Spring Web Developer - Assessment

1. Consider below **GreetingService class**.

```
package com.pack.service;
public class GreetingService {
   private String greeting ;
   public String getGreeting() {
      return greeting;
   }
   public void setGreeting(String greeting) {
      this.greeting = greeting;
   }
}
```

A Spring configuration file(config.xml) defining bean of the GreetingService class is shown below:

```
<bean id = "greetingService" class = "com.pack.service.GreetingService"  />
```

The main method of the application includes the following statements:

```
ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");

GreetingService greetingService = (GreetingService) context.getBean("greetingService");
greetingService.setGreeting("Hi, Welcome to Spring Course!!");
System.out.println(greetingService.getGreeting());
```

What will happen when we execute the above main method?

a. Displays the greeting message "Hi, Welcome to Spring Course! !"
b. Displays null as the greeting property is not initialized
c. POJO class setter method cannot be invoked on the bean
d. None of the above

2. Consider MobileService class with two properties **mobileNumber** and **model** as shown below.

```
package com.pack;
public class MobileService{
   private long mobileNumber;
   private String model;

   //parameterized constructor
   public void MobileService(long mobileNumber, String model){
         this.mobileNumber = mobileNumber;
         this.model = model;
   }

}
```

A Spring configuration file defining a bean of the MobileService class is shown below:

```
<bean id="mobileService" class="com.pack.MobileService"/>
```

The main method for the application includes the following instructions:

```
ApplicationContext ac=new ClassPathXmlApplicationContext("config.xml");
MobileService m = (MobileService)ac.getBean("mobileService");
```

What will happen when we execute the above main method?

a. The default constructor of the MobileService class will be called to create instance and initialize with the default values as mobileNumber with 0 and model with null.
b. Throws an exception as no values are passed to the parameterized constructor using constructor-arg tag in the bean definition
c. Throws an exception: no matching constructor found
d. None of the above

3. Consider the following classes defined in a Spring application:

```
package com.pack.service;
public class MobileService {
   private String mobileName;
   public MobileService() {    }
   //getter and setter methods
}
package com.pack.service;
public class PersonService {
    private MobileService mob;
     //getter and setter methods
}
```

The Spring configuration file includes the following declarations:

```
<bean id="mobileService" class="com.pack.service.MobileService">
     <property name="mobileName" value="Nokia"/>
</bean>
<bean id="personService" class="com.pack.service.PersonService" autowire="byName" />
```

The main method for the application is defined as follows:

```
ApplicationContext ac= new ClassPathXmlApplicationContext(“config.xml");
PersonService p1= (PersonService) ac.getBean("personService");
System.out.println("I have : "+p1.getMob().getMobileName());
```

What is the output of above code execution?

a. Throws NullPointerException
b. Display mobile name as “Nokia”
c. Display mobile name as null
d. None of the above

4. Consider below code.

```
package com.pack.service;
public class MobileService{
   private long mobileNumber;
   private String model;
   //default constructor
   //getter and setter methods
}
package com.pack.service;
public class ShopService {
     private String location;
     private MobileService mobileService;
     //default constructor
     //getter and setter methods
     @Autowired
     public void setMobileService(MobileService mobileService){
          this.mobileService = mobileService;
     }
}
```

A Spring configuration file defines the beans of MobileService and ShopService classes as shown below:

Assume the required namespaces declaration is present in the configuration file.

```
<context:annotation-config />
<bean id="shopService" class="com.pack.service.ShopService">
     <property name="location" value="Pune"/>
</bean>
<bean  id="m1" class="com.pack.service.MobileService">
     <property name="mobileNumber" value="201"/>
     <property name="model" value="Nokia"/>
</bean>
<bean  id="m2" class="com.pack.service.MobileService">
     <property name="mobileNumber" value="202"/>
     <property name="model" value="Samsung"/>
</bean>
```

The main method of the application is shown below:

```
ApplicationContext ac=new ClassPathXmlApplicationContext("config.xml");
ShopService shop = (ShopService)ac.getBean("shopService");
```

What is the cause of a runtime exception when the main method is executed?

a. @Autowired annotation wires dependency based on type of bean and hence @Qualifier("beanname") must be used to specify bean name along with @Autowired when multiple beans of MobileService class are present in the configuration
b. Parametarize constructor is required in ShopService class for wiring the dependency
c. @Autowired annotation is not allowed to use on setter methods
d. None of the above

