KANDIDAT

251

PRØVE

DATA1700 1 Webprogrammering

Emnekode	DATA1700
Vurderingsform	Skriftlig eksamen under tilsyn
Starttid	22.05.2023 07:00
Sluttid	22.05.2023 10:00
Sensurfrist	10.06.2023 21:59

PDF opprettet	12.09.2024 09:32

Section 1

Oppgave	Tittel	Oppgavetype
ℹ	Eksamensinformasjon	Informasjon eller ressurser
ℹ	Forside	Informasjon eller ressurser
1	Task 1	Programmering
2	Task 2	Programmering
3	Task 3	Programmering
4	Task 4	Programmering
5	Task 5	Programmering
6	Task 6	Programmering
7	Task 7	Programmering

1 Task 1

We start from the presumption that we have already created a new, clean Java Spring Boot project where we added all the necessary dependencies for a basic CRUD(create, read, update, delete) web application.

1. Let's create a simple UI first. Create a form using HTML,CSS,JS. You should have simple user validation such as check for null on all fields. Please add fields for name, surname, date of birth, Social Security Number, phone number, email address, city, street and an HTML button. Please also add custom validation for email address and phone number (hint: regex).

Skriv ditt svar her

```
<html>
    <head>
        <meta charset ="UTF-8">
        <script>JQUERY LINK</script>
         <title>
        Information
    </title>

    </head>
    <body>
        <table>
            <tr>

             <tr>
                <td>name</td>
                <td><input type ="text" id="name" placeholder="Write name" onchange
                <span id="wrongname" style="color:red"></span></td>
             </tr>

             <td>Surname</td>
                <td><input type ="text" id="surname" placeholder="Write name" oncha
                <span id="wrongsurname" style="color:red"></span></td>
             </tr>
              <tr>
                <td>Date of birth</td>
                <td><input type ="text" id="date" placeholder="Write date of birth"
                <span id="wrongdate" style="color:red"></span></td>
             </tr>

              <tr>
                <td>Social security number</td>
                <td><input type ="text" id="ssn" placeholder="Write Social Security
                    .value)">
                <span id="wrongssn" style="color:red"></span></td>
             </tr>

              <tr>
                <td>Phone number</td>
                <td><input type ="text" id="number" placeholder="Write Phone number
                    >
                <span id="wrongnumber" style="color:red"></span></td>
             </tr>

              <tr>
                <td>Email</td>
                <td><input type ="text" id="email" placeholder="Write Email" onchar
                <span id="wrongemail" style="color:red"></span></td>
             </tr>

              <tr>
                <td>City</td>
```

```
50                <td><input type ="text" id="City" placeholder="Write City" onchange
51                <span id="wrongcity" style="color:red"></span></td>
52              </tr>
53
54               <tr>
55                <td>Street</td>
56                <td><input type ="text" id="street" placeholder="Write Streetname"
57                <span id="wrongstreet" style="color:red"></span></td>
58              </tr>
59
60               <tr>
61                <td></td>
62                <td><input type ="button" id="register" value ="Register" onclick="
63                <span id="status"></span></td>
64              </tr>
65          </table>
66
67          <script>
68
69          function validatename(name){
70              if(name){
71                  return true;
72              } else{
73                  $("#wrongname").html("Write name")
74                  return false;
75              }
76
77          }
78
79            function validatesurname(surname){
80              if(surname){
81                  return true;
82              } else{
83                  $("#wrongsurname").html("Write surname")
84                  return false;
85              }
86
87          }
88
89            function validatedate(date){
90              if(date){
91                  return true;
92              } else{
93                  $("#wrongdate").html("Write date of birth")
94                  return false;
95              }
96
97          }
98
99            function validatessn(ssn){
100             if(ssn){
101                 return true;
102             } else{
103                 $("#wrongssn").html("Write social security number")
104                 return false;
105             }
106
107         }
108
109           function validatephone(phone){
110             if(phone){
111                 return true;
112             } else{
113                 $("#wrongphone").html("Write Phone number")
114                 return false;
115             }
116
117         }
118
```

```
        function validateemail(email){
          if(email){
              return true;
          } else{
              $("#wrongemail).html("Write enail")
              return false;
          }

      }

        function validatecity(city){
          if(city){
              return true;
          } else{
              $("#wrongcity").html("Write city")
              return false;
          }

      }

        function validatestreet(street){
          if(street){
              return true;
          } else{
              $("#wrongstreet").html("Write street")
              return false;
          }

      }

      function validatephonereg (phone){
          let regex = /^[0-9]{8}$/;
          if(regex.test(phone)){
              return true
          } else {
              $("#wrongnumber").html("There has to be eight numbers");
              return false;
          }
      }

      function validateemailreg (email){
          let regex = /^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\-[A-Za-z]{2,}$/;
          if(regex.test(email)){
              return true
          } else {
              $("#wrongemail").html("Write a email");
              return false;
          }
      }



      </script>
    </body>
</html>
```

2 Task 2

2. Create a JS method with a JS object that will take into account all the fields described in the first task.Display the information you get into your new object inside a console log or an alert. (Please make sure to show the proper code for activation of this method inside the HTML button tag - you can copy the button tag you used in the first task and just add code on top of it). Also, please make a call towards a Java rest endpoint using Jquery where to send the object you just filled. (We don't have the endpoint for now, but let's imagine it's name is : "/saveCitizen")

Skriv ditt svar her

```
Button from task 1

<tr>
                <td></td>
                <td><input type ="button" id="register" value ="Register" onclick="F
                <span id="status"></span></td>
            </tr>


function validateinfo (info){
    nameOK = valiedatename(info.name);
    surnameOK = valiedatesurname(info.surname);
    dateOK = valiedatedate(info.date);
    ssnOK = valiedatessn(info.ssn);
    phoneOK = valiedatephone(info.phone);
    emailOK = valiedateemail(info.email);
    cityOK = valiedatecity(info.city);
    streetOK = valiedatestreet(info.street);

    if(nameOK && surnameOK && dateOK && ssnOK && phoneOK && emailOK && cityOK && str
        return true;
    } else {
        return false;
    }

}

function Registerinfo(){
    let info = {
        "name" : $("#name").val(),
        "surname" : $("#surname").val(),
        "date" : $("#date").val(),
        "ssn" : $("#ssn").val(),
        "phone" : $("#phone").val(),
        "email" : $("#email").val(),
        "city" : $("#city").val(),
        "street" : $("#street").val(),
        };
        if(validateinfo (info)){
            console.log(info)
            $.post("/saveCitizen", info, function(){
                $("#status").html("Info is stored");

                $("#name").val(""),
                $("#surname").val(""),
                $("#date").val(""),
                $("#ssn").val(""),
                $("#phone").val(""),
                $("#email").val(""),
                $("#city").val(""),
                $("#street").val(""),

            });
        } else {
```

```
55              $("#status").html("Info is not stored. Make sure every field has an corr
56
57
58          }
59  }
```

3 Task 3

Java task: Create your first Controller class with the proper annotations. Create an endpoint "/hello" to test that your controller is configured correctly. The endpoint should return a string with a message that will be displayed on the browser when someone interrogates that particular endpoint. (Be careful about the type of mapping you use for your endpoint).

Skriv ditt svar her

```
1  @Restcontroller
2  public class testController{
3      @Getmapping ("/hello")
4      public String hello (){
5          return "Hello Cosmin, i hope this message is displayed on the browser :)";
6      }
7  }
```

4 Task 4

Java & SQL task: Create a new model class in Java that would map the input fields you created in the first task. Make sure to have all the field types similar. If you are going to use Hibernate JPA, please make sure you use the proper adnotations. Also, please write the SQL code necessary for the creation of a table that follows the rules mentioned above.

NB: Don't worry if the editor is set for Java, I don't search for SQL sintax perfection:).

Skriv ditt svar her

```
// Since i have used Hibernate to create a table, do i not find it necessary to crea

@Entity // this adnotaion will let Hibernate know taht we want a SQL table that cont
    below
@Table (name = "Citizen") // name for the table
@Data // instead of @Getter and @Setter, which is used for getters and setters of ea
    these adnotations)
@NoArgsConstructor // empty constructor

public class Citizen {
    @id // defines PK for the SQL table
    @GeneratedValue // PK will also get AUTO_INCREMENT (autogenerated value for id)

    private Integer id;
    private String name;
    private String surname;
    private String date;
    private Integer ssn;
    private Integer phone;
    private String email;
    private String city;
    private String street;


public Citizen (Integer id, String name, String surname, String date, Integer ssn. I
    , String street){
this.name = name;
this.surname = surname;
this.date = date;
this.ssn = ssn;
this.phone = phone;
this.email = email;
this.city = city;
this.street = street;

}

@Override
public String toString (){

    return "Citizen{" + "id=" +id+ "name="+name+ "surname="+surname+ "date=" +date+
        +email+ "city=" +city+ "street=" +street+ '}';


}
}

```

5 Task 5

Create a new endpoint in your controller that will take care of the input it receives from JS. (The JS object you created at task 2.) Make sure that all the information you received is mapped in the model class that you defined at task 4. Now comes the funny part, you have to save that information into the DB.

Let's consider that you already set up a connection for the DB and it works fine. You can choose any way you want to save the data in the DB, if you use the "new way" with Hibernate and JPA, please also define the interface. If the transaction with the DB is not successful make sure to handle the error.

Skriv ditt svar her

```
@Restcontroller

public class CitizenController {

    Logger logger = LoggerFactory.getLogger(CitizenController.class);

    @Autowired
    CitizenRepository citizenRepo /*Just to show even though i dont have a repo*/

    @Autowired
    private HttpSession session;

    @Postmapping ("/saveCitizen")
    public String AddCitizentoDB(@RequestBody Citizen citizen) throws IOError {

        try{ // tries to save the cictizen object in DB, if not the server will hand
        System.out.println(citizen.toString());
        logger.info(String.valueOf(citizen));
        citizenRepo.save(citizen);
        return "Citizen is saved in database";
    } catch (Exception e){
        logger.error(e.getMessage());
        return e.getStackTrace().toString();
    }
}
```

6 Task 6

Java task: You have the next scenario: A user who is logged in (let's say an administrator) would like to operate some sensitive changes to the data bases. Let’s think of a situation where you don’t like that citizens < 18 registered. If that’s the case please delete the citizens < 18 from the DB and then logout.
You will need to create 2 endpoints to manage sessions. The first endpoint for login, the second one for logout. (pay attention on how you use the session object).

You will also have to create an endpoint to operate the changes in the DB. First, check if you are logged in. If you are, then procede with the changes. (Pay attention to the calls you have to make in the Data Base. We first need to retrieve the list, then check the condition (citizen age < 18), and in the end we have to interogate the DB again in order to delete those who don't fit the description).

Skriv ditt svar her

```
@Getmapping ("/login")
public String login(){
    if(session.getAttribute("Innlogget")!=null){
        List<Citizen> citizenlist = citizenrepo.findAll();
        List<Citizen> citizenunder18 = citizenlist.stream().filter(citizen->citizen.
            .toList));
        // i dont know the function to find age that is less that 18, however i woul
            something like this.
        citizenlist.forEach(citizen-> {citizen.setEmail("No longer interested in thi
        citizenrepo.save(citizen);});
    }
    session.invalidate
}
```

7 Task 7

Retrieve all the citizens from the application using a new endpoint and send them in the browser as a json response. Use a Logger to show all this data in your server. Return the info by sorting alphabetically ascending using a Java method.

Skriv ditt svar her

```
@Getmapping ("/getCitizenfromDB")
public List<Citizen> getCitizenFromDB(){
    List<Citizen> citizenlist = citizenrepo.findAllByOrderByNameASC();
    citizenlist.foreach(x-> logger.info(String.valueOf(x)));
    return citizenlist;
}
```