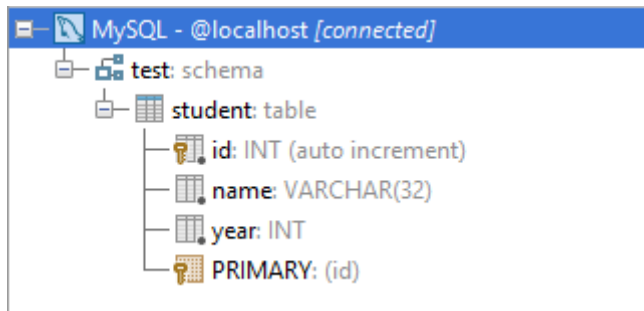


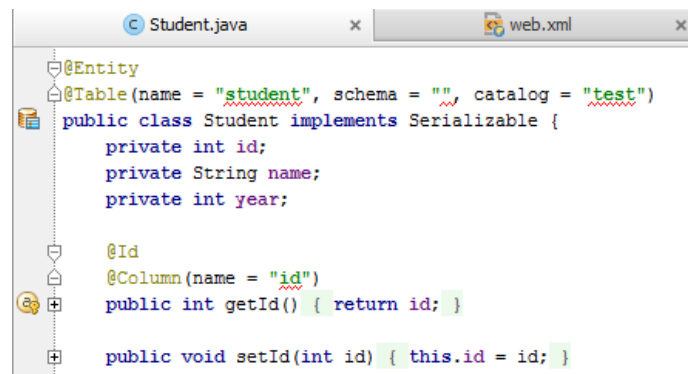
Laboratory work №3

In this laboratory work I had to implement a web application for editing students list. I should have a list with edit and delete buttons for each student and form for adding new student. According to exercise, I could use Hibernate (library for Java) that enables to more easily write application. Also all data must be stored in SQL database, therefore I used open server.exe and create the table with ID, name and age columns.

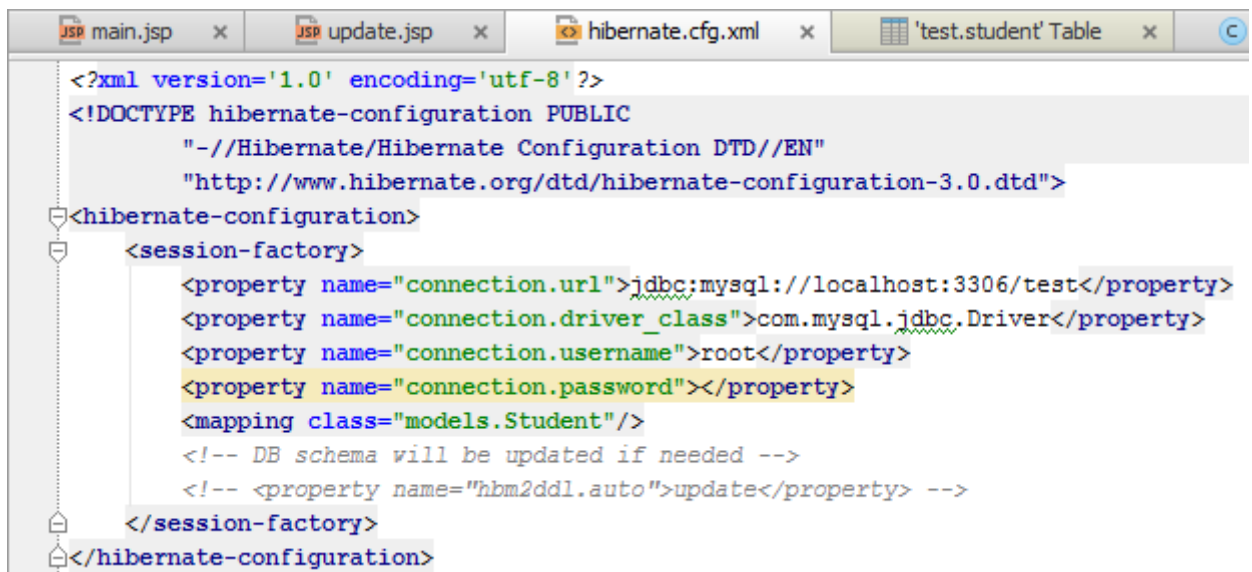
Creating the table student:



As you can see, the table student consists of name, year and id that has primary key and auto increment function (to increment the id when I add student). Then my table was generated into code, like this:



Creating file hibernate.cfg.xml:



