# ВЕБ ПРОГРАМИРАЊЕ

1. Кој од следниве парчиња код се валидни како REST сервис за добивање на податоци за сите вработени.

   Забелешка: Сметајте дека соодветните мапирања и поставувања на ниво на класа и контролер се направени соодветно на потребите.

   Select one or more:

   a)
   ```
   @GetMapping
   private Flux<Employee> getAllEmployees() {
       return employeeRepository.findAllEmployees();
   }
   ```

   **b)**
   ```
   @Bean
   RouterFunction<ServerResponse> getAllEmployeesRoute() {
     return route(GET("/employees"),
       req -> ok().body(
         employeeRepository().findAllEmployees(), Employee.class));
   }
   ```

   **c)**
   ```
   @Bean
   RouterFunction<ServerResponse> composedRoutes() {
     return
       route(GET("/employees"),
         req -> ok().body(
           employeeRepository().findAllEmployees(), Employee.class))

       .and(route(GET("/employees/{id}"),
         req -> ok().body(
           employeeRepository().findEmployeeById(req.pathVariable("id")), Employee.class)))
   ```

```
            .and(route(POST("/employees/update"),
              req -> req.body(toMono(Employee.class))
                .doOnNext(employeeRepository()::updateEmployee)
                .then(ok().build())));
        }
```

d)
```
@GetMapping
private Mono<Employee> getAllEmployees() {
    return employeeRepository.findAllEmployees();
}
```

2. Кои од следните тврдења се точни за AuthenticationProvider?

   **Select one or more:**

   a. Во иста апликација може да има конфигурирано повеќе од еден AuthenticationProvider
   **b. AuthenticationProvider служи за најава со Authentication објект**
   **c. AuthenticationProvider го управува процесот на најава кај Spring Security**
   d. AuthenticationProvider служи за најава само со корисничко име и лозинка

3. Даден е ентитетот:

```
@Entity
@Table(name = "book_details")
public class BookDetails extends BaseEntity {

  @Column(length = 5000)
  public String description;

  @OneToOne
  public Book book;
}
```

Доколку сакаме да ги пронајдеме сите вредности кои имаат одреден опис, кој метод треба да се додаде во следната класа подолу за Spring Data да ни овозможи добивање на валидните резултати:

```
@Repository
public interface BookDetailsRepository extends CrudRepository<BookDetails, Long> {}
```

**Select one:**

a. **List<BookDetails> findByDescription(String text);**
b. List<BookDetails> searchDetails(String text);
c. List<BookDetails> findByBookDetailsDescriptionLike(String text);
d. List<BookDetails> findByDescriptionLike(String text);

4. Доколку го имаме следното мапирање на ентитетите, кои табели ќе се изгенерираат од страна на JPA.

```
@Entity
@Table(name = "STUDENTS")
public class Student extends BaseEntity {

  @Column(name = "student_index")
  public String index;
  public String firstName;
  public String lastName;
  @ManyToMany
  public List<Course> courses;
}

@Entity
@Table(name = "COURSES")
public class Course extends BaseEntity {

  public String name;
  @ManyToMany
  public List<Student> students;
}
```

Притоа ја имаме следната конфигурација:

```
<property name="jpaVendorAdapter">
    <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"
        p:database="${jpa.database}"
```

```
        p:generateDdl="true"
        p:showSql="false"/>
</property>
```

Select one or more:

**a. STUDENTS**
**b. STUDENTS_COURSES**
**c. COURSES_STUDENTS**
**d. COURSES**

5. Потребно е да се дефинира нарачка за ресторан. За таа цел е дефиниран ентитетот:

```
public class Order {
  public String orderId;
  public String clientContact;
  public int persons;
  public String table;
  public DateTime date;
}
```

Кој од следните мапирања се валидни за овој ентитет?

Select one or more:

a.
```
@Table(name = "RESTAURANT_ORDER")
public class Order {
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
public String orderId;

public String clientContact;
public int persons;
public String table;
public DateTime date;
}
```

b.
```
@Table(name = "RESTAURANT_ORDER")
@Entity
public class Order {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public String orderId;

  public String clientContact;
  public int persons;

  @Column(name = "table_number")
  public String table;
  public DateTime date;
}
```

c.
```
@Entity
public class Order {
  public String orderId;
  public String clientContact;
  public int persons;
  public String table;
  public DateTime date;
}
```

**d.**
```
@Table(name = "RESTAURANT_ORDER")
@Entity
public class Order {

  @Id
  public String orderId;

  public String clientContact;
```

```
public int persons;

@Column(name = "table_number")
public String table;

public DateTime date;
}
```

e.
```
@Entity
public class Order {

 @Id
 public String orderId;

 public String clientContact;
 public int persons;
 public String table;
 public DateTime date;
}
```

6. Кое од следните мапирања на класите Training и Group е валидно?

