmebaasi tabel. Selles tabelis peaks olema lisaks võtmeväljale veel vähemalt 4 andmevälja millest vähemalt üks võiks olla tekstiväli.

Tabeli nimi ja väljade nimed ei tohi kokku langeda selle ülesandega seotud näite andmebaasi tabeli nime (CAR) ja väljade nimedega. Näites kasutatud andmebaasitabel (PostgreSQL jaoks) on järgmine:

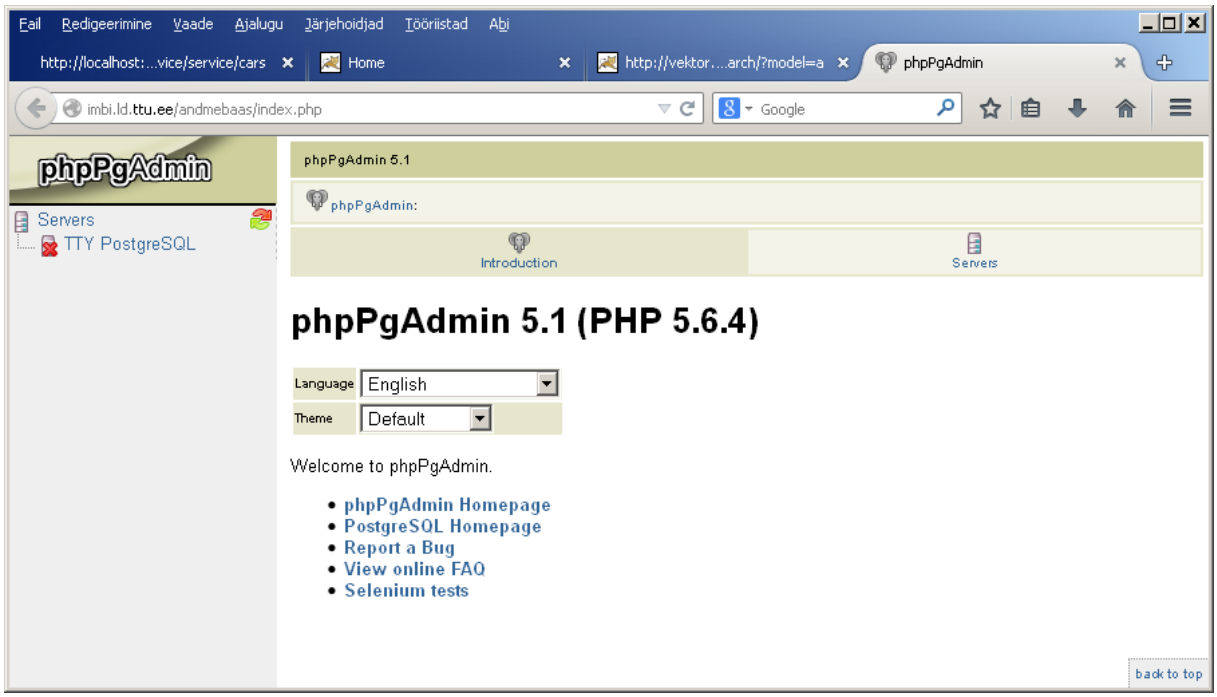
CAR	
id	numeric
make	text
model	text
series	text
year	numeric

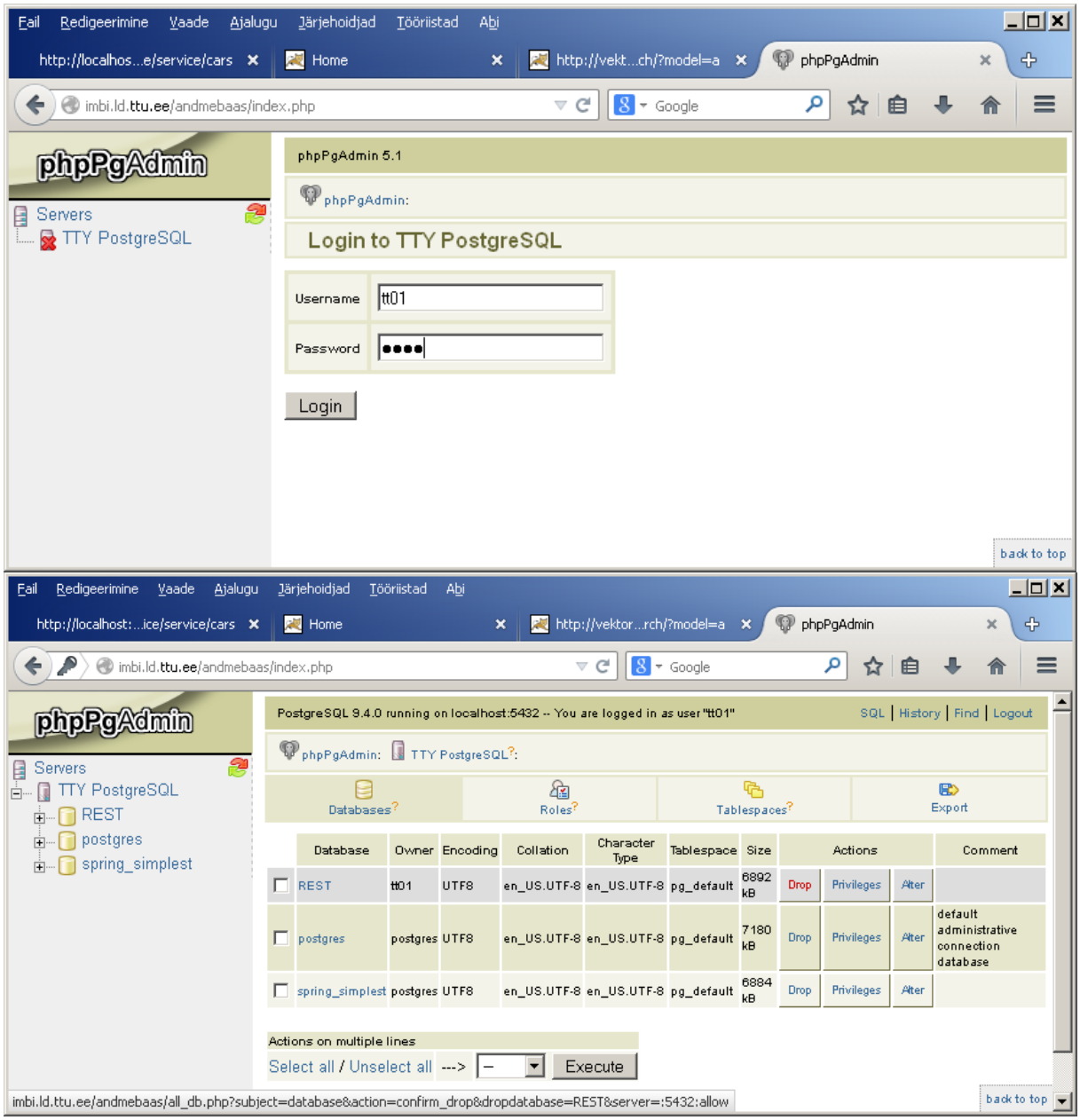
```
CREATE SEQUENCE car_id ;

CREATE TABLE car
( id numeric(10,0) NOT NULL DEFAULT nextval('car_id'),
  make text,
  model text,
  series text,
  year numeric(4,0),
  CONSTRAINT car_pk PRIMARY KEY (id)
) ;
```

3.1.1 Tehke andmebaas ja sinna teie poolt valitud struktuuriga tabel PgAdmin veebiliidese abil.

Võite kasutada Eclipse'i JavaDB või oma arvutisse installeeritud (PostgreSQL-i , MySQL-i,..) andmebaasisüsteemi. Andmebaasisüsteemi installi jooksul installige ka selle andmebaasisüsteemi jdbc driverid (MySQL korral Connector/J).

Minge aadressile (NB! kui leht annab veateate, siis kasutage PgAdmin opsüsteemi-põhist kasutajaliidest, vt. 3.1.2).





Tehke andmebaas ja sinna sisse andmebaasi tabel.

NB ! Ärge kustutage ega muutke teiste tehtud andmebaase!

Java rakendustest kasutamiseks andmebaasi JDBC aadress on imbi serveri puhul järgmine:

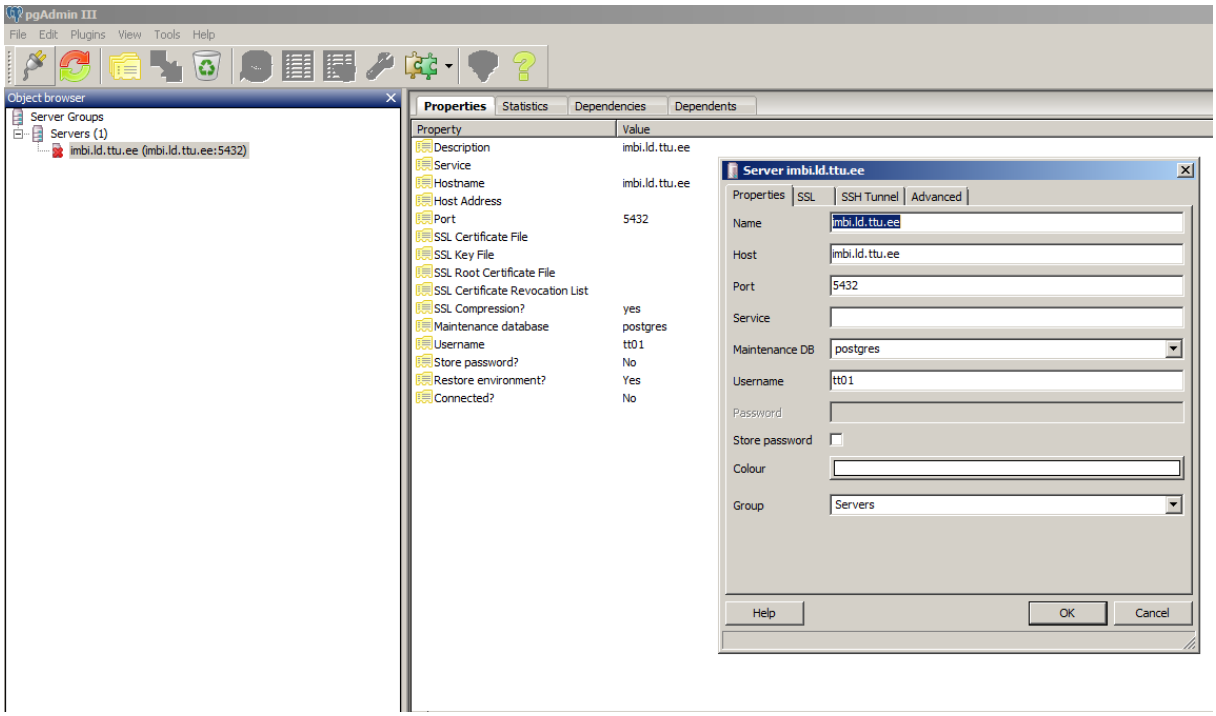
jdbc:postgresql://imbi.ld.ttu.ee/<teie andmebaasi nimi>