Creating Hibernate Utility class that will take care of the beginning of the session:

```
package models;

import ...

public class HibernateUtil {

    private static final SessionFactory sessionFactory = buildSessionFactory();
    private static ServiceRegistry serviceRegistry;

    private static SessionFactory buildSessionFactory() {
        try {
            // Создает сессию с hibernate.cfg.xml
            Configuration configuration = new Configuration();
            configuration.configure();
            serviceRegistry = new ServiceRegistryBuilder().applySettings(configuration.getProperties()).buildServiceRegistry();

            return configuration.buildSessionFactory(serviceRegistry);
        } catch (Throwable ex) {
            System.err.println("Initial SessionFactory creation failed." + ex);
            throw new ExceptionInInitializerError(ex);
        }
    }

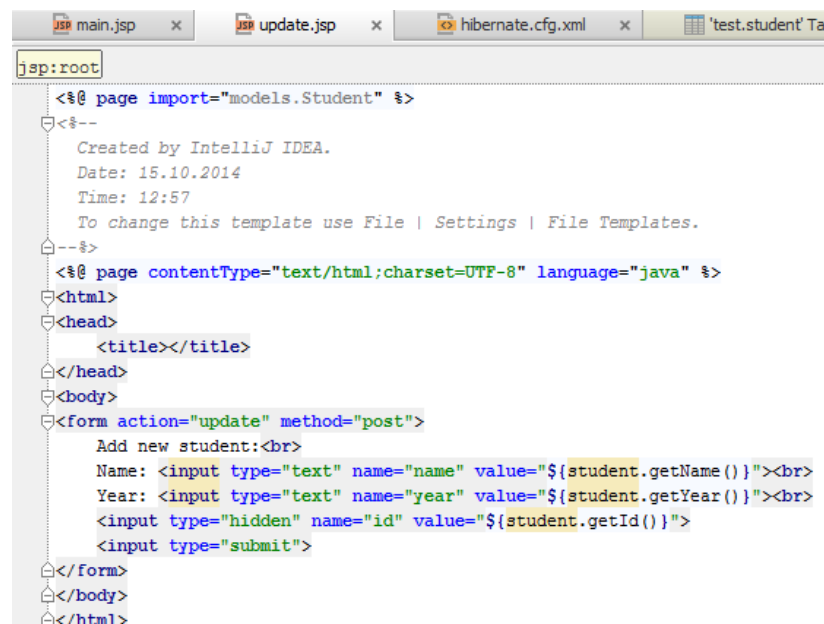
    public static SessionFactory getSessionFactory() { return sessionFactory; }

    public static void shutdown() {
        // Чистит кеш и закрывает соединение с БД
        getSessionFactory().close();
    }
}
```

For interface of my application I used two pages, where one of them is main page (for adding and deleting the student) and another for editing the information about them.

A JSP page is basically a web page with traditional HTML and bits of Java code. When a JSP page is called, it will be compiled (by the JSP engine) into a Java servlet. At this point the servlet is handled by the servlet engine, just like any other servlet. The servlet engine then loads the servlet class and executes it to create dynamic HTML to be sent to the browser. The servlet creates any necessary object, and writes any object as a string to an output stream to the browser.

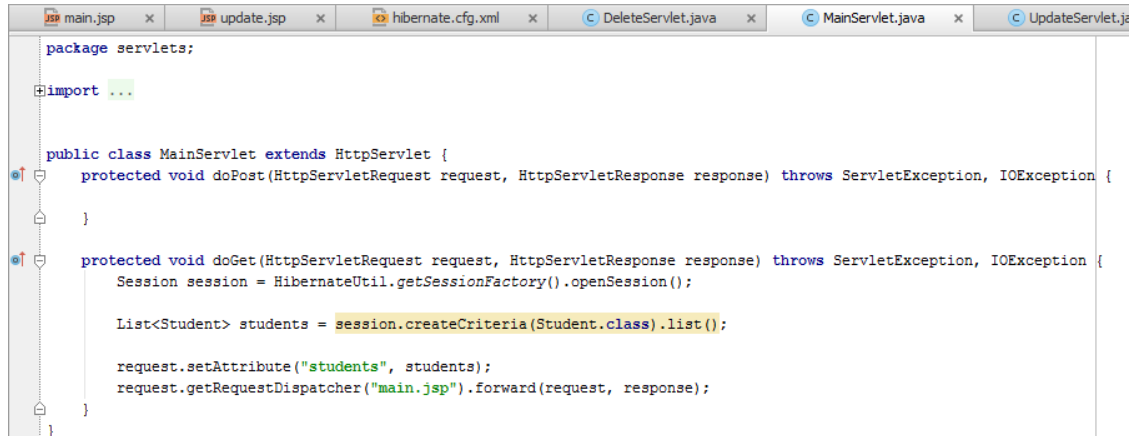
Update jsp page:



```
jsp:root
<%@ page import="models.Student" %>
<!--
Created by IntelliJ IDEA.
Date: 15.10.2014
Time: 12:57
To change this template use File | Settings | File Templates.
-->
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
<title></title>
</head>
<body>
<form action="update" method="post">
    Add new student:<br>
    Name: <input type="text" name="name" value="${student.getName()}"><br>
    Year: <input type="text" name="year" value="${student.getYear()}"><br>
    <input type="hidden" name="id" value="${student.getId()}">
    <input type="submit">
</form>
</body>
</html>
```