Select one or more:

a.
```
@Table(name = "TRAINING")
@Entity
public class Training {
 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 public Long id;
 public String name;
 @OneToMany
 public Group group;
```

```
}

@Table(name = "TRAINING_GROUP")
@Entity
public class Group {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public Long id;
  public String name;
  @ManyToOne(mappedBy = "group")
  List<Training> trainings;
}
```

b.
```
@Entity
public class Training {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public Long id;
  public String name;
  @ManyToOne
  public Group group;
}

@Entity
public class Group {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public Long id;
  public String name;
  @OneToMany(mappedBy = "TRAINING")
  List<Training> trainings;
}
```

c.
```
@Table(name = "TRAINING")
@Entity
public class Training {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public Long id;
  public String name;
  @ManyToOne
  public Group group;
}

@Table(name = "TRAINING_GROUP")
@Entity
public class Group {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public Long id;
  public String name;
  @OneToMany(mappedBy = "training")
  List<Training> trainings;
}
```

**d.**
```
@Table(name = "TRAINING")
@Entity
public class Training {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public Long id;
  public String name;
  @ManyToOne
  public Group group;
}
```

```
@Table(name = "TRAINING_GROUP")
@Entity
public class Group {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public Long id;
  public String name;
  @OneToMany(mappedBy = "group")
  List<Training> trainings;
}
```

e.
```
@Table(name = "TRAINING")
@Entity
public class Training {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public Long id;
  public String name;
  @ManyToOne
  public Group group;
}

@Table(name = "TRAINING_GROUP")
@Entity
public class Group {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  public Long id;
  public String name;
  @Transient
  List<Training> trainings;
}
```

7. Нека во веб апликација е поставена следната конфигурацијана view-resolver:

```
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/views/"/>
    <property name="suffix" value=".jsp"/>
</bean>
```

Која ќе биде датотеката која ќе се избере од ViewResolver-от за приказ (view template) при повик на question() методот дефиниран во продолжение?

```
@Controller
@RequestMapping("/exam")
public class ExamController {
  @RequestMapping(value = "/question", method = RequestMethod.POST)
  public String question() {
    return "question";
  }
}
```

Select one:

a. /views/question.jsp

b. **Нема да се селектира ниту една датотека, ќе се генерира JSON**

c. Нема да се селектира ниту една датотека, ќе се изврши редирекција

d. /WEB-INF/jsp/question.jsp

e. /WEB-INF/views/question.jsp

8. Во кој од подолу предложените слоеви е најлогично да биде следната имплементација:

```
public List<User> getAllUsers() throws IOException {
  File usersSource = new File(SOURCE_PATH);
  BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(usersSource)));
  CSVFormat format = CSVFormat.DEFAULT.withHeader().withDelimiter(',');
  CSVParser parser = new CSVParser(br, format);
  List<User> users = new ArrayList<User>();
  for (CSVRecord record : parser)
    users.add(parseUser(record));
  parser.close();
  br.close();
  return users;
}
```

Select one:

a. User Interface
**b. Service**
c. Persistence
d. Domain Model
e. Presentation

9. Кои од следните анотации ги вклучуваат класите во Spring контекстот?

Select one or more:

**a. @Bean**

**b. @Service**

c. @Singleton

d. @Context

e. @Prototype

**f. @Configuration**

10. Во кои од следните сценарија логично е атрибутите да се чуваат во Servlet Context (application scope)?

Select one or more:

**a. Подржани валути за плаќање**

b. ID на тековната сесија

**c. Корисничкото име при најава**

d. Јазикот на корисникот

e. Резултати од пребарување

**f. Контакт информации за страната**

11. Доколку сакате да се регистрирате за добивање на пораки на одреден Publisher на податоци, во кој метод од сервлетот ќе го сторите тоа?

Select one:

a. Конструктор

b. doPost

c. service

**d. init**

e. doGet

f. destroy

12. Различни корисници од различни прелистувачи (browsers) имаат пристап до ист податок зачуван во:

Select one or more:

a. Request

b. Session

c. Query String

**d. ServletContext (application context)**

e. Cookie

13. Доколку во ServletA повикаме dispatcherToServletB.forward(req, resp), што од следното е точно:

Select one or more:

**a. Во ServletB можеме да ги пристапиме Request атрибутите поставени во ServletA**

**b. Во ServletB можеме да ги пристапиме Session атрибутите поставени во ServletA**

**c. Контејнерот ќе го повика сервлетот мапиран на servletB.do**

**d. Во ServletB можеме да ги пристапиме параметрите пратени до ServletA**

e. Клиентот ќе направи ново барање до /servletB.do

14. Селектирајте дали исказите опишуваат Attribute или Parameter во J2EE контејнерот.

Тип String **Parameter**

Може да се променат во сервлетите **Attribute**

Не може да се променат во сервлетите **Parameter**

Тип Object **Attribute**

Овозможуваат структурирање на кодот според MVC шаблонот **Attribute**

15. Што од следното е валидно за состојба (state) во React.JS?

Select one or more:

a. Менувањето на вредностите може да се направи директно со доделување на вредност на објектот this.state надвор од конструкторот.

**b. Менувањето на вредностите може да се направи директно со користење на функцијата this.setState() надвор од конструкторот.**

c. Вредностите кои веќе еднаш се поставени во state на компонента, не може да се менуваат.

**d. Вредностите кои веќе еднаш се поставени во state на компонента, може да се менуваат.**

e. Во рамките на конструкторот на компонентата не може да се прави доделување на вредности на објектот this.state

**f. Во рамките на конструкторот на компонентата може да се прави доделување на вредности на објектот this.state**

16. Кој од понудените опции прикажува валидна синтакса на функцијата decrement за асинхроно намалување на вредноста за 1 на бројачот во променливата counter сместена во состојбата на компонентата при клик на копчето Decrement? Почетната вредност на бројачот е 100.

render() функцијата изгледа вака:

```
render() {
    return (
        <div className="text-center">
            <label className="control-label">Counter value</label>
            <h3>{this.state.counter}</h3>
            <button className="btn btn-primary" onClick={this.decrement}>Decrement</button>
        </div>
    )
}
```

Функционалноста на компонентата е дадена во следните 2 слики:

counter value 100 decrement

По клик на копчето Decrement, потребно е да го даде следниот излез:

counter value 99 decrement

Select one:

a.

```
decrement => {
    this.setState((state)=> ({
        counter:state.counter-1
    }));
}
```

**b.**

```
decrement = (e) => {
    this.setState((state)=> ({
        counter:state.counter-1
    }));
}
```

c.

```
decrement {
    this.setState((state)=> ({
        counter:state.counter-1
    }));
}
```

d.

```
decrement() {
    this.state = {
        counter:this.state.counter-1
    }
}
```

e.

```
function decrement(e) {
    this.state.counter--;
}
```

17. Потребно е да се имплементира форма за додавање на нов вработен во некој систем за менаџирање на вработени во рамките на една компанија. Податоците за новиот вработен се додаваат преку соодветна форма. За секој вработен треба да знаеме име, презиме и работен стаж. Сите податоци се внесуваат преку текстуални полиња. Внесувањето на нов вработен се прави со клик на копчето Додади. Податоците за вработените се чуваат во соодветна JSON листа во состојбата на компонентата. Додавањето на нов вработен во листата, треба да биде асинхроно.

Кодот на целосната компонента EmployeeComponent изгледа вака:

```
import React, {Component} from 'react'

class EmployeeComponent extends Component {
    constructor(props) {
        super(props);
        this.state = {
            employees: [
                {
                    name: "Petre",
                    surname: "Petrov",
                    workExperience: 34
                },
                {
                    name: "Petar",
                    surname: "Petrov",
                    workExperience: 12
                }
            ]
        }
    }

    handleSubmit = ?

    render() {
```

```
    let employeesList = this.state.employees.map((item, index) => (
        <li>{item.name}</li>
    ));

    return (
        <div className="container-fluid">
            <ul>
                {employeesList}
            </ul>
            <form onSubmit={this.handleSubmit}>
                <div className="wrapper">
                    <div className="row">
                        <div className="col-md-12">
                            <label className="control-label">Name:</label>
                            <input name={"name"} type={"text"}
                                   className="form-control"/>
                        </div>
                        <div className="col-md-12">
                            <label className="control-label">Surname:</label>
                            <input name={"surname"} type={"text"}
                                   className="form-control"/>
                        </div>

                        <div className="col-md-12">
                            <button type={"submit"} className="btn btn-success">Submit</button>
                        </div>
                    </div>
                </div>
            </form>
        </div>
    )
}
```

```
}
```

export default EmployeeComponent;

Кој од следните имплементации нуди синтаксички валиден код за функцијата handleSubmit кој е соодветен во веќе приложениот код?

Select one:

a.

```
handleSubmit(e) {
    e.preventDefault();
    const employee = {
        name: e.target.name.value,
        surname: e.target.surname.value
    }
    this.setState((state) => {
        return {
            "employees":
                [state.employees,employee]
        }
    });
};
```

b.

```
handleSubmit = (e) => {
    e.preventDefault();
    const employee = {
        name: e.target.name.value,
        surname: e.target.surname.value
    }
    this.setState((state) => {
        return {
            "employees":
                [...state.employees,employee]
```

```
        }
    });
};
```

**c.**

```
handleSubmit = (e) => {
    e.preventDefault();
    let employee = {
        name: e.target.name.value,
        surname: e.target.surname.value
    }
    this.setState({
            "employees":
                [...state.employees,employee]
        }
    );
};
```

d.

```
handleSubmit(e) {
    e.preventDefault();
    let employee = {
        name: e.target.name.value,
        surname: e.target.surname.value
    }
    this.setState((state) => {
        return {
            "employees":
                [...state.employees,employee]
        }
    });
};
```

18. Избери која од понудените опции дава валидна синтакса во ReactJS за прикажување на детали за предмет во една студиска програма. Името на компонентата е Course. Податоците за курсот се предаваат како својство на следниот начин:

```
let obj = {
     id:1,
    courseName:"WP",
    courseFund:"2+2+1"
};
<Course course={obj}/>
```

Select one:

a.

```
import React, {Component} from 'react'

class Course extends Component {

   render() {
      return (
         <li key="{this.props.course.id}" className="list-group-item">
            <div >
               <div className="row">
                  <div className="col-md-2">
                     {this.course.id}
                  </div>
                  <div className="col-md-6">
                     {this.course.courseName}
                  </div>
                  <div className="col-md-4">
                     {this.course.courseFund}
                  </div>
               </div>
```

```
            </div>
        </li>
      )
   }
}

export default Course
```

b.

```
import React, {Component} from 'react'

class Course extends Component {

   return() {
      render(
         <li className="list-group-item">
            <div >
               <div className="row">
                  <div className="col-md-2">
                     {props.course.id}
                  </div>
                  <div className="col-md-6">
                     {props.course.courseName}
                  </div>
                  <div className="col-md-4">
                     {props.course.courseFund}
                  </div>
               </div>
            </div>
         </li>
      )
   }
}
```

export default Course

**c.**

```
import React, {Component} from 'react'

class Course extends Component {
    constructor(props) {
        super(props);
    }
    render() {
        return (
            <li key="{this.props.course.id}" className="list-group-item">
                <div >
                    <div className="row">
                        <div className="col-md-2">
                            {this.props.course.id}
                        </div>
                        <div className="col-md-6">
                            {this.props.course.courseName}
                        </div>
                        <div className="col-md-4">
                            {this.props.course.courseFund}
                        </div>
                    </div>
                </div>
            </li>
        )
    }
}

export default Course
```

d.

```
import React, {Component} from 'react'
```

```
class Course extends Component {

    render {
        return (
            <li className="list-group-item">
                <div >
                    <div className="row">
                        <div className="col-md-2">
                            {this.props.id}
                        </div>
                        <div className="col-md-6">
                            {this.props.courseName}
                        </div>
                        <div className="col-md-4">
                            {this.props.courseFund}
                        </div>
                    </div>
                </div>
            </li>
        )
    }
}

export default Course
```

19. Која од понудените компоненти користи валидна синтакса за креирање на едноставна компонента WelcomeComponent за приказ на следниот HTML код:

```
<div>
   <b>Welcome</b>
</div>
```

Select one or more:

a.

```
import React, {Class} from 'react'

class WelcomeComponent extends Class {
   return() {
      const message = <h3>Welcome</h3>
      render(
         <span>
            {message}
         </span>
      )
   }
}
export default WelcomeComponent;
```

b.

```
import React from 'react'

class WelcomeComponent extends React.Component {
   render() {
      const message = <span>Welcome</span>
      return (
         <div>
            {message}
         </div>
      )
```

```
    }
}
export default WelcomeComponent;
```

**c.**

```
import React from 'react';

const welcomeComponent = (props) => {
    let msg = <b>Welcome</b>
    return (
        <div>
            {msg}
        </div>
    )
}
export default welcomeComponent;
```

d.

```
import React from 'react';

const welcomeComponent = () => {
    let msg = <b>Welcome</b>
    render (
        <div>
            {{msg}}
        </div>
    )
}
export default welcomeComponent;
```

20. Поврзете ги настаните од животниот циклус на Bean-овите во Spring Container-от.

**4** Повик на методот setBeanFactory на имплементациите на BeanFactoryAware интерфејсот

**2** Поставување на својствата (Populate Properties)

**8** Повик на иницијализациските методи (@PostConstruct)

**11** Извршување на destroy од имплементацииите на DisposableBeans интерфејсот

**7** Извршување на afterPropertiesSet од имплементацииите на InitializingBean интерфејсот

**12** Повик на методите за уништување на bean-овите (@PreDestroy)

**1** Иницијализација (Initialize)

**10** Bean-от е креиран и постои во контејнерот

**3** Повик на методот setBeanName на имплементациите на BeanNameAware интерфејсот

**5** Повик на методот setApplicationContext на имплементациите на ApplicationContextAware интерфејсот

**9** Извршување на postProcessAfterInitialization од имплементацииите на BeanPostProcessor интерфејсот

**6** Извршување на postProcessBeforeInitialization од имплементацииите на BeanPostProcessor интерфејсот