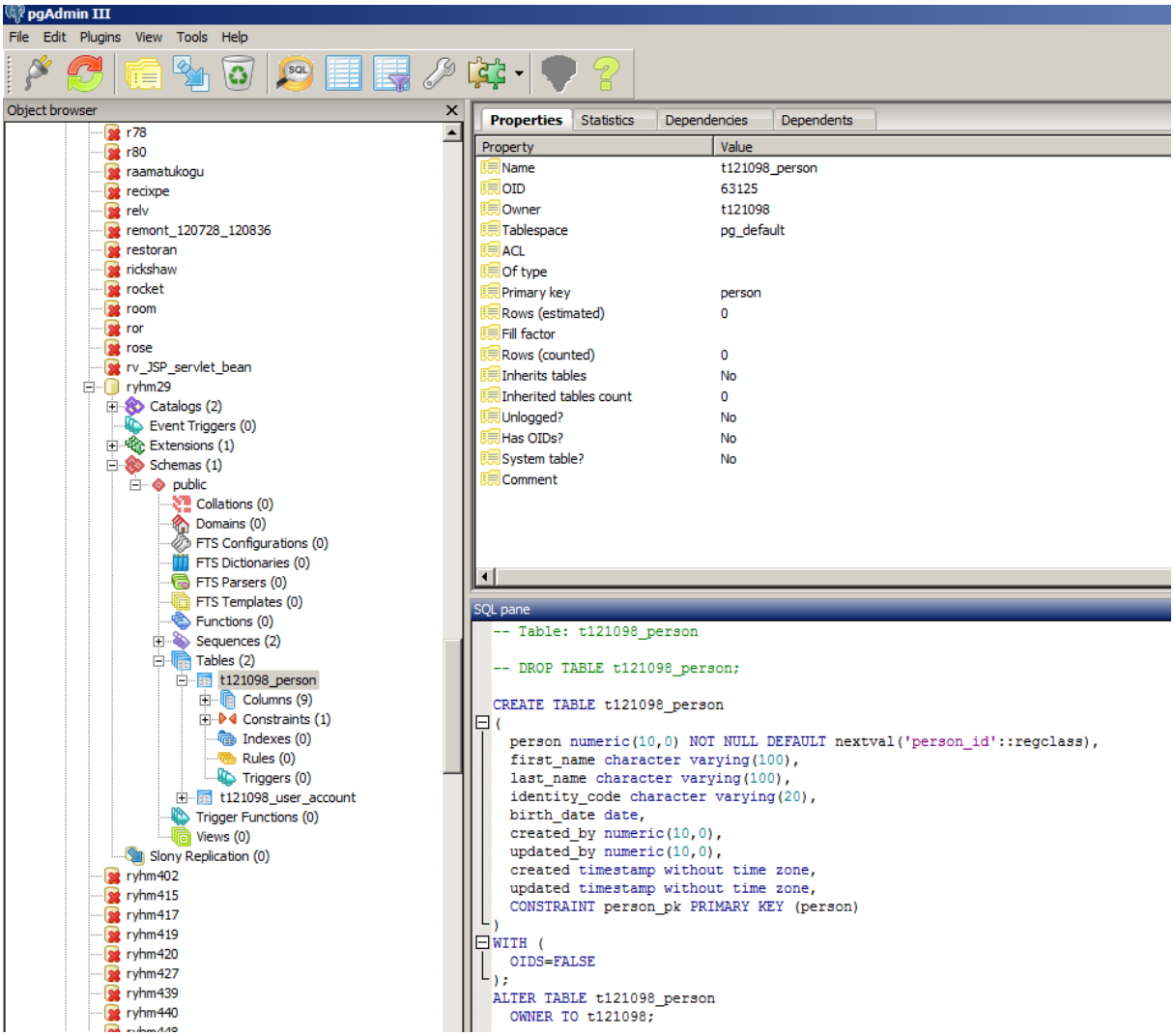
Näiteks töötavas lahenduses on andmebaasi aadressiks

jdbc:postgresql://imbi.ld.ttu.ee/REST

3.1.2 Avage andmebaasiserver PgAdmin opsüsteemi-keskse kasutajaliidese abil.

Windowsi installiga tuleb kaasa PgAdmin III programm (see on olemas ka TTÜ arvutiklassis).
Valige File menüüst Add Server, täitke ühenduse seaded.
Kasutajanimi ja parool on mõlemad **tt01**
OK.





Looge oma andmetabel ja looge oma REST rakendus (või seadistage näidisprojekt ümber) kasutama väljavalitud andmetabelit. Andmeid andmetabelis saate muuta / lisada. Kustutamise jaoks lisage ise enne üks kirje ja kustutage siis ära.

Java rakendustest kasutamiseks andmebaasi JDBC aadress on imbi serveri puhul järgmine:

jdbc:postgresql://imbi.ld.ttu.ee/<teie andmebaasi nimi>

Näiteks töötavas lahenduses (näidisprojektis) on andmebaasi aadressiks

jdbc:postgresql://imbi.ld.ttu.ee/REST

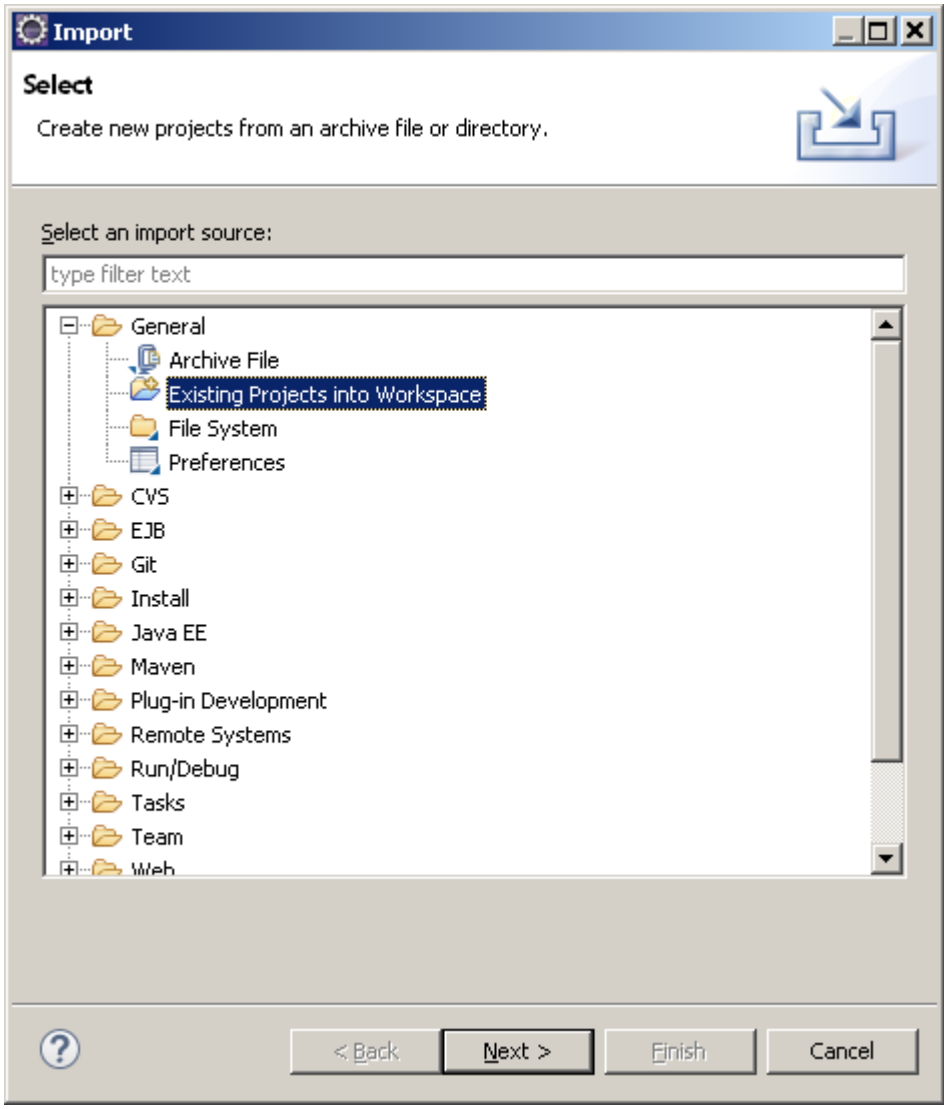
3.2. Eclipse.

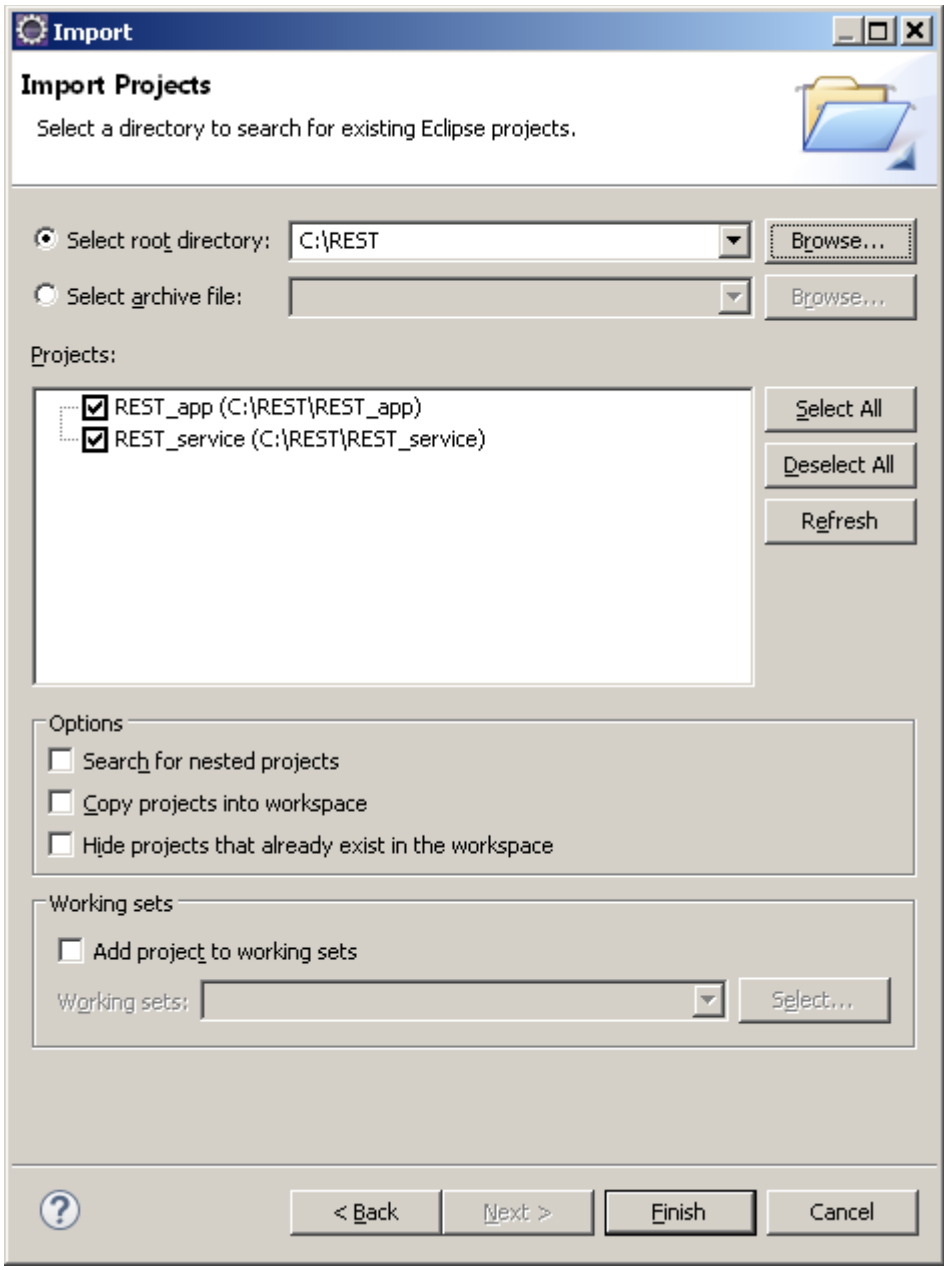
Kui kasutate Eclipse siis olge kindlad järgmises:

* Teil on Eclipse JEE versioon.

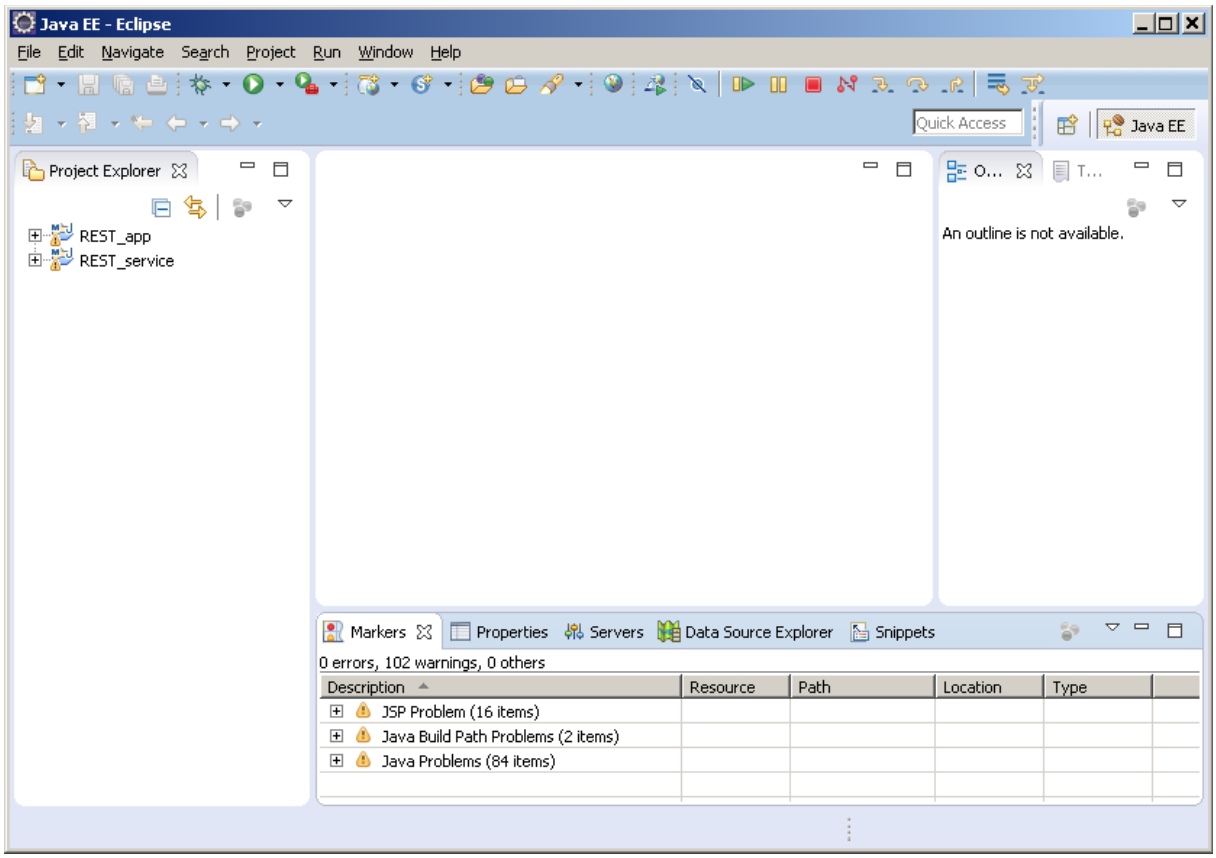
* Teie Eclipses on installeeritud Maven-i plugin (kõige uuemas Eclipses alates „Luna“ on see vaikimisi olemas).