Servlets are the Java platform technology of choice for extending and enhancing Web servers. They have access to the entire family of Java APIs, including the JDBC API to access enterprise databases. Also can access a library of HTTP-specific calls and receive all the benefits of the mature Java language.

For example MainServlet code:



```
package servlets;

import ...

public class MainServlet extends HttpServlet {

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        Session session = HibernateUtil.getSessionFactory().openSession();

        List<Student> students = session.createCriteria(Student.class).list();

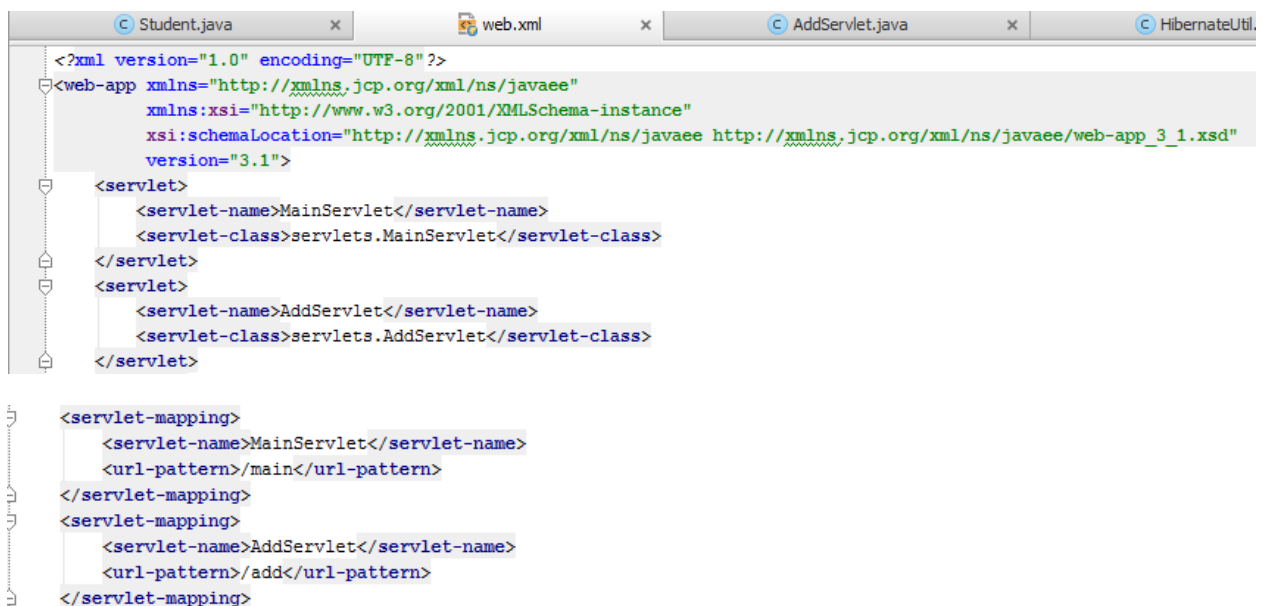
        request.setAttribute("students", students);
        request.getRequestDispatcher("main.jsp").forward(request, response);
    }
}
```

I created session, put all students into list and can read them from list.

Also I downloaded Glass Fish Application Server which can also be used as a Web Server (Http Server). There was some difficulties in connecting Hibernate, GlassFish and Maven with IntelliJ Idea. Moreover I didn't understand how to connect with SQL database, but then I recognized that I can create table inside of IntelliJ Idea.

A web Server means Handling HTTP requests (usually from browsers). A Servlet Container means that it can handle servlets & JSP. An Application Server (GlassFish) means it can manage Java EE applications (usually both servlet/JSP).