5. Consider the below given code snippets:

```
package com.pack;
```

```
@Aspect
public class MyAspect {
    @Before("execution( * com.pack.AdderService.add(int,int))")
     public void logBeforeExecutionAdvice(){
                System.out.print(" within Before Advice ");
    }
}
package com.pack;
public class AdderService {
       private int a;
       private int b;
       public int add(int a,int b){
                    return (a+b);
        }
       public int add(int a){
                    return (a+10);
        }
      //getter and setter methods
}
package com.pack;
public class Main {
       public static void main(String[] args) {
          ApplicationContext ctx=new ClassPathXmlApplicationContext("config.xml");
          AdderService adder=(AdderService)ctx.getBean("adderService");
          System.out.print("Result=" + adder.add(2));
      }
}
```

Spring configuration file is as below:

```
<beans>
<aop:aspectj-autoproxy/>
<bean id="myAspect" class="com.pack.MyAspect"/>
<bean id="adderService" class="com.pack.AdderService" />
</beans>
```

What is the output of the above code execution?

a. Result=12 within Before Advice
b. Result=12
c. Runtime Exception
d. within Before Advice Result=12

HIBERNATE

1. Given the below entity, identify the correct annotation to map the dateOfBirth property to compatible database type.

```
public class Student{
  @Id
```

```
  private Integer rollNo;
  private Calender dateOfBirth;
  // getter and setter methods
}
```

a. @DateAndTime
b. @Transient
c. @Temporal
d. @Date

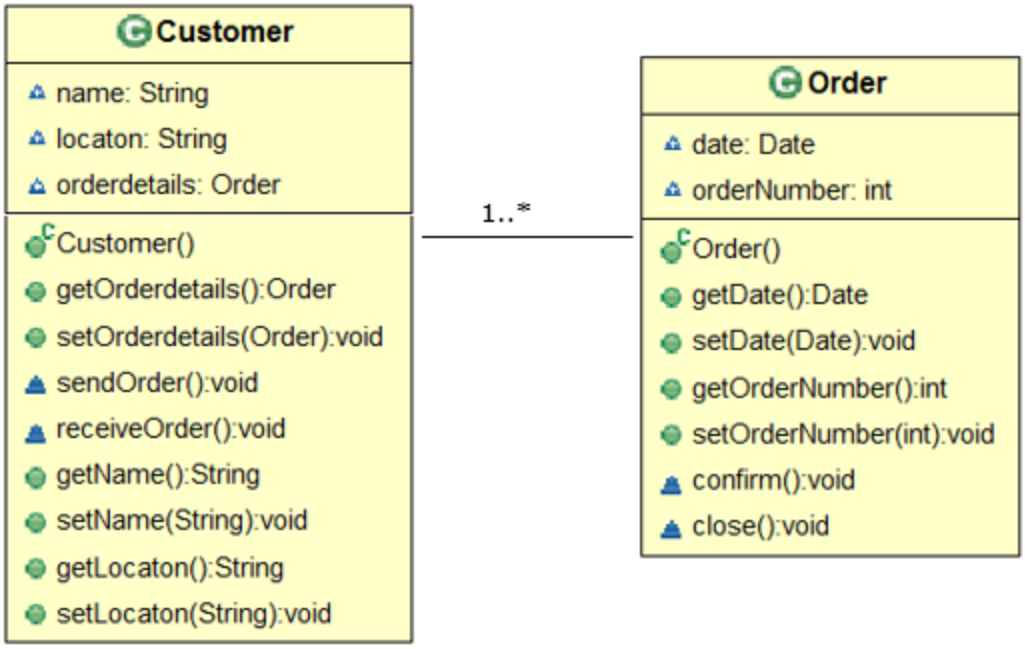
2. Read the below code snippet and identify the different states of Student entity in hibernate.

```
Student entity = new Student();
entity.setRollNo(202);
entity.setStudentName("Anandi");
SessionFactory factory = configuration.buildSessionFactory(serviceRegistry);
Session session = factory.openSession();
session.beginTransaction();
session.persist(entity);
session.getTransaction().commit();
session.close();
```

a. new,persistent,detached
b. new,transient,detached
c. transient,detached,persistent
d. None of the above

3. A developer who is working on an online shopping application, wants to store the details of a Customer object and Order object into a single database table called OrderDetails. The class diagram is given below.

   Identify the paradigm mismatch occurs?



a. Problem of Granularity
b. Problem of Subtype
c. Problem of Association
d. Problem of Identity

4. Identify the correct key word to fill in the query mentioned below for the place holders 1& 2 to sort the results by descending order.

```
String hql = "FROM Employee employee WHERE employee.empid > 3779 1 employee.salar
y 2";

Query query = session.createQuery(hql);

List results = query.list();
```

a. 1- GROUP BY , 2- DESC
b. 1- ORDER BY, 2 - DESC
c. 1- ORDER BY , 2 - DSC
d. 1- GROUP BY, 2- DSC

5. Which of the following will be used to load all the persistent objects of com.infy.Student Entity into Hibernate memory using HQL.

```
a). String hql = "FROM com.infy.Student";

Query query = session.createQuery(hql);

List results = query.list();
```

```
b). String hql = "All from com.infy.Student";
Query query = session.createQuery(hql);
List results = query.list()
```

```
c). String hql = "select s from com.infy.Student s";
Query query = session.createHQLQuery(hql);
List results = query.list()
```

```
d). String hql = "select all from com.infy.Student";
Query query = session.createQuery(hql);
List results = query.list()
```

a. a and c are correct
b. Only b is correct
c. All are correct
d. Only a is correct

Spring Data JPA

1. Development team is planning to use Spring ORM integration with Java Persistence API. In this process, the team works on configuring the details to enable this integration. Following is the package structure of the application.

   Entity classes with mapping details through annotations in **com.pack.entity**

   Repository classes in **com.pack.repository**

   Service classes in **com.pack.service**

   Observe the below given entityManagerFactory bean definition present in the Spring configuration.

```
<bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainerEnti
tyManagerFactoryBean">
        <property name="packagesToScan" value="__________" />
        <property name="dataSource" ref="myDataSource" />
        <property name="jpaVendorAdapter" ref="hbAdapterBean"></property>
```

```
</bean>
```

Assume referenced beans myDataSource and hbAdapterBean are present in the configuration.

Identify the RIGHT package value to be specified for **packagesToScan** property.

a. com.pack.repository
b. com.pack.entity
c. com.pack.service
d. None of the above

2. Identify which among the following are correct usages of @Transactional annotation.

```
I. public class ApplicantOperations {
           @Transactional
           public  void applicantSuccess(){
                 //code goes here
           }
    }
II. @Transactional
   public class ApplicantOperations {
          public  void applicantSuccess(){
                  //code goes here
          }
          public  void applicantFailure(){
                  //code goes here
          }}
III.   @Transactional
     public  interface applicantData{
             public void applicantSuccess();
             public void applicantFailure();
    }
IV.   @Transactional
     public  class applicantData{
```

```
            public  void applicantSuccess(){
                   //code goes here
           }
          @Transactional(readOnly=true)
          public  void applicantFailure(){
                  //code goes here
         }
}
```

a. Only I
b. Only II
c. Only III
d. Only IV

3. Consider a class called “ApplicationConfig”, a Java based configuration file used to configure the beans and other configuration related details as shown below:

```
@Configuration
public class ApplicationConfig{
       @Bean
       public  DataSource dataSource(){
       }
      //rest other beans configured}
```

Which of the following annotation enables Spring’s annotation driven transaction management capability?

a. @EnableTransactionManagement
b. @Repository
c. @EnableTransaction
d. @Transactional

4. Consider the below given entity class in Employee management application.

```
@Entity
@Table(name="EmployeeTable")
```

```
public class Employee {
       @Id
       private long employeeId;
       @Column(name="EmailAddress")
       private String emailAddress;
                //getter and setter methods
}
```

Development team would like to query employee record based on email address details.

Help the team to know the different possible approaches available in Spring Data JPA for query operation by choosing RIGHT options.

```
I.
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
        @Query("select emp from Employee emp where emp.emailAddress = ?1")
        Employee  getEmployee(String emailAddress);
}
```

```
II.
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
        @Query("select * from EmployeeTable where EmailAddress = ?", nativeQuery="tru
e")
         Employee getEmployee(String emailAddress);
}
III.
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
         Employee  findByEmailAddress(String emailAddress);
}
```

Provide query at class level using @NamedQuery as shown below

```
IV.
@Entity
@Table(name="EmployeeTable")
@NamedQuery(name = "Employee.getEmployee", query = "select emp from Employee emp wher
e emp.emailAddress = ?1")
```

```
public class Employee {
    ------------------
}
```

```
public interface EmployeeRepository extends JpaRepository<Employee, Long> {
    Employee getEmployee(String email);
}
```

a. Only (i) and (ii)
b. Only (iii)
c. Only (iv)
d. All the given options

5. InfyData team needs to perform Spring ORM integration with Hibernate. Which of the following has to be defined in the Spring configuration file?
    a. Define Data Source
    b. Define SessionFactory bean with required properties set
    c. Define EntityManagerFactory bean with required properties set
    d. Define Repository beans

6. TechData back end team is planning to use Spring ORM integration with Java Persistence API. In this process, the team works on configuring the details to enable this integration. Observe the below bean definitions in the configuration file and help the team choose valid statements that can help in the proper configuration support.

```
<bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainer
EntityManagerFactoryBean">
        <property name="packagesToScan" value="com.pack.domain" />
        <property name="dataSource" ref="myDataSource" />
        <property name="jpaVendorAdapter" ref="hbAdapterBean"></property>
</bean>


 <bean id="myDataSource" class="org.springframework.jdbc.datasource.DriverManager
DataSource">
        <property name="driverClassName" value="org.h2.Driver"></property>
        <property name="url" value="jdbc:h2:tcp://localhost/~/test"></property>
```

```
        <property name="username" value="sa"></property>
        <property name="password" value=""></property>
</bean>

<bean id="hbAdapterBean" class="org.springframework.orm.jpa.vendor.HibernateJpaVe
ndorAdapter">
        <property name="generateDdl" value="true"></property>
        <property name="databasePlatform" value="org.hibernate.dialect.H2Dialect"
></property>
</bean>
```

a. “hbAdapterBean” bean is to provide Hibernate as the JPA provider with appropriate Hibernate/JPA properties and dialect details.
b. “hbAdapterBean” bean is to provide required mapping details.
c. "myDataSource" bean is to provide current database connection details
d. packagesToScan attribute of "entityManagerFactory" is to provide mapping details

7. Development team wants to implement data access layer of retail application using Spring Data JPA. Team would like to implement pagination to display 5 customer records from the first page of query result. Identify the right option from the below given statements.
   a. Pageable pageable = new PageRequest(1, 5);

      Page<Customer> customers = customerRepository.findAll(pageable);

   b. Pageable pageable = new PageRequest(0, 5);

      Page<Customer> customers = customerRepository.findAll(pageable);

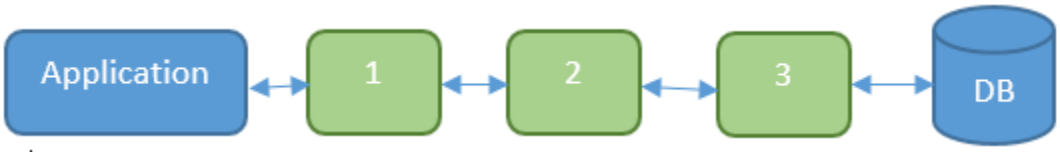
   c. Pageable pageable = new PageRequest(1,4);

      Page<Customer> customers = customerRepository.findAll(pageable);

   d. Pageable pageable = new PageRequest(0,4);

      Page<Customer> customers = customerRepository.findAll(pageable);

8. An application data access layer has been developed using Spring Data JPA. Identify from the given options the values for layers 1-3 in the diagram?



   a. JPA Provider, JpaRepository, JDBC API

b. JpaRepository, JPA Provider, JDBC API
c. JpaRepository, JDBC API, JPA Provider
d. JDBC API, JpaRepository, JPA Provider

9. Which of the following minimum artifact/s would be required to implement data access layer to perform insert and remove database operation on Customer entity using Spring Data JPA?
   a. CustomerRepository interface
   b. CustomerRepositoryImpl implementation
   c. CustomerRepository interface and CustomerRepositoryImpl implementation
   d. CustomerDAO interface, CustomerRepository interface and CustomerRepositoryImpl implementation

10. In Spring Data JPA, which of the following annotation can be used to bind names to positional parameters in the query?
    a. @Query
    b. @NamedParam
    c. @Param
    d. @NamedQuery

Spring MVC

1. Which part of the below code snippet defines the predefined abstract method and writes the pdf document to the response?

```
public abstract class AbstractITextPdfView extends AbstractView {

    public AbstractITextPdfView() {

        setContentType("application/pdf");

    }

    protected void buildPdfMetadata(Map<String, Object> model, Document document,
HttpServletRequest request) {}


    protected abstract void buildPdfDocument(Map<String, Object> model,
                                             Document document,
                                             PdfWriter writer,
                                             HttpServletRequest request,
                                    HttpServletResponse response) throws Exception;
```

```
    @Override
    protected void renderMergedOutputModel(Map<String, Object> model,
                                           HttpServletRequest request,
                                           HttpServletResponse response) throws E
xception {


        ByteArrayOutputStream out = createTemporaryOutputStream();
        Document document = new Document(PageSize.A4);
        PdfWriter writer = PdfWriter.getInstance(document, out);
        writer.setViewerPreferences(PdfWriter.ALLOW_PRINTING | PdfWriter.PageLayo
utSinglePage);
        buildPdfMetadata(model, document, request);


        document.open();
        buildPdfDocument(model, document, writer, request, response);
        document.close();
        writeToResponse(response, out);
    }
    @Override
    protected boolean generatesDownloadContent() {
        return true;
    }
}
```

```
i.  protected void buildPdfMetadata(Map<String, Object> model, Document document,
HttpServletRequest request) {}
```

```
ii. @Override
     protected void renderMergedOutputModel(Map<String, Object> model,
                                           HttpServletRequest request,
                                  HttpServletResponse response) throws Exception {
       ByteArrayOutputStream out = createTemporaryOutputStream();
       Document document = new Document(PageSize.A4);
```

```
        PdfWriter writer = PdfWriter.getInstance(document, out);
        writer.setViewerPreferences(PdfWriter.ALLOW_PRINTING |
                                    PdfWriter.PageLayoutSinglePage);
        buildPdfMetadata(model, document, request);
        document.open();
        buildPdfDocument(model, document, writer, request, response);
        document.close();
        writeToResponse(response, out);
    }
```

```
iii.  protected boolean generatesDownloadContent() {
              return true;
    }
```

```
iv. protected abstract void buildPdfDocument(Map<String, Object> model,
                                             Document document,
                                             PdfWriter writer,
                                             HttpServletRequest request,
                                      HttpServletResponse response) throws Exception;
```

a. Only (i)
b. Only (ii)
c. Only(iii)
d. Only(iv)

2. Sharon has undergone a training on Spring MVC. She was asked to predict the output for the below code snippet.

```
@Controller
public class MyController {
     @RequestMapping("/hello.htm")
     public ModelAndView sayGreeting(){
         String msg="Hi, Welcome to Spring MVC 3.2";
         return new ModelAndView("WEB-INF/jsp/hello.jsp","message","msg");//Line6
```

```
    }
}
```

a. Hi, Welcome to Spring MVC 3.2
b. msg
c. Compilation fails at Line6
d. Runtime exception will be thrown at Line6

3. Shreya has designed a POJO class as below and tries the code snippet without providing an input for name and pwd. What could be the output for it?

```
public class User {
  @NotEmpty(message = "Username must not be blank.")
  private String name;
  @NotEmpty(message = "Password must not be blank.")
  @Size (min=8, max=10, message = "Password must be between 8 to 10 Characters."
//Line5
  private String pwd;
  // Getter and setter methods
}
```

a. Username must not be blank.
   Password must not be blank
b. Compilation fails at Line5. Message cannot be overridden.
c. Nothing will be displayed
d. Username must not be blank.

   Password must not be blank

   Password must be between 8 to 10 Characters

4. Which of the following code is required in the Spring configuration file during internationalization and localization?

```
i. <bean id="localeResolver"
        class="org.springframework.web.servlet.i18n.SessionLocaleResolver">
        <property name="defaultLocale" value="en" />
   </bean>
```

```
ii. <bean id="localeChangeInterceptor"
```

```
        class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
        <property name="paramName" value="language" />
    </bean>
```

```
iii. <bean class="org.springframework.web.servlet.mvc.support.ControllerClassNameHandlerMapping" >
        <property name="interceptors">
           <list>
                <ref bean="localeChangeInterceptor" />
           </list>
        </property>
     </bean>
```

a. Only (i)
b. Only (ii)
c. Both (ii) and (iii)
d. (i), (ii) and (iii)

5. Levin is working on Spring MVC project involving dynamic URI. Help him to identify the output of the below code when user clicks on book1.