3.3. Näiteprojektide import.





Peale importimist läheb mõned minutid aega kuni Maven tõmba teie Eclipse alla nende projektide jaoks vajalikud Java teegid.



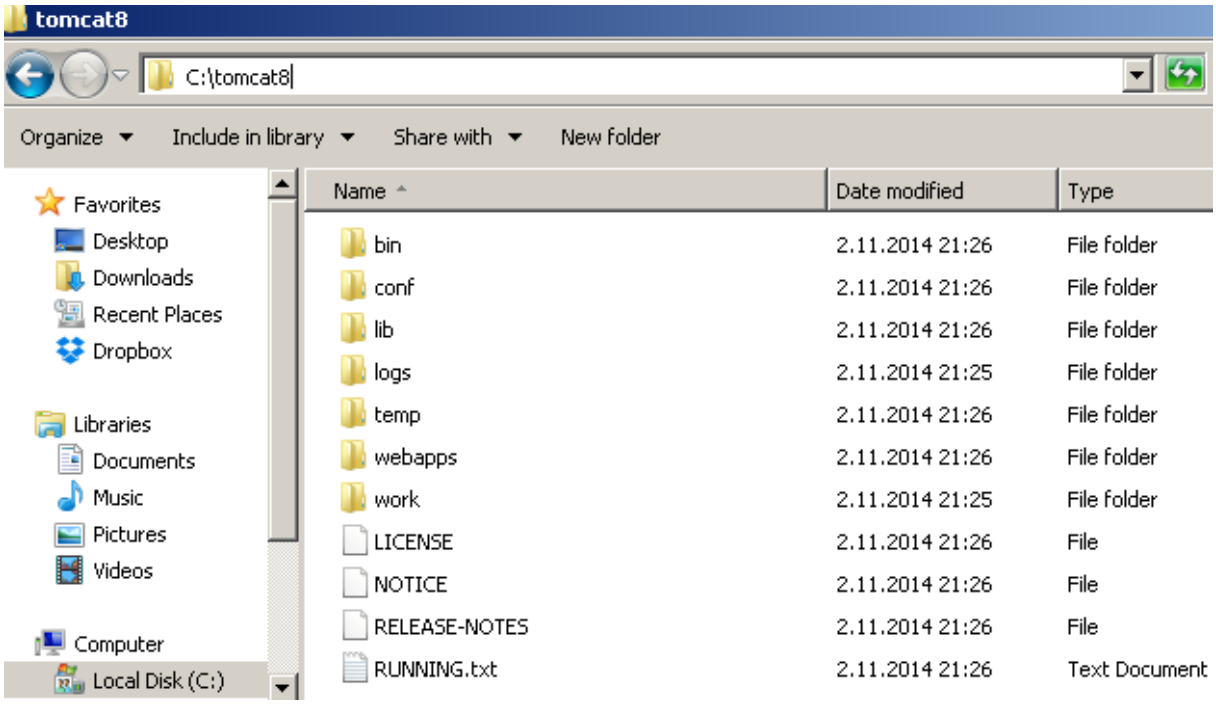
3.4. Tomcati serveri seadistamine.

Tõmmake endale arvutisse Apache Tomcat-i zip-versioon ja pakkige kuhugi lahti.

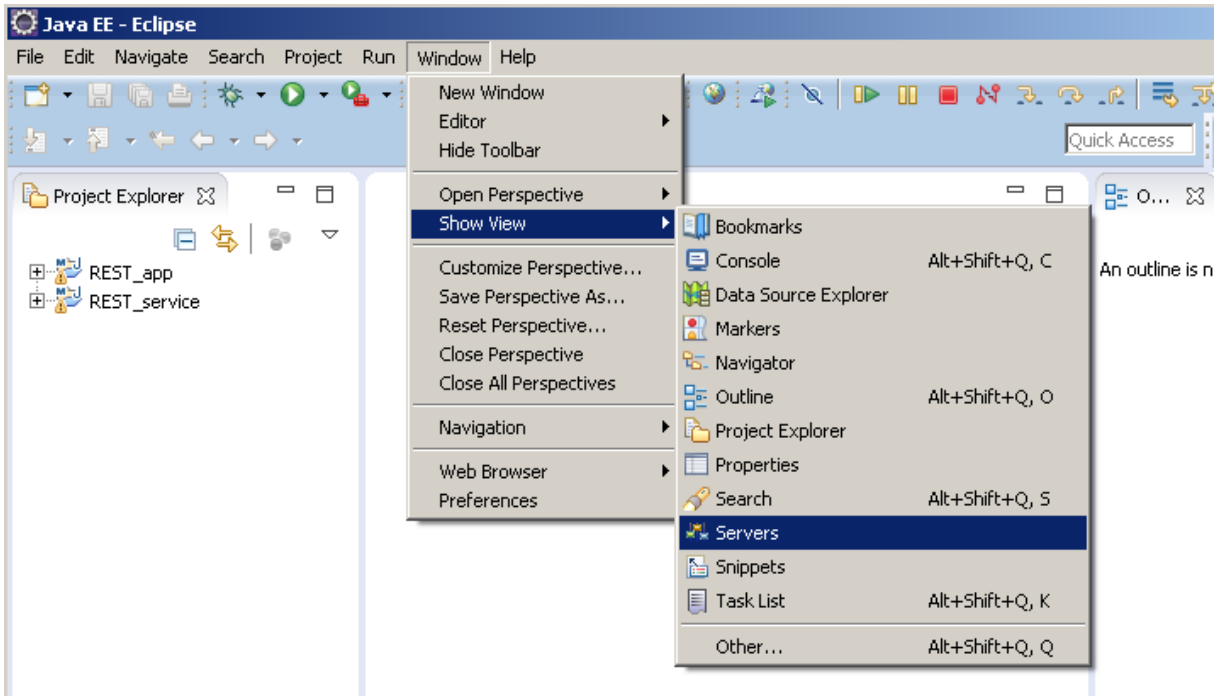
http://tomcat.apache.org/

Näiteks 64-bit Tomcat 8:
http://www.eu.apache.org/dist/tomcat/tomcat-8/v8.0.32/bin/apache-tomcat-8.0.32-windows-x64.zip

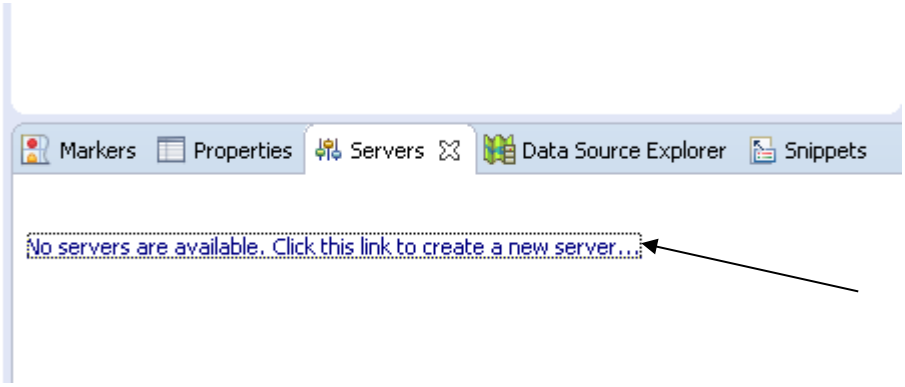
Pakkige lahti näiteks kataloogi C:\tomcat8

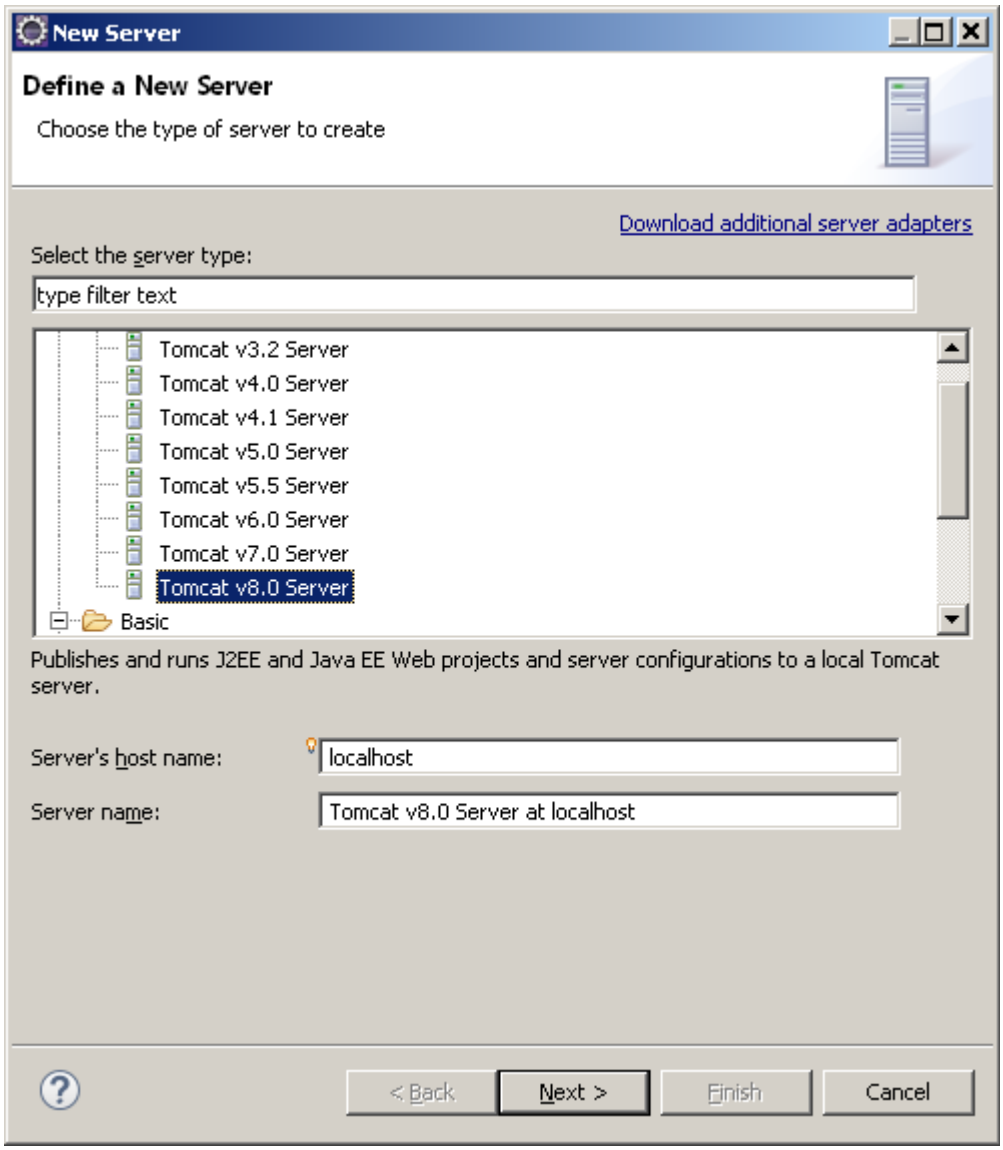


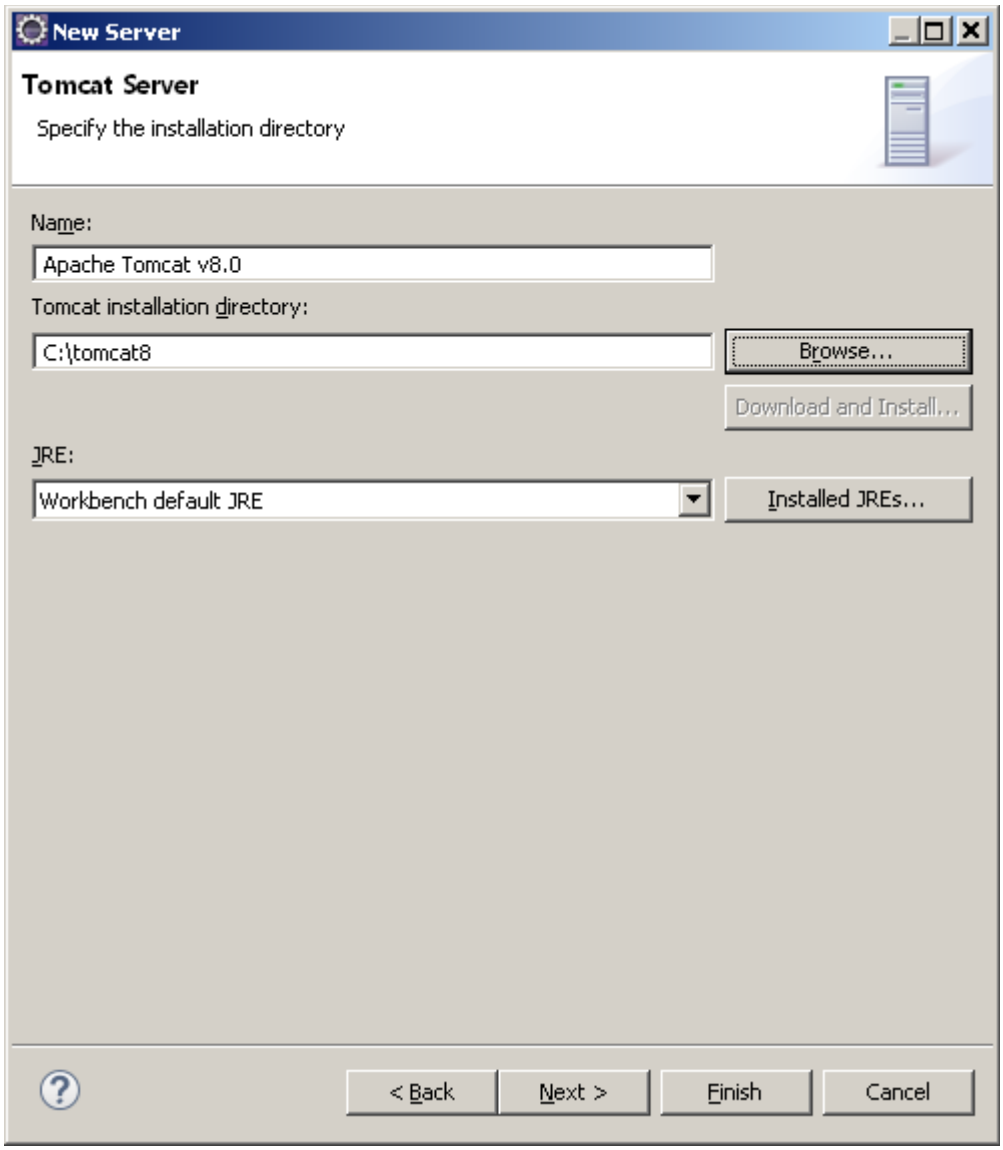
Nüüd võtke „Show view“ -> „Servers“.



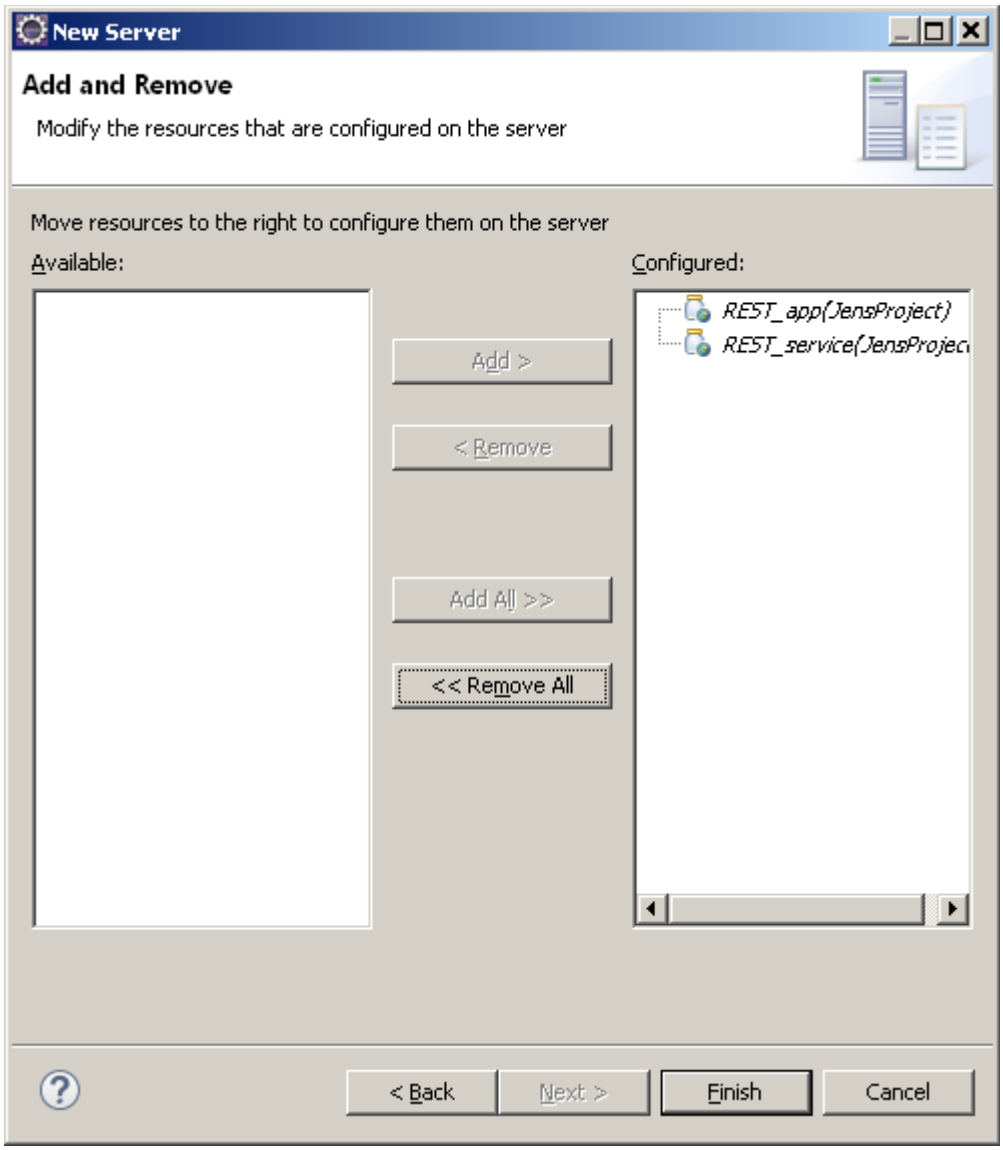
Ja defineerige Eclipse all Tomcat-i server


Markers
Properties
Servers
Data Source Explorer
Snippets
No servers are available. Click this link to create a new server...

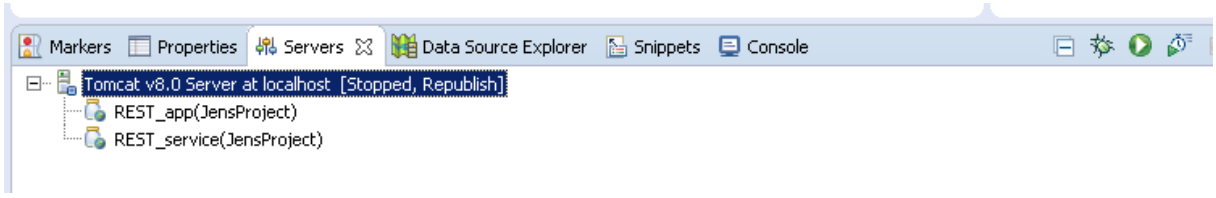

New Server
Define a New Server
Choose the type of server to create
Download additional server adapters
Select the server type:
type filter text
Tomcat v3.2 Server
Tomcat v4.0 Server
Tomcat v4.1 Server
Tomcat v5.0 Server
Tomcat v5.5 Server
Tomcat v6.0 Server
Tomcat v7.0 Server
Tomcat v8.0 Server
Basic
Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.
Server's host name:
localhost
Server name:
Tomcat v8.0 Server at localhost
< Back
Next >
Finish
Cancel

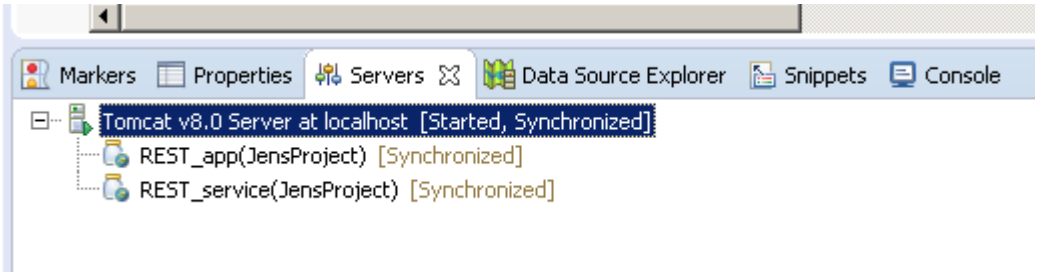


Kui näiteprojektid on eelnevalt Eclipse alla juba imporditud siis võib need ka kohe serverile panna. Või siis hiljem projekti päise peal hüpikmenüüst Run as – Run on server.



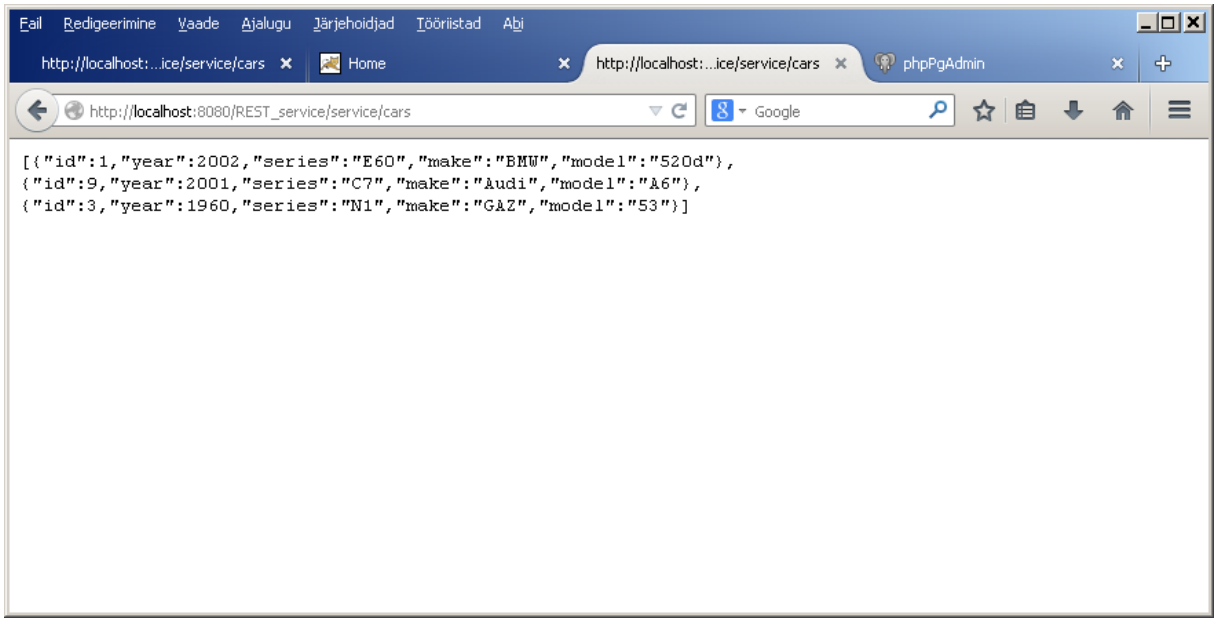
Käivitage server.