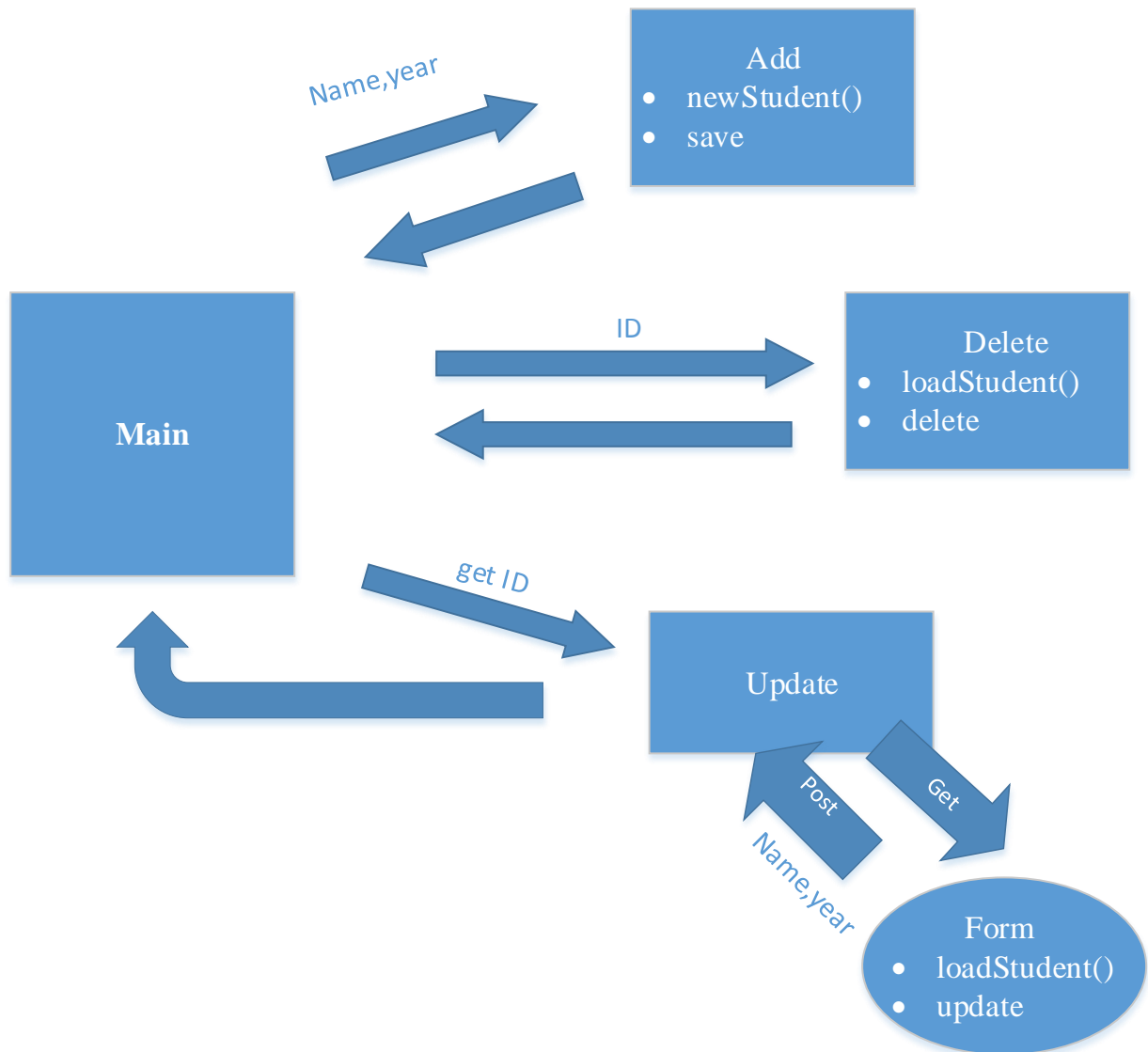
I wrote web.xml file, because without this file glassfish can't see servlets:



```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">
    <servlet>
        <servlet-name>MainServlet</servlet-name>
        <servlet-class>servlets.MainServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>AddServlet</servlet-name>
        <servlet-class>servlets.AddServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>MainServlet</servlet-name>
        <url-pattern>/main</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>AddServlet</servlet-name>
        <url-pattern>/add</url-pattern>
    </servlet-mapping>
</web-app>
```

All servlets:



Finally after switching on the Open Server.exe I got my web application:

List of all students:

ID	Name	Year		
1	Adema	19	[Update]	[Delete]
5	Dosya	19	[Update]	[Delete]

Add new student:

Name

Year

Main:

Add new student:

Name:

Year:

and Edit: