

Spring Certification Questions

Free on-line [spring certification test available at https://www.springmockexams.com](https://www.springmockexams.com)

Enrol now to get full access to a set of four Spring Mock Exams using our exam simulator.

ENROLLING NOW YOU WILL GET ACCESS TO [200 UNIQUE SPRING CERTIFICATION QUESTIONS.](#)

QUESTION 1

<https://www.springmockexams.com/spring/mock/spring-mock-exam.html>

Spring Certification Question: Which of the following is true regarding the @Autowired annotation?

Select Your Answer:

- A: It is possible to provide all beans of a particular type from the ApplicationContext by adding the annotation to a field or method that expects an array of that type.
- B: Typed Maps can be autowired as long as the expected key type is String.
- C: By default, the autowiring fails whenever zero candidate beans are available.
- D: All of the above.

The answer is: D

Explanation:

```
A: public class MovieRecommender {  
  
    @Autowired  
    private MovieCatalog[] movieCatalogs;  
  
    // ...  
}
```

```
B: public class MovieRecommender {
```

```
private Map<String, MovieCatalog> movieCatalogs;

@Autowired
public void setMovieCatalogs(Map<String, MovieCatalog> movieCatalogs) {
    this.movieCatalogs = movieCatalogs;
}

// ...
}
```

C: This is true. If no candidate are available, an exception will be thrown.

D: They are all true.

QUESTION 2

Spring Certification Question: By default, when you use `XmlWebApplicationContext`, the configuration will be taken from `"/WEB-INF/applicationContext.xml"` for the root context, and `"/WEB-INF/test-servlet.xml"` for a context with the namespace `"test-servlet"`. Which of those pieces of code can override the default config location?

Select all that apply:

A:

```
<servlet>
    <servlet-name>accounts</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/mvc-config.xml
        </param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
```

B:

```
<context-param>

</context-param>
<context-param>
    <param-name>spring.profiles.active</param-name>
    <param-value>jpa</param-value>
</context-param>
```

C:

```
<listener>
    <listener-class>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/app-config.xml</param-value>
    </listener-class>
```

</listener>

D: None of the above

The answers are: A, B

Explanation:

A: The config location defaults can be overridden via the "contextConfigLocation" context-param of ContextLoader.

B: The config location defaults can be overridden via the servlet init-param of FrameworkServlet.

C: The listener section is used to define a ContextLoaderListener and it does not impact the application context location.

D: A and B are true.

QUESTION 3

Spring Certification Question: Which are valid method signatures for the method ClassPathXmlApplicationContext.getBean():

Select all that apply:

A: Object getBean(String name) throws BeansException.

B: <T> T getBean(String name, Class<T> requiredType) throws BeansException.

C: <T> T getBean(String name, String requiredType) throws BeansException.

D: <T> T getBean(Class<T> requiredType) throws BeansException.

E: All of the above.

The answers are: A,B,D

Explanation:

A: Return an instance, which may be shared or independent, of the specified bean name.

B: Behaves the same as getBean(String), but provides a measure of type safety by throwing a BeanNotOfRequiredTypeException if the bean is not of the required type.

C: This method signature does not exist.

D: Return the bean instance that uniquely matches the given object type, if any.

E: Only A,B,D are true.

QUESTION 4

Spring Certification Question: Which of these is the best description of an AOP Aspect?

Select Your Answer:

A: A point in the execution of a program such as a method call or field assignment.

B: An expression that Selects one or more Join Points.

- C: Code to be executed at a Join Point that has been Selected by a Pointcut.
- D: A module that encapsulates pointcuts and advice.
- E: None of the above.

The answer is: D

Explanation:

- A: This is a Join Point.
- B: This is a Pointcut.
- C: This is an Advice.
- D: This is an Aspect.
- E: D is true.

QUESTION 5

Spring Certification Question: Which of the following are false regarding Spring AOP?

Select all that apply:

- A: It can advice any Join Points.
- B: Can only apply aspects to Spring Beans.
- C: Spring adds behaviour using dynamic proxies if a Join Point is declared on a class.
- D: If a Join Point is in a class with no interface, Spring will use CGLIB for weaving.
- E: CGLIB proxies can be applied to final classes or methods.

The answers are: A, C, E

Explanation:

- A: False. It can advice only public Join Points.
- B: True. This one of the limitation.
- C: False. Spring uses dynamic proxies if a Join Point is declared on an interface.
- D: True. CGLIB is used for weaving class aspects.
- E: False. It cannot be applied to final classes or methods.

QUESTION 6

Spring Certification Question: Which of the following is false regarding the following code and HttpInvokerProxyFactoryBean?

```
<bean id="httpInvokerProxy"  
class="org.springframework.remoting.httpinvoker.HttpInvokerProxyFactoryBean">
```

```
<property name="serviceUrl" value="http://remotehost:8080/remoting/AccountService"/>
<property name="serviceInterface" value="example.AccountService"/>
</bean>
```

Select one answer:

- A: This is client-side code.
- B: Spring will translate your calls to HTTP POST requests.
- C: HttpInvokerProxy uses Commons HttpClient by default.
- D: Spring will send HTTP POST request to the defined URL which is `http://remotehost:8080/remoting/AccountService`.
- E: The service URL must be an HTTP URL exposing an HTTP invoker service.

The answer is: C

Explanation:

By default, the `HttpInvokerProxy` uses the J2SE HTTP functionality, but you can also use the Commons `HttpClient` by setting the `httpInvokerRequestExecutor` property:

```
<property name="httpInvokerRequestExecutor">
  <bean
class="org.springframework.remoting.httpinvoker.CommonsHttpInvokerRequestExecutor"/>
</property>
```

This is a difficult question but we found something similar on the real exam.

QUESTION 7

Spring Certification Question: Which of the following is true regarding the annotation `@RequestParam` in the following piece of code:

```
@Controller
@RequestMapping("EDIT")
@SessionAttributes("site")
public class PetSitesEditController {

    // ...

    public void removeSite(@RequestParam("site") String site,
        HttpServletResponse response) {
        this.petSites.remove(site);
        response.setRenderParameter("action", "list");
    }

    // ...
}
```

Select all that apply:

- A: The `@RequestParam` annotation is used to extract a parameter from the HTTP response and bind them to a method parameter.
- B: The `@RequestParam` annotation can automatically perform type conversion.
- C: Parameters using this annotation are required by default.
- D: It differs from `@PathVariable` because with the latest you can extract value directly from the request URL using the URI Templates.
- E: All of the above.

The answers are: B, C, D

Explanation:

- A: False. The `@RequestParam` annotation is used to bind request parameters to a method parameter in your controller. This `@Controller` will be an entry point for mapping HTTP requests.
- B: This is true. In this case, a request of the form `http://localhost:8080/...../EDIT.html?site=xxx` would convert xxx in a string and assign it to the method parameter "site".
- C: This is true as well, but you can specify that a parameter is optional by setting `@RequestParam's` annotation's `required` attribute to false (e.g., `@RequestParam(value="id", required=false)`) making it optional.
- D: This is true as well. `@PathVariable` can take advantage from the usage of placeholders that will extract the method parameter directly from the request URL. For instance:

```
@Controller
@RequestMapping(value = "/pets/{petId}", method = RequestMethod.GET, produces="application/
json")
@ResponseBody
public Pet getPet(@PathVariable String petId, Model model) {
    // implementation omitted
}
```

E: Only B, C and D are true so this one is false.

QUESTION 8

Spring Certification Question: Which of the following are true regarding the following piece of code?

```
<tx:annotation-driven/>

<bean id="txManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource" ref="dataSource"/>
</bean>
<jdbc:embedded-database id="dataSource">
  <jdbc:script location="classpath:rewards/testdb/schema.sql"/>
  <jdbc:script location="classpath:rewards/testdb/test-data.sql"/>
</jdbc:embedded-database>
```

Select all that apply:

- A: It is declaring a container-managed datasource (via JNDI).

B: DataSourceTransactionManager it is a subclass of AbstractPlatformTransactionManager.

C: It is defining a Transaction manager with id txManager for supporting transaction management.

D: It is defining a bean Post-processor that proxies @Transactional annotated bean

<tx:annotation-driven/>.

E: None of the above.

The answers are: B, C, D

Explanation:

A: False, it is declaring a local datasource using the tags <jdbc:embedded-database> ... </jdbc:embedded-database>. The preceding configuration creates an embedded HSQL database populated with SQL from schema.sql and testdata.sql resources in the classpath. The database instance is made available to the Spring container as a bean of type javax.sql.DataSource. This bean can then be injected into data access objects as needed.

B: True. It is a PlatformTransactionManager implementation for a single JDBC DataSource. It binds a JDBC Connection from the specified DataSource to the current thread, potentially allowing for one thread-bound Connection per DataSource.

C: True, this is needed for providing Spring transaction support.

D: True. This is the most tricky question because the declaration of the bean post processor it is hidden in the tag <tx:annotation-driven/>. Remember that you can mark any method with the @Transactional annotation but the mere presence of the @Transactional annotation is not enough to activate the transactional behavior. <tx:annotation-driven/> element switches on the transactional behavior.

E: A is not true so this does not apply.

QUESTION 9

Spring Certification Question: The method "convertAndSend" of the JmsTemplate interface, it is used to send an object to a destination, converting the object to a JMS message. Which of the following are valid definitions of this method?

Select all that apply:

A: convertAndSend(Destination destination, Object message).

B: convertAndSend(Object message).

C: convertAndSend(String destinationName, Object message).

D: convertAndSend(Object message, Destination destination).

E: All of the above.

The answers are: A, B, C

Explanation:

A: Send the given object to the specified destination, converting the object to a JMS message with a configured MessageConverter.

B: Send the given object to the default destination, converting the object to a JMS message with a configured MessageConverter.

C: Send the given object to the specified destination, converting the object to a JMS message with a configured MessageConverter.

D: There is no such method definition. This is a compiler error.

E: Only A,B,C are true.

QUESTION 10

Spring Certification Question: When using JMX which one is false regarding the following piece of configuration?

```
<beans>
```

```
<bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
  <property name="beans">
    <map>
      <entry key="bean:name=testBean1" value-ref="testBean"/>
    </map>
  </property>
</bean>
```

```
<bean id="testBean" class="org.springframework.jmx.JmxTestBean">
  <property name="name" value="TEST"/>
  <property name="age" value="100"/>
</bean>
```

```
</beans>
```

Select all that apply:

A: The bean "exporter" will export a bean to the JMX MBeanServer.

B: "testBean" bean is exposed as an MBean under the ObjectName bean:name=testBean1.

C: The bean "exporter" can be lazily initialized.

D: By default, all public properties of the bean are exposed as attributes and all public methods are exposed as operations.

E: All of the above.

The answers are: C, E

Explanation:

A: This is exactly the aim of the exporter. The bean "testBean" will be exported as an MBean with name of "testBean1".

B: This is true. The key of each entry in the beans Map is used as the ObjectName for the bean referenced by the corresponding entry value by default.

C: This is false. Exporter bean must not be lazily initialized if the exporting is to happen. If you configure a bean with the MBeanExporter that is also configured for lazy initialization, then the MBeanExporter will not break this contract and will avoid instantiating the bean. Instead, it will register a proxy with the MBeanServer and will defer obtaining the bean from the container until the first invocation on the proxy occurs.

D: This is true. The default policy is to expose all properties and all public methods of the bean.

E: False. Only C is false of the above so this is false as well.