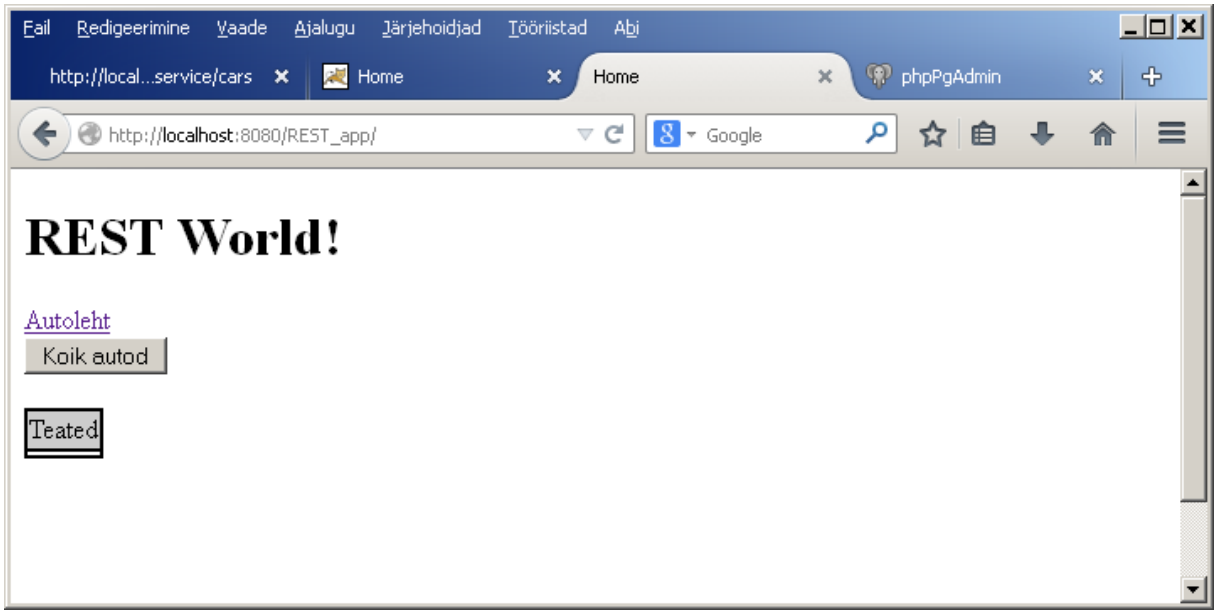
Proovige kõigepealt kas ühe REST-teenusega rakendus ((rakendus nimega „REST_service“) töötab.

http://localhost:8080/REST_service/service/cars

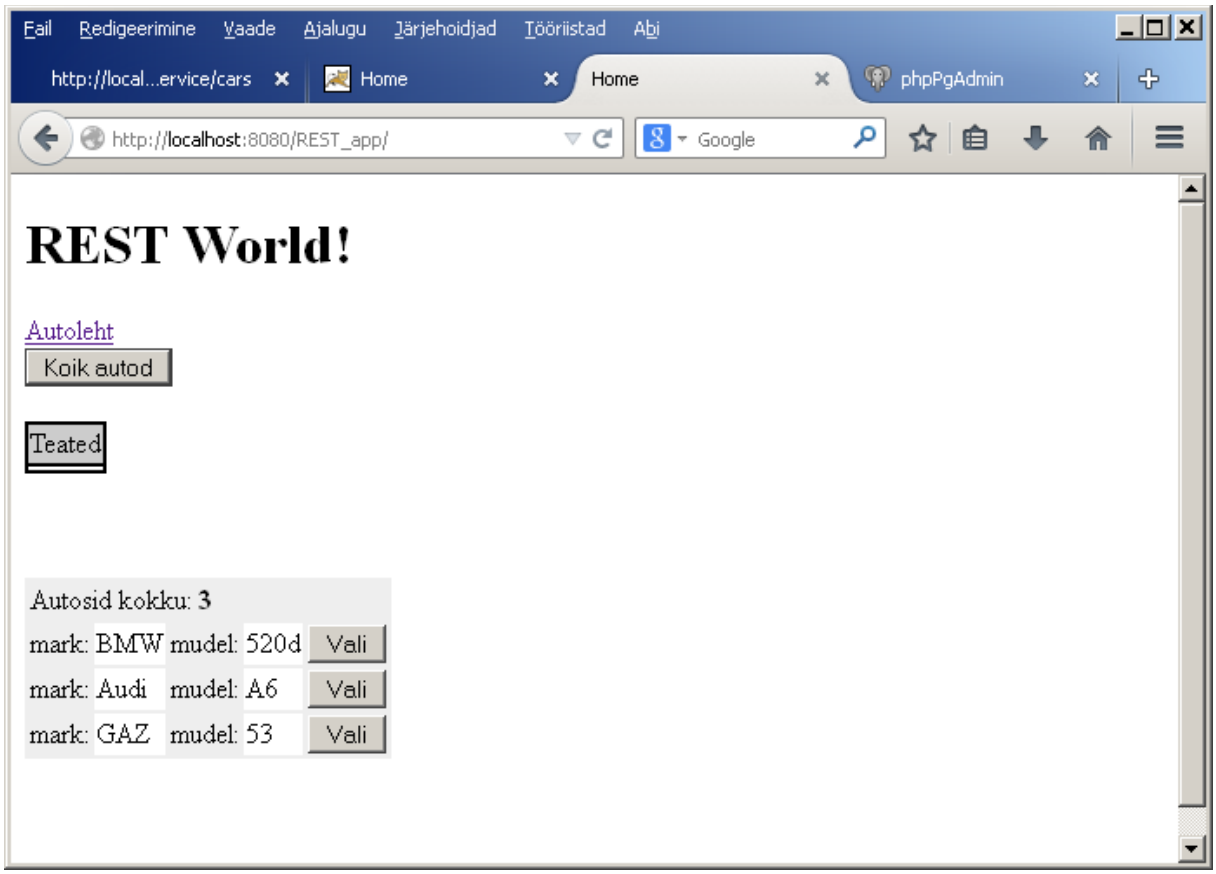


Proovige kas veebirakendus töötab (rakendus nimega „REST_app“).

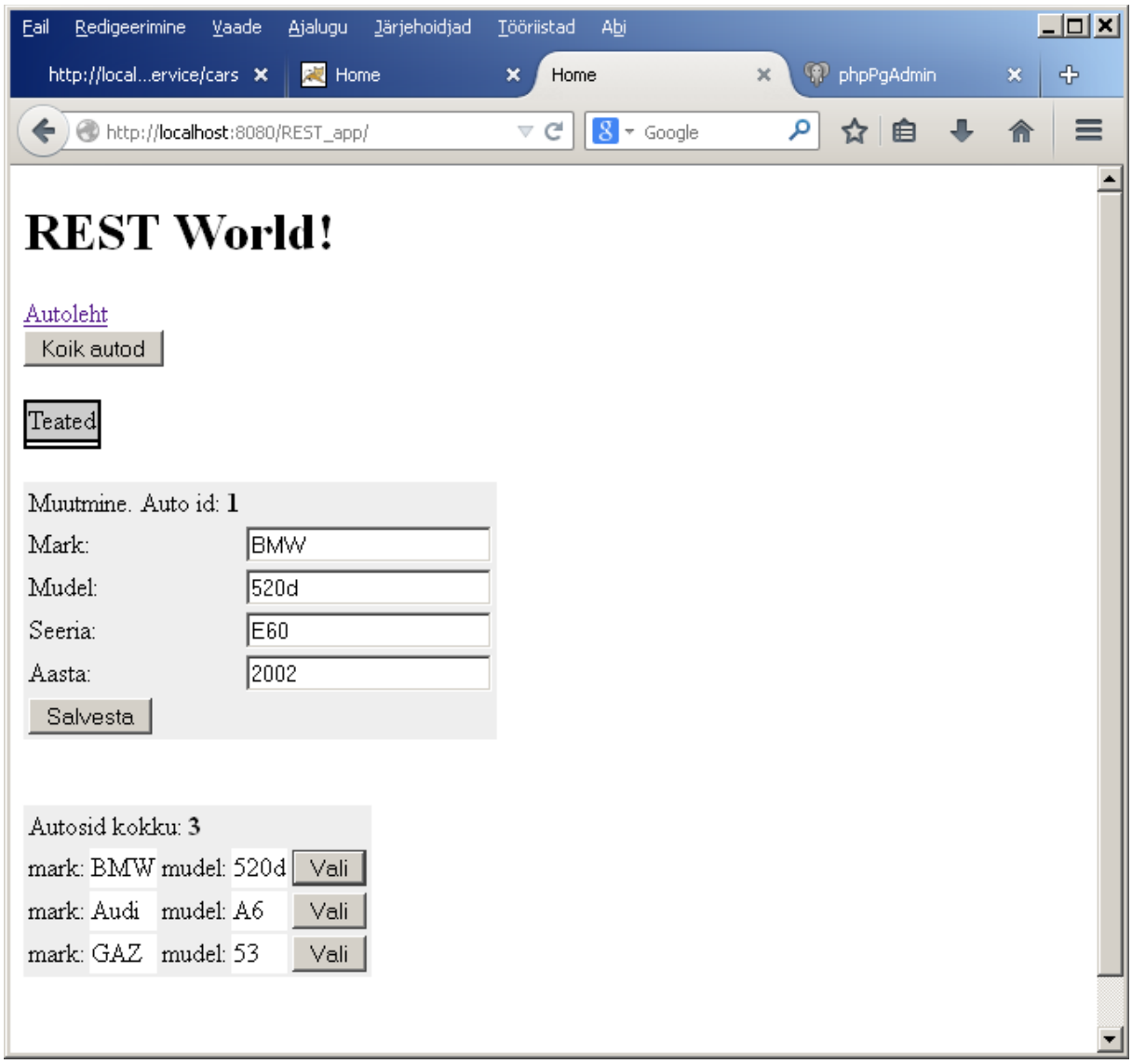
http://localhost:8080/REST_app/



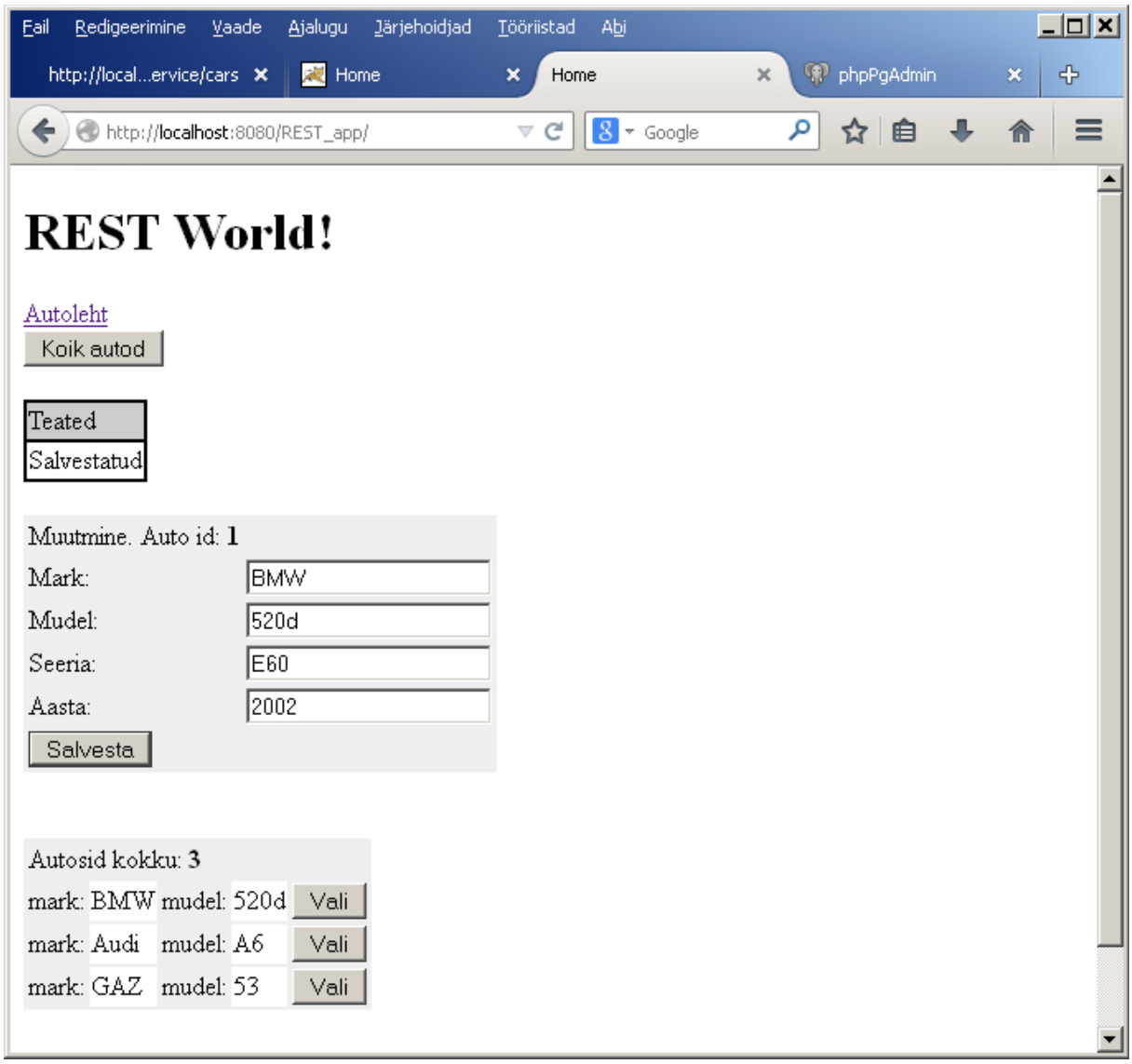
Vajutage nupule „Kõik autod“.


Fail Redigeerimine Vaade Ajalugu Järjehoidjad Tööriistad Abi
http://local...ervice/cars
Home
Home
phpPgAdmin
http://localhost:8080/REST_app/
Google
REST World!
Autoleht
Koik autod
Teated
Autosid kokku: 3
mark: BMW mudel: 520d Vali
mark: Audi mudel: A6 Vali
mark: GAZ mudel: 53 Vali

Valige auto.

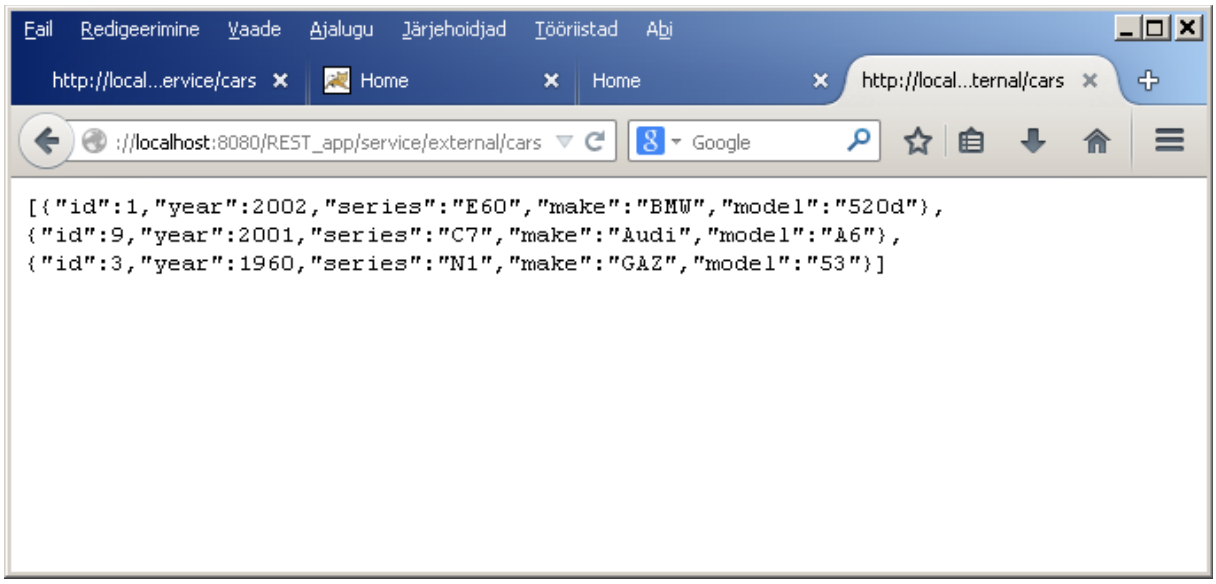


Ka salvestus peaks töötama.



Vaadake kas toimib kahe rakenduse vaheline andmevahetus.

http://localhost:8080/REST_app/service/external/cars



Mida siis teha et sellest näitest saaks ülesandes soovitud lahendus:

1. Näide tuleb panna tööle oma andmebaasi tabeli peal.
2. Näites puuduvad osad meetodid ja nende kasutamine veebirakenduses (see on näha ka kui vaatate näite veebiliidest) – need tuleb ise juurde teha.

HTTP meetod/teenuse lühinimetus	kas on näites
1. GET / „anna kõik“	olemas
2. GET / “anna üks“	olemas
3. POST / „salvesta muutused“	olemas
4. PUT / „lisa uus“	PUUDU
5. DELETE /“kustuta objekt“	PUUDU
6. GET / „otsi mingi tekstivälja järgi“	PUUDU
7. GET / „anna kõik välisest süsteemist“	olemas

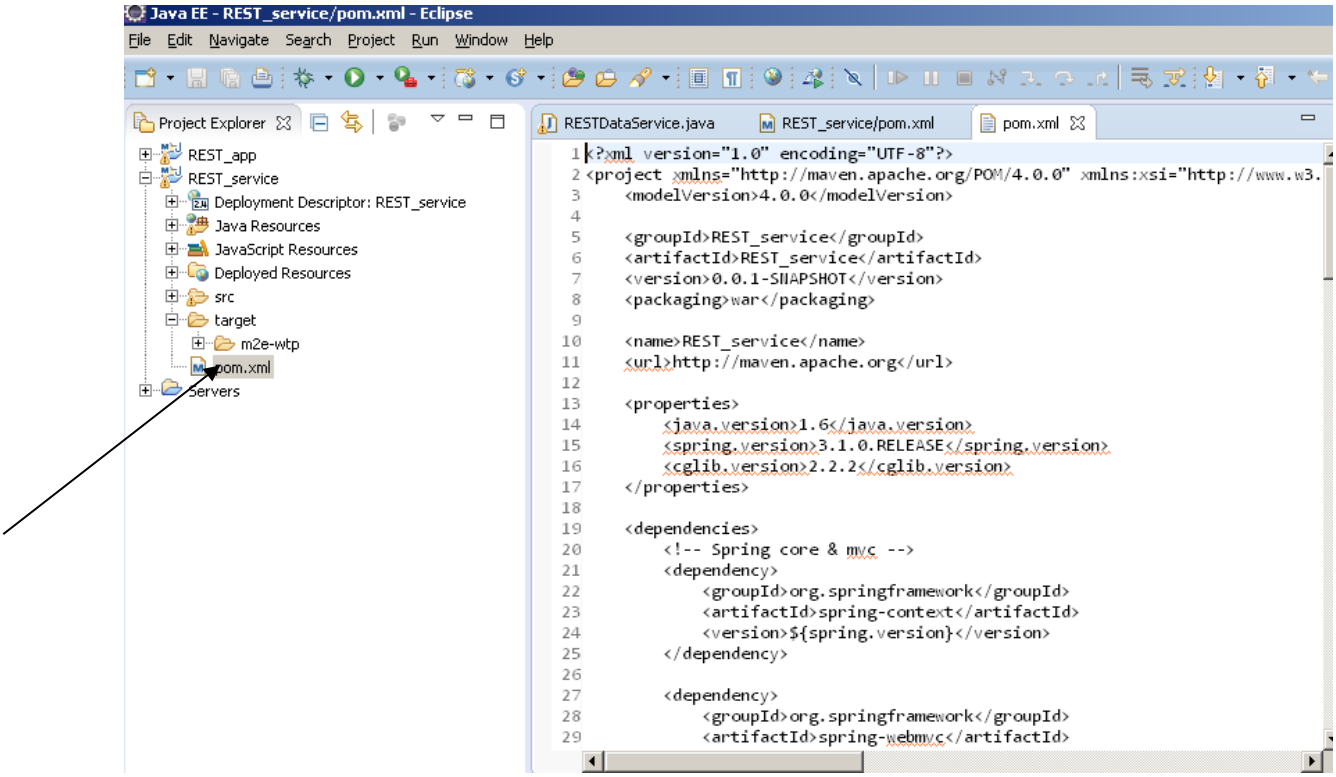
4. Näide.

4.1. Näites kasutatud tehnoloogiad.Kuidas see näide on Eclipses loodud.

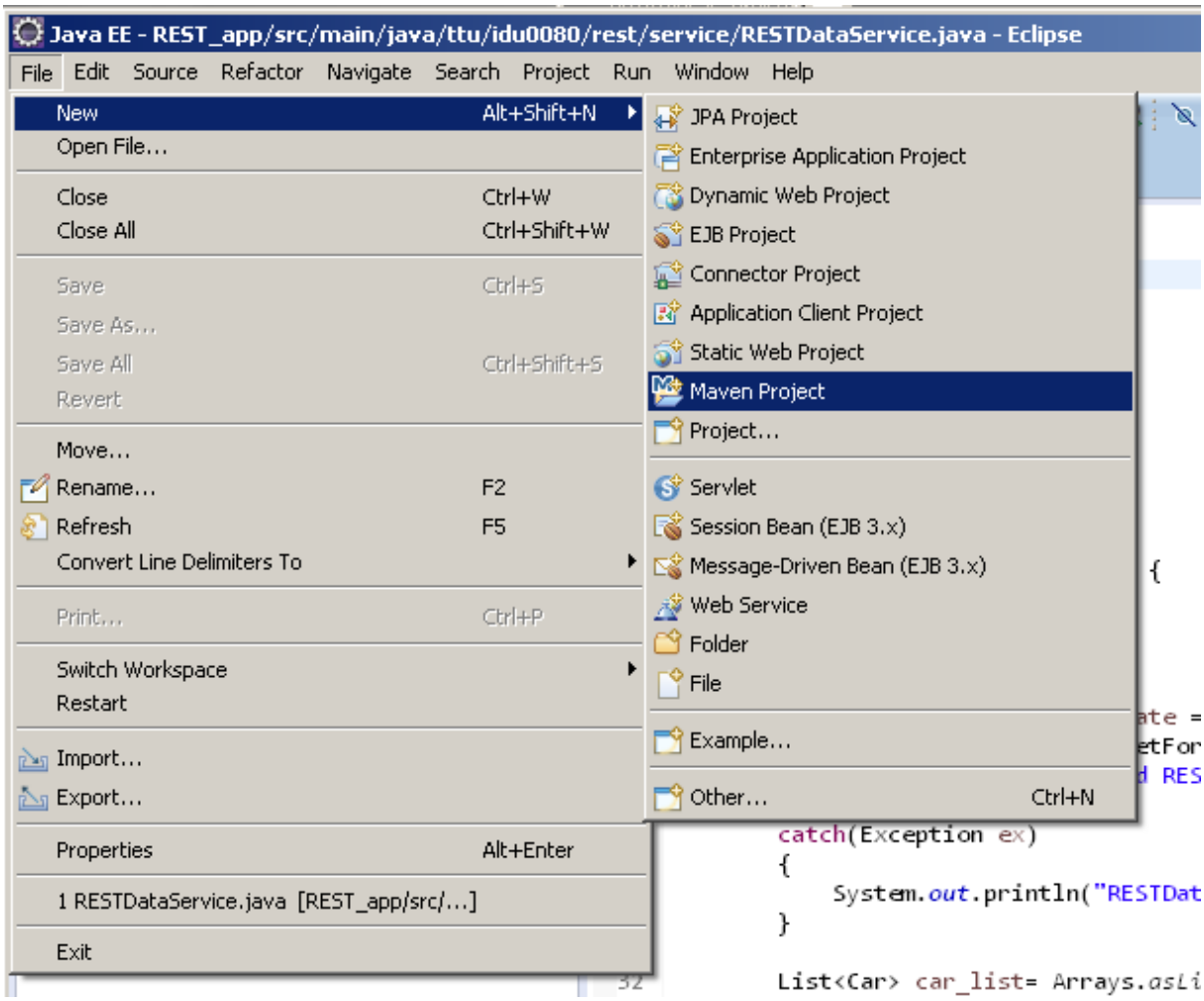
Näites on kasutatud järgmisi tehnoloogiaid.

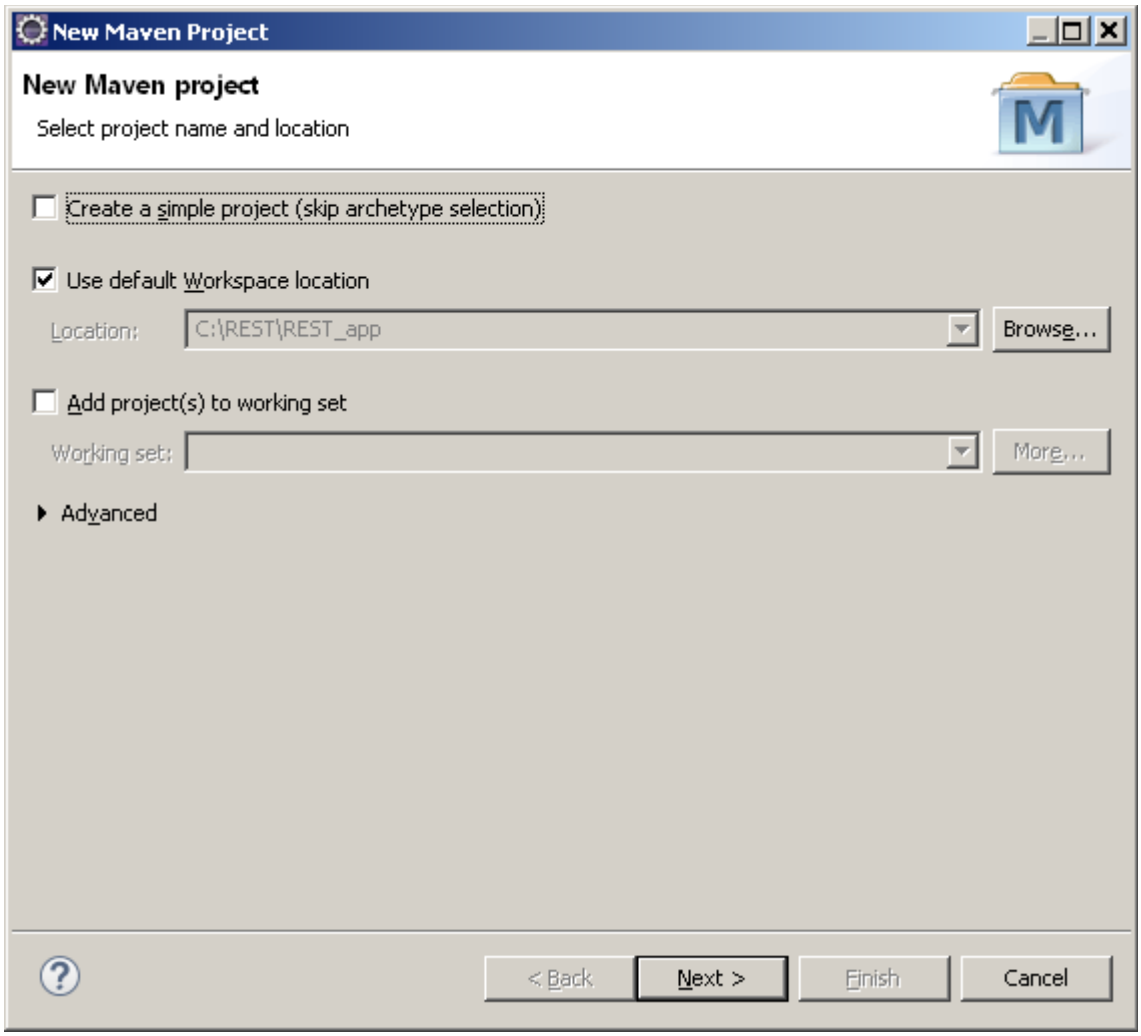
Java
Eclipse
Maven
Tomcat 8
Spring MVC
JPA
Hibernate
Jackson
PostgreSQL
JSON
JQuery

Ülatoodud asjadega eelnevalt mitte kokku puutunud inimesele võib seda tunduda natuke palju. Õnneks ei ole enamikku neist vaja selle ülesande tegemisel eriti uurida – nad on lihtsalt vajalikud et ma saaksime lihtsa vaevaga ja vähese programmeerimisega REST-teenuse käima. Suure osa tööst teeb ära Maven mis pom.xml faili salvestatud tarkvarasõltuvuste info alusel tõmbab võrgust projekti jaoks vajalikud Java teegid.



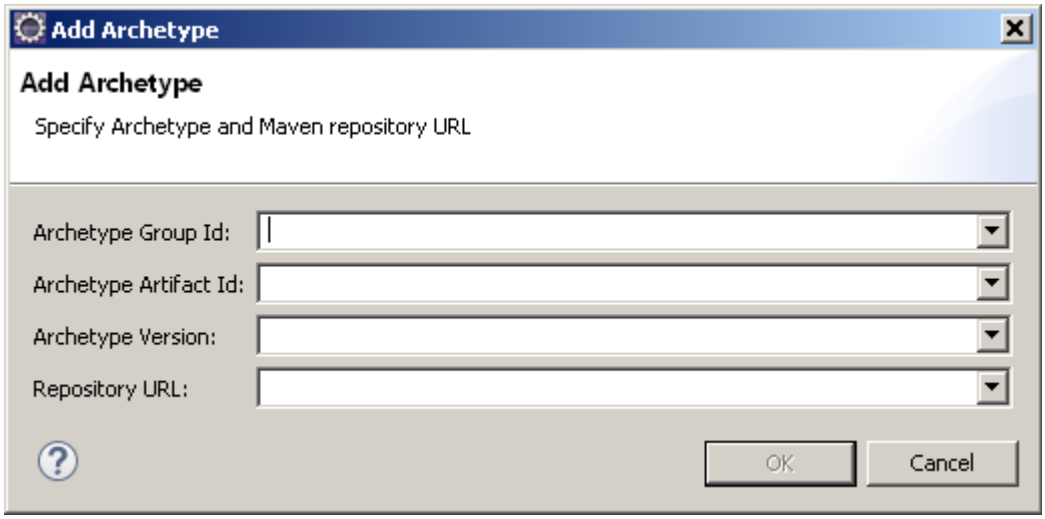
Näite Eclipse projekti loomine toimus nii (võib-olla on kasulik seda teada):



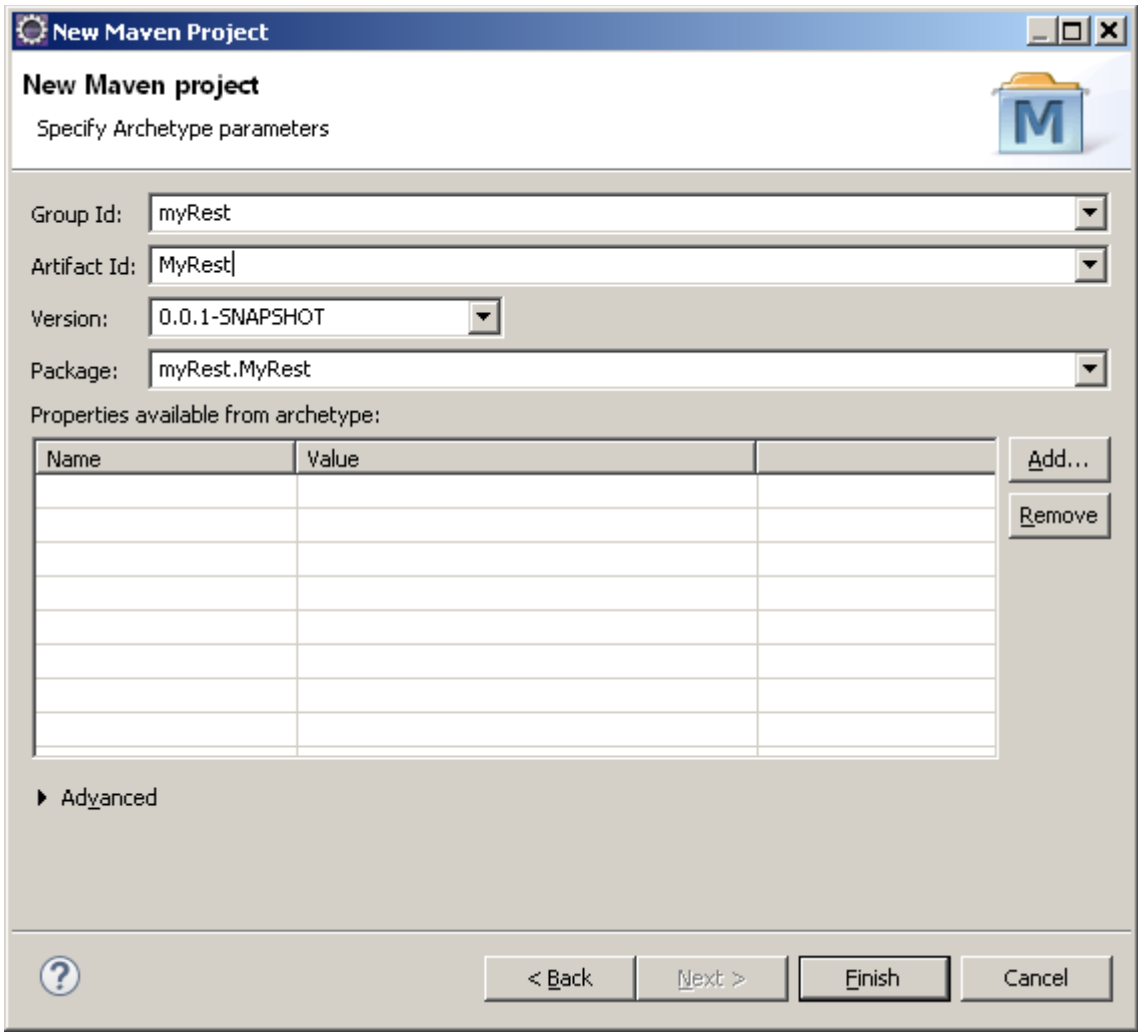


Nüüd tuleb valida arhetüüp. Kui „co.ntier“ ei ole valitav siis „Add Archetype“. vaja on spring-mvc-archetype.Sisestada järgmsied andmed ettetulevasse vormi.

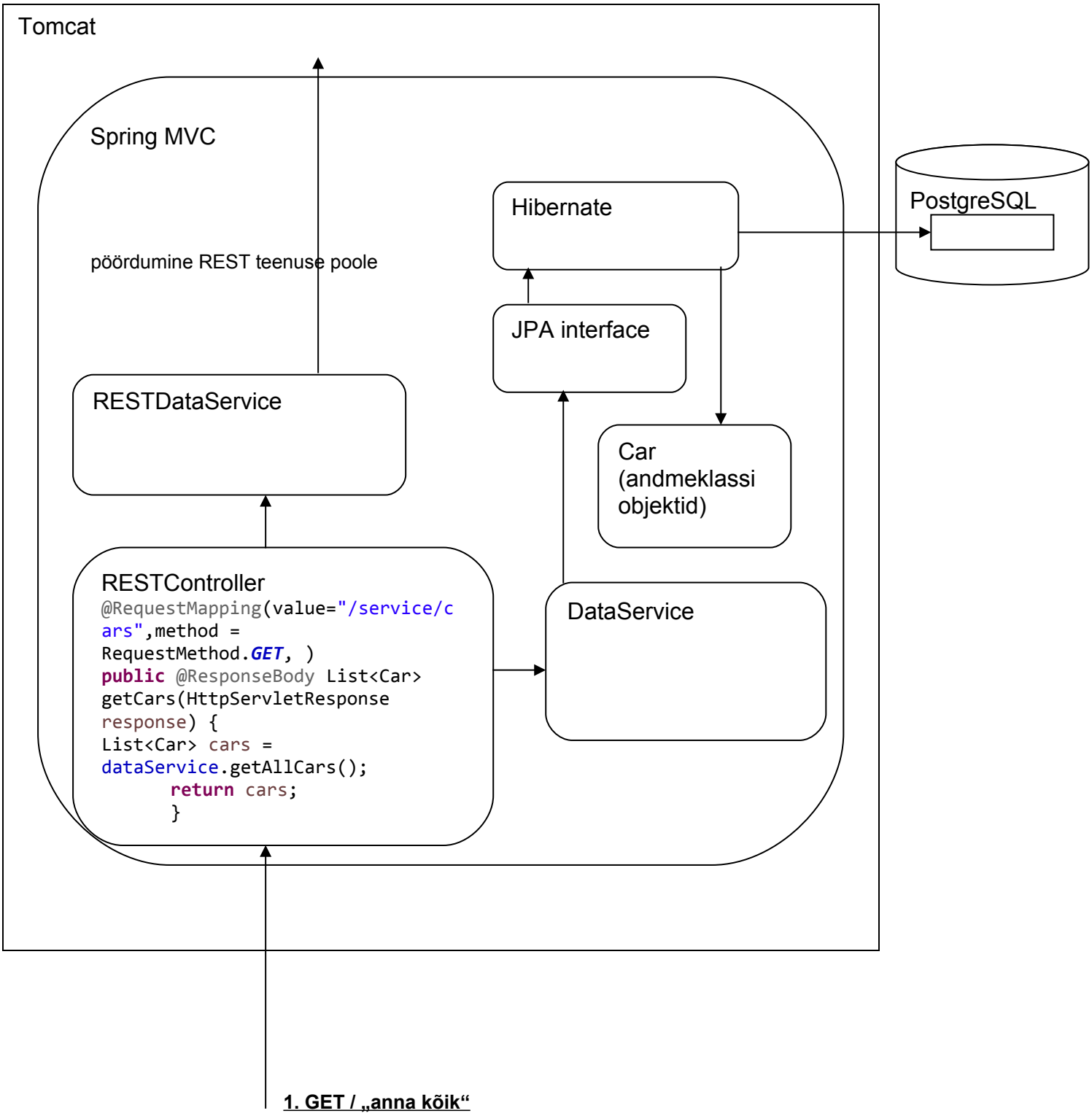
Archetype Group Id: co.ntier
Archetype Artifact Id: spring-mvc-archetype
Archetype Version: 1.0.2
Repository URL: http://maven-repository.com/artifact/co.ntier/spring-mvc-archetype/1.0.2



Kui see on juba üks kord tehtud siis rohkem ei pea enam tegema, uue Maveni projekti loomisel on co.ntier spring-mvc-archetype nüüd valitav.