   Note: Prefix and suffix properties are included in the dispatcher-servlet.xml file.

index.jsp

```
<body>
<a href = "book/book1.htm?show=yes">book1</a>
<a href = "book/book2.htm?show=yes">book2</a>
<a href = "book/book3.htm?show=yes">book3</a>
</body>
```

MyController.java

```
@Controller
public class MyController
{
@RequestMapping(value="/book/{item}", params="show=yes", method=RequestMethod.GET)
```

```
public String processBooks(@PathVariable("item") String item){
```

```
  System.out.println("You have selected the book:" + item);
  return "success.jsp";
}}
```

success.jsp

```
<body>
  <h1>Success Page</h1>
</body>
```

a. You had selected the book: book1

   Success Page

b. 404 error. Unable to find success.jsp
c. Compilation fails
d. Success Page

6. Leon was asked to store the model attributes of a Spring MVC application in session so that they could span across the user requests. Which of the following code has to be used by him?

```
i. @Controller
   @RequestMapping("/login.htm")
    public class LoginController {
         @SessionAttributes("user");
}
```

```
ii. @Controller
    @RequestMapping("/login.htm")
    public class LoginController {
          @Session("user");
    }
```

```
iii. @Controller
```

```
    @SessionAttributes("user")
    @RequestMapping("/login.htm")
    public class LoginController {
              // Insert code here
  }
```

```
iv. @Controller
    @SessionParam("user")
    @RequestMapping("/login.htm")
     public class LoginController {
              // Insert code here
     }
```

a. Only(i)
b. Only(ii)
c. Only(iii)
d. Only(iv)

7. While developing a Spring web application with Spring Rest, Prem was asked to place a composed annotation which can also act as a short cut for @RequestMapping(method=RequestMethod.POST) to map the http requests to Spring Controller methods. Which of the following annotation could help here?

```
i. @GetMapping("/get/{id}")
ii. @PostMapping("/post/{id}")
iii. @PutMapping("/put/{id}")
iv. @DeleteMapping("/delete/{id}")
```

a. Only (i)
b. Only(ii)
c. Only(iii)
d. Only(iv)

8. Suja was asked to write code for Star Solutions, that will add some attributes like technologies list and the reference of User class to ModelMap while rendering register.jsp page.Which of the following code could help her?

```
i.   @RequestMapping(method = RequestMethod.GET)
   public String showForm(ModelMap m) {
```

```
     Map<String,String> technologyList=new HashMap<String,String>();
     technologyList.put("J2EE", "J2EE");
     technologyList.put("DotNET", ".NET");
     m.addAttribute("technologyList",technologyList);
     User user=new User();
     m.addAttribute("user", user);
     return "/WEB-INF/jsp/register.jsp";
   }
ii. @RequestMapping(method = RequestMethod.GET)
    public String showForm(ModelMap m) {
     Map<String,String> technologyList=new            HashMap<String,String>();
     technologyList.put("J2EE", "J2EE");
     technologyList.put("DotNET", ".NET");
     m. addParam ("technologyList",technologyList);
     User user=new User();
     m.addParam("user", user);
     return "/WEB-INF/jsp/register.jsp";
    }
```

a. Only (i)
b. Only (ii)
c. Both (i) and (ii) are valid
d. Both (i) and (ii) are invalid

9. Riya has a Java configuration file. And she wants to import the Spring MVC configuration into the Java configuration file. Which of the following code snippet has to be used by her?

```
i. @Configuration
   @EnableWebMvc
   public class WebConfig
   {        ...      }
```

```
ii. @Configuration
    @EnableMvc
     public class WebConfig
```

```
     {          ...      }
```

```
iii. @Configuration
     @ComponentScan
     public class WebConfig
     {          ....         }
```

```
iv. @EnableWeb
    public class WebConfig
    {         ....           }
```

a. Only (i)
b. Only (ii)
c. Only (iii)
d. Only (iv)

10. Can we have more than one configuration file in a Spring MVC application?

```
i. Yes, By specifying all the Spring configuration files in web.xml file using
contextConfigLocation init parameter.
   <servlet>
        <servlet-name>springConfig</servlet-name>
        <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
        </servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>
                WEB-INF/spring-basics.xml,
                WEB-INF/spring-mvc.xml,
                WEB-INF/spring-dao.xml
            </param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>springConfig</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
```

ii. Yes, By importing all the Spring configuration files into an existing configuration file that has been configured already.

```
<beans>
  <import resource="spring-dao-hibernate.xml"/>
  <import resource="spring-services.xml"/>
  <import resource="spring-security.xml"/>
</beans>
```

a. No
b. Option (i) is valid
c. Both option (i) and (ii) are valid
d. Option (ii) is valid