KANDIDAT

177

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	11.09.2024 14:27

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

```
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>USER</title>
<script src="index.js"></script>
//Jquery
</head>
<body>
<table>
    <tr>
        <td>Name: </td>
        <td><input type="text" onchange="validateName(this.value)" id="name" />
        <span id="wrongName" style="color: red"></span></td>
    </tr>
    <tr>
        <td>Surname: </td>
        <td><input type="text" onchange="validateSurname(this.value)" id="surname"
        <span id="wrongSurname" style="color: red"></span></td>
    </tr>
    <tr>
        <td>Date of Birth: </td>
        <td><input type="text" onchange="validateDate(this.value)" id="date" />
        <span id="wrongDate" style="color: red"></span></td>
    </tr>
    <tr>
        <td>Social Security Number: </td>
        <td><input type="number" onchange="validateSecurity(this.value)" id="securi
        <span id="wrongSecurity" style="color: red"></span></td>
    </tr>
    <tr>
        <td>Phone Number:</td>
        <td><input type="number" onchange="validateNumber(this.value)" id="number"
        <span id="wrongNumber" style="color: red"></span></td>
    </tr>
    <tr>
        <td>Email: </td>
        <td><input type="text" onchange="validateEmail(this.value)" id="email" />
        <span id="wrongEmail" style="color: red"></span></td>
    </tr>
    <tr>
        <td>City: </td>
        <td><input type="text" onchange="validateCity(this.value)" id="city" />
        <span id="wrongCity" style="color: red"></span></td>
    </tr>
    <tr>
        <td>Street: </td>
        <td><input type="text" onchange="validateStreet(this.value)" id="street" />
        <span id="wrongStreet" style="color: red"></span></td>
    </tr>
    <tr>
```

```
    <td><button id="register" onclick="registerInfo()">Register</button></td>
    </tr>
</table>
</body>
</html>

//functions

function validateName(name){
if(){
$("#wrongName").html("");
return true;
} else {
$("#wrongName").html("This cannot be empty!");
return false;
}
}
function validateNumber(number){
let regex = /^[0-9]{8}$/;
if(regex.test(number)){
$("#wrongNumber").html("");
return true;
} else{
$("#wrongNumber").html("It has to be 8 numbers!");
return false;
}
}
function validateEmail(email){
let regex = /^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$/;
if(regex.test(email)){
$("#wrongEmail").html("");
return true;
} else{
$("#wrongEmail").html("It has to be 8 numbers!");
return false;
}
}
function validateSurname(surname){
if(){
$("#wrongSurname").html("");
return true;
} else {
$("#wrongSurname").html("This cannot be empty!");
return false;
}
}
function validateDate(date){
if(){
$("#wrongDate").html("");
return true;
} else {
$("#wrongDate").html("This cannot be empty!");
return false;
}
}
function validateSecurity(securitynumber){
if(){
$("#wrongSecurity").html("");
return true;
} else {
$("#wrongSecurity").html("This cannot be empty!");
return false;
}
}
function validateCity(city){
if(){
$("#wrongCity").html("");
return true;
```

```
} else {
$("#wrongCity").html("This cannot be empty!");
return false;
}
}
function validateStreet(street){
if(street){
$("#wrongStreet").html("");
return true;
} else {
$("#wrongStreet").html("This cannot be empty!");
return false;
}
}
function registerInfo(info){  //call this function when you want to validate everyt
nameOK = validateName(info.name);
surnameOK = validateSurname(info.surname);
dateOK = validateDate(info.date);
securityOK = validateSecurity(info.securitynumber);
numberOK = validateNumber(info.number);
emailOK = validateEmail(info.email);
cityOK = validateCity(info.city);
streetOK = validateStreet(info.street);

if(nameOK && surnameOK && dateOK && securityOK && numberOK && emailOK && cityOK &&
return true;
} else {
return false;
}

}


```

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

```
// using button from task 1
// creating new JS method and calling it sendInfo
    <tr>
    <td><button id="register" onclick="sendInfo()">Register</button></td>
    </tr>


function sendInfo(){

info {
"name": $("#name").val();
"surname": $("#surname").val();
"date": $("#date").val();
"securitynumber": $("#securitynumber").val();
"number": $("#number").val();
"email": $("#email").val();
"city": $("#city").val();
"street": $("#street").val();
}

if(registerInfo(info)){
$.post("/saveCitzen",info,function()){
    console.log(info)
});
}


}

```

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

```
@RestController
public class modelController {

@GetMapping(/hello)
public String hello () {
    return "Hello!";
}
}
```

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

```java
@Entity
@Table(name = "Model")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Model{
    @Id
    private Integer socialnumber;
    private String name;
    private String surname;
    private String date;
    private String number;
    private String email;
    private String city;
    private String street;


}

@GETTER
@SETTER
@NoArgsConstructor
@AllArgsConstructor

public class ModelDTO {
    private Integer modelNumber
    private String modelName;
    private String modelSurname;
    private String date;
    private String number;
    private String email;
    private String city;
    private String street;


    public CustomerDTO (Integer modelNumber,String modelSurname, String date, String
        String city, String street ){
        this.modelNumber = modelnumber;
        this.modelName = modelname;
        this.modelSurname = modelsurname;
        this.date = date;
        this.number = number ;
        this.email = email;
        this.city = city ;
        this.street = street;

    }

}

// i assume that you mean that we should make a class(which makes a table) and DTO,
```

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 ModelController {

    @Autowired
    ModelRepository modelRepository


    @PostMapping("/saveClient")
    public String saveClient(@RequestBody ModelDTO modelDTO){
        Model model = new Model(ModelDTO.getModelNumber(),ModelDTO.getModelName(),Mo
            .getDate(),ModelDTO.getNumber(),ModelDTO.getEmail(),ModelDTO.getCity(),M
        try{
        modelRepository.save(Customer);
        } catch(exception e){
            return"Something went wrong while sending to DB";
        }
        return"Client was succesfully saved.";
    }
}

@Repository

public interface ModelRepository extends JpaRepository<Model,Integer>{


    @Autowired
    private JdbcTemplate db;

}
```

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

```
@RestController
public class ModelController {
    @Autowired
    HttpSession session;

    @Autowired
    ModelRepository modelRepository

    @PostMapping("login")
    public void login(Model model){
        if(modelRepository.checkSocialnumber(model){
            session.setAttribute("loggedin", model)
        }
    }

    @GetMapping("logout")
    public void logout() {
        session.removeAttribute("loggedin");
    }

    public boolean checkLogin(){
        if(session.getAttribute("loggedin") != null){
            return true;
        }else {
            return false;
        }
    }


    @GetMapping("/DeleteUnder18")
    public void DeleteUnder18(Model model, HttpservletResponse response) throws IOEx
        if(checkLogin()){
         deleteUnder18()
        }
    }
}

//I will be checking if you are over 18 using social number
public interface ModelRepository extends JpaRepository<Model,Integer>{

    @Autowired
    private JdbcTemplate db;

    public boolean checkSocialnumber(Model model){
        String sql = "SELECT * FROM Model WHERE YEAR(date)>2004 AND socialnumber=?"
```

```
        String sql = "SELECT * FROM Model WHERE YEAR(date)>2004 AND SocialNumber=?
        Model dbModel = db.queryForObject(sql,BeanPropertyRowMapper.newInstance(Mode
        if(dbModel){  //check if empty
            return true;
        } else {
            return false;
        }

    }

    public boolean deleteUnder18(Model model){
          String sql = "DELETE FROM Model WHERE YEAR(Date)<2004"
          db.update(sql)
    }
}
```

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

```
@RestController
public class ModelController {

@Autowired
ModelRepository modelRepository

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

@GetMapping("\GetAll")
public List<Model> GetAll(){
    modelRepository.findAll();
    try{
    List <Model> model = modelRepository.findByOrderByName();
    logger.print(model);
    return model;
    } catch {
        logger.error("Mistake in getting out data");
    }
}


}

public interface ModelRepository extends JpaRepository<Model,Integer>{


    List<Model> findByOrderByName();

}
//JSON- response
function GetAll(){
$.get("/GetAll",function(info){
    if(info.length>0){
        return formatData(info)
    }
}
}
function formatData(info){
    let out = "<table><tr><th></th></tr>";
    for (const model of info){
        ut +="<tr><td>"+model.name ->...
    }

    return ut;
}
```