4.2. Näite komponendid.



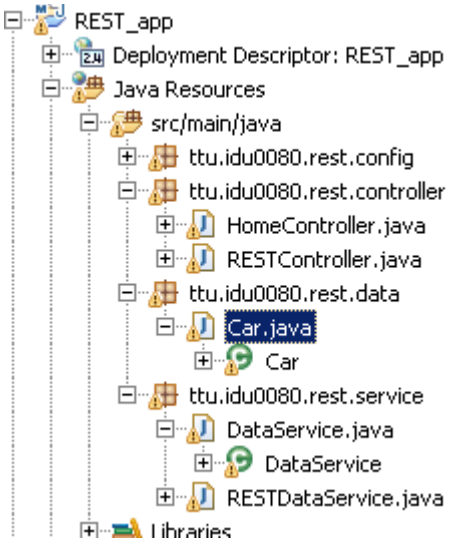
Kõik brauserilt (pole vahet kas AJAX või otse URL-i sisestuse tulemusel) tulnud HTTP pöördumised võetakse vastu RESTController-i poolt, sealt näete millistele URL-idel on võimalik selle rakenduse poole pöörduda.

```
	@RequestMapping(value="/service/cars",method = RequestMethod.GET, produces =
"application/json")
	public @ResponseBody List<Car> getCars(HttpServletResponse response) throws
IOException{

		List<Car> cars = dataService.getAllCars();
		return cars;
	}
```

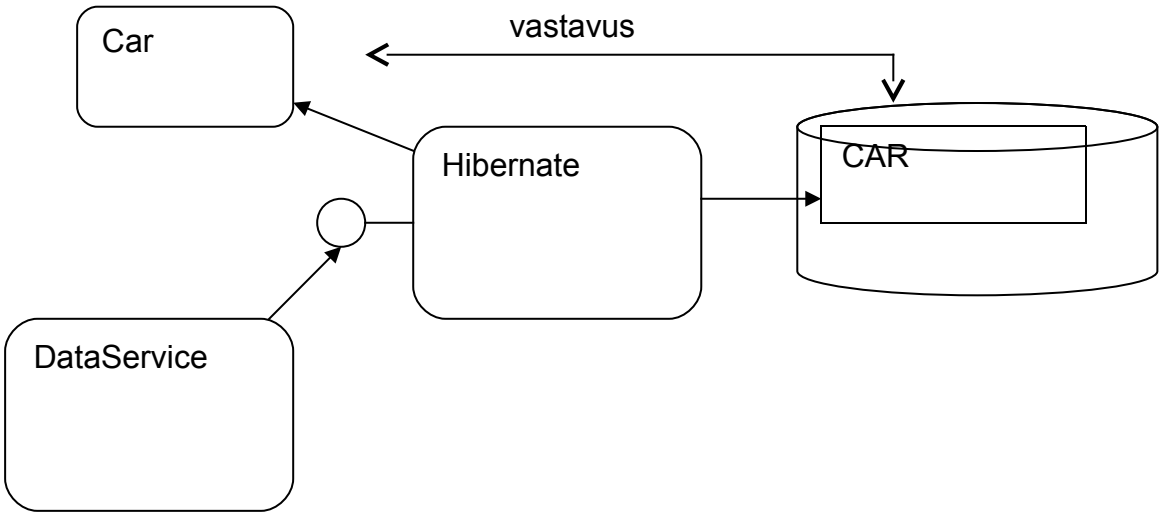
RESTController pöördub andmete saamiseks (või salvestamsieks või lisamiseks või kustutamiseks) DataService poole. DataService pöördub omakorda andmebaasi poole

NB ! Klassis DataService on realiseeritud ka sellised andmebaasipäringud mida teil on vaja nende REST-teenuste realiseerimiseks mida näites ei ära tehtud ei ole (PUT, DELETE, osting). Muidugi peate te neid meetodeid enda vajaduseks mõnevõrra kohandama.

4.3. Andmevahetus rakenduse ja andmebaasi vahel.



Andmebaasiga suhtlemise jaoks on rakenduses DataService , andmeklass Car ja andmebaaside vahevara Hibernate'i millega DataService suhtleb JPA programmeerimisliidese vahendusel.



Car.java

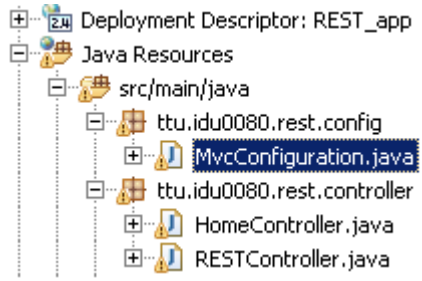
```
@Entity
@Table(name="CAR")
public class Car implements java.io.Serializable  {
        @Id
        @Column(name="id")
        @GeneratedValue(strategy=GenerationType.IDENTITY)
        private long id ;
        private String make ;
        private String model ;
        private String series ;
        private int year ;


        public Car() {
        }


        public long getId() {
                return id;
        }

...
```

Column	Type	Not Null	Default	Constraints
id	numeric(10,0)	NOT NULL	nextval('car_id'::regclass)	
make	text			
model	text			
series	text			
year	numeric(4,0)			

Veel olulisi kohti näites mida vaadata.

* Andmebaasiühenduse parameetrid

MvcConfiguration.java




```
public class MvcConfiguration extends WebMvcConfigurerAdapter{
```

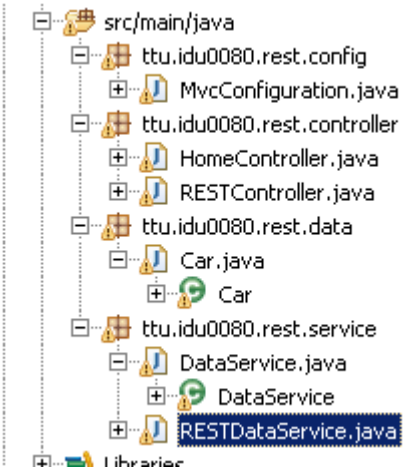
```
      private static final String PROPERTY_NAME_DATABASE_DRIVER =
"org.postgresql.Driver";
    private static final String PROPERTY_NAME_DATABASE_PASSWORD = "";
    private static final String PROPERTY_NAME_DATABASE_URL =
"jdbc:postgresql://imbi.ld.ttu.ee/REST";
    private static final String PROPERTY_NAME_DATABASE_USERNAME = "postgres";
```

Või **MySQL** kasutamise korral:

```
public class MvcConfiguration extends WebMvcConfigurerAdapter{


      private static final String PROPERTY_NAME_DATABASE_DRIVER =
"com.mysql.jdbc.Driver"; //"org.postgresql.Driver";
    private static final String PROPERTY_NAME_DATABASE_PASSWORD = "idumees42";
    private static final String PROPERTY_NAME_DATABASE_URL =
"jdbc:mysql://localhost:3306/car"; //"jdbc:postgresql://imbi.ld.ttu.ee/REST";
    private static final String PROPERTY_NAME_DATABASE_USERNAME = "idumees";
//"postgres";

    private static final String PROPERTY_NAME_HIBERNATE_DIALECT =
"org.hibernate.dialect.MySQLDialect"; //"org.hibernate.dialect.PostgreSQLDialect";
```

* Teise REST teenusega ühenduse võtmise klass ja aadress.

RESTDataService.java



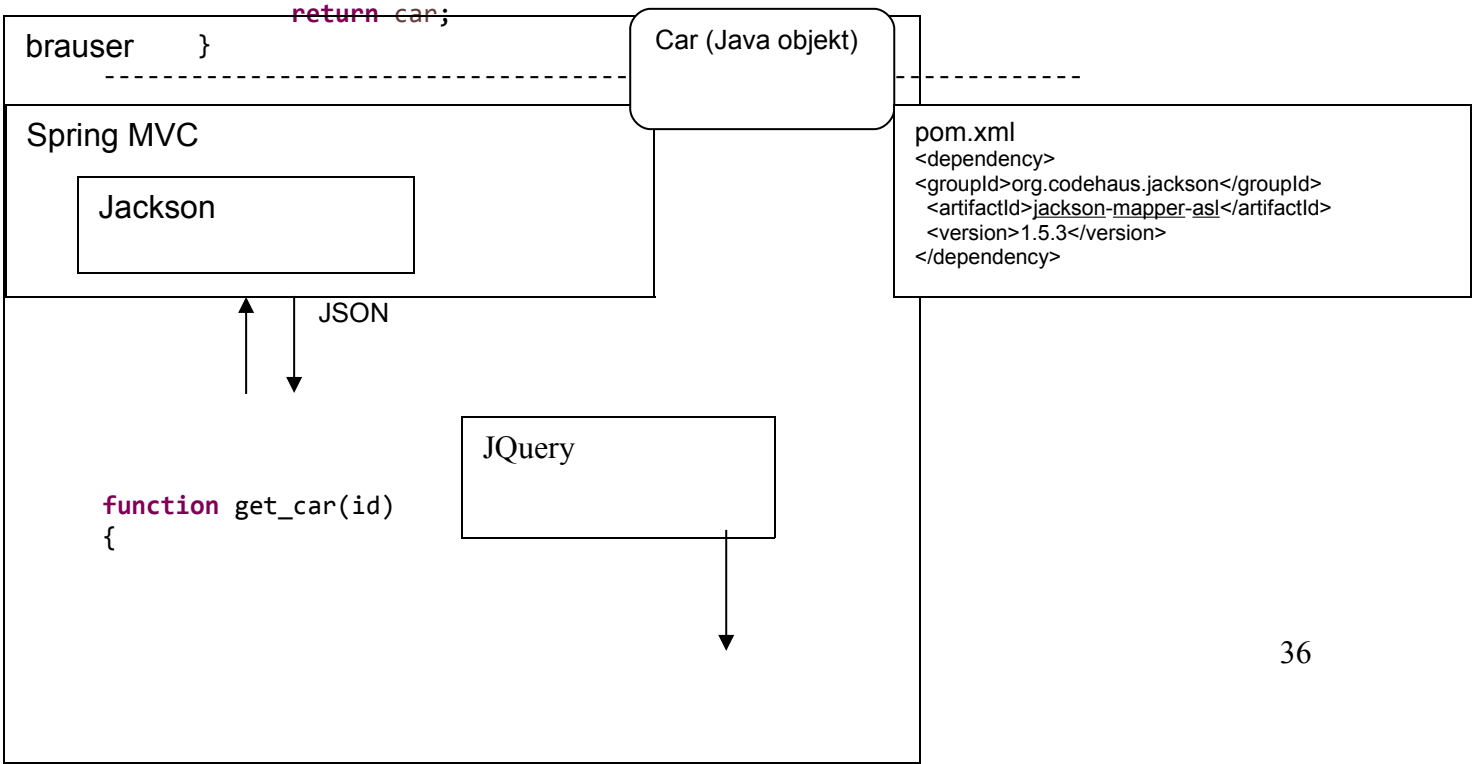

```
public List<Car> getAllCars()  {

            Car[] car_array = null;
            try
            {
                  RestTemplate restTemplate = new RestTemplate();
            car_array =
restTemplate.getForObject("http://localhost:8080/REST_service/service/cars",
Car[].class) ;
            System.out.println("Autosid REST-teenusest:" + car_array.length);
            }
```

```
                catch(Exception ex)
                {
                        System.out.println("RESTDataService.getAllCars():"+
ex.getMessage());
                }

                List<Car> car_list= Arrays.asList(car_array);
                return car_list;
        }
```

4.4. Andmevahetus brauseri ja Spring MVC vahel

Kuidas tekivad JSON formaadis andmed mis server brauserisse saadab? Spring MVC teeb selle teisenduse automaatselt kui kontrolleris on vastav meetod õigesti kirjutatud.Meetod tagastab Java objekti, JSON-iks konverteerib andmed Spring MVC eeldusel et projekti teekide hulgas on Jackson-i teegid.

```
--------------------RESTController-----------------------------------------------------
        @RequestMapping(value="/service/car/{id}",method = RequestMethod.GET,
produces = "application/json")
        public @ResponseBody Car getCar(@PathVariable int id) throws IOException{

                Car car = dataService.getCarById(id);
                return car;
        }
```



```
$.ajaxSetup({ cache: false });
$.ajax({

    url: 'service/car/' + id ,
    type: "GET",
    dataType: 'json',
    success: function(data) {
       car_from_server = data;
       display_car(data);
         console.log(JSON.stringify(data));

    }
  });


}
```

Car (Javascripti objekt)

Üle võrgu liiguvad andmed tekstikujul (JSON), kliendi poolel (braiuseris) saadakse nendest Javascripti objekt JQuery vahendusel.

4.5. Põhilised muudatused näite edasiarendamisel enda lahenduse suunas.

1. Muuta ära andmebaasiühenduse parameetrid failis MvcConfiguration.java

2. Teha uus , endale sobiv andmeklass (kasutades klassi Car eeskuju).

3. Muuta DataService klassi meetodeid , need peavad niid klassi Car asemel viitama teie andmeklassile.

4. Muuta RESTControllerit

5. Muuta ja täiendada Javascripti osa.