**Q.** Which of the following statement(s) is/are true?  
  
i. All <aop:\*> elements must be contained within <aop:config> element.  
ii. <aop:aspectj-autoproxy> elements enable annotation driven aspects using @AspectJ  
iii. Single aspect can be declared using <aop:aspect> element.  
iv. <aop:config> element must be contained within <aop:aspect> element.

**1)**. **Statements i , ii and iii are true.**

**2)**. Statements i and ii are true.

**3)**. **Statements ii and iii are true.**

**4)**. All the statements are true.

**5)**. ok

**Q.** What is role of Spring Framework?

**1)**. **Provides a comprehensive programming and configuration model for modern Java-based   
enterprise applications  on any kind of deployment platform.**

**2)**. **Builds on Spring MVC and allows implementing the  flows of a web application.**

**3)**. **Can be used to integrate with groovy for building high-productivity dynamic application.**

**4)**. **Used to simplify the development of big data applications.**

**Q.** Examine the below code:  
  
@Controller  
@RequestMapping("/employee")  
public class EmployeeController {  
 @RequestMapping(value = "/save", method = RequestMethod.POST)  
 public ModelAndView saveEmployee(@ModelAttribute("employee") Employee  employee,  
   BindingResult result) { }}  
  
Which URL request will invoke the saveEmployee method in the above controller?   
Consider the application name is EmployeeMaintenance.

**1)**. **http://localhost:8080/EmployeeMaintenance/employee/save.obj**

**2)**. http://localhost:8080/EmployeeMaintenance/employee.obj

**3)**. http://localhost:8080/EmployeeMaintenance/save.obj

**4)**. **http://localhost:8080/EmployeeMaintenance/employee?value="save.obj"**

**Q.**A developer need to return product object from controller to a view page "DisplayProduct.jsp".   
Which code snippet inclusion in Controller will be best suitable?  
Consider the below Product Spring bean class exists:  
  
package com.cg.beans;  
class Product  
{  
      private String name;  
      private double price;  
     // getters and setter methods for these properties exist.  
}

**1)**. public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response)  
  throws ServletException, IOException {  
 Product p=new Product();  
 p.setName("Pen");  
 p.setPrice(20);  
  return new ModelAndView("DisplayProduct.jsp","product",p);  
}

**2)**. public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response)  
  throws ServletException, IOException {  
 Product p=new Product();  
 p.setName("Pen");  
 p.setPrice(20);  
  return new ModelAndView("product",p,"DisplayProduct.jsp");  
}

**3)**. **public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response)  
  throws ServletException, IOException {  
Product p=new Product();  
 p.setName("Pen");  
 p.setPrice(20);  
  return new ModelAndView("product",p,"DisplayProduct");  
}**

**4)**. **public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse response)  
  throws ServletException, IOException {  
 Product p=new Product();  
 p.setName("Pen");  
 p.setPrice(20);  
  return new ModelAndView("DisplayProduct","product",p);  
}**

**Q.**@NamedQueries annotation on entity Student for which of the given syntax is correct to use  
fetching all student records from database?

**1)**. **@Entity  
@Table(name = "students")  
@NamedQueries(  
@NamedQuery(name = "getAllStudents", query = "SELECT student FROM Student student"))  
public class Student implements Serializable {  .............. }**

**2)**. @Entity  
@Table(name = "students")  
  
@NamedQuery(name = "getAllStudents", query = "SELECT student FROM Student student")  
public class Student implements Serializable {  .............. }

**3)**. **@Entity  
@Table(name = "students")  
@NamedQueries(  
name = "getAllStudents", query = "SELECT student FROM Student student")  
public class Student implements Serializable {  .............. }**

**4)**. @Entity  
@Table(name = "students")  
@NamedQueries(  
@NamedQuery(type = "getAllStudents", query = "SELECT student FROM Student student"))  
public class Student implements Serializable {  .............. }

**Q.** Examine the below class:  
  
public class Employee  
{  
 public Employee(int empId, int age){  
  ....  
 }  
 ...  
}  
Consider Spring Configuration code snippet for the above class:  
<bean id="employee" class="Employee">  
<constructor-arg \_\_\_\_\_\_\_\_\_\_\_ value="2001"/>  
<constructor-arg \_\_\_\_\_\_\_\_\_\_\_ value="24"/>  
</bean>  
  
Select correct option(s) to fill the blank lines in the above code for completing the configuration   
of the bean to inject values for the properties through constructor injection.

**1)**. **type="int"  
type="double"**

**2)**. **index="0"  
index="1"**

**3)**. index="1"  
index="2"

**4)**. order="0"  
order="1"

**Q.**  EntityManager injected by which of the following annotation ?

**1)**. **@PersistenceUnit**

**2)**. **@PersistenceContext**

**3)**. @Persistence

**4)**. @Context

**Q.** Which of the given statement is true about above code?  
  
EntityManagerFactory factory = Persistence  
    .createEntityManagerFactory("JPA-PU");

**1)**. **JPA-PU is the name of persistence unit, which must be defined in persistence.xml**

**2)**. EntityManagerFactory is a class, using which we can create EntityManager

**3)**. **both 0 and 1**

**4)**. none of the above

**Q.** Select the most appropriate code to create a Web service end-point interface:

**1)**. @WebService  
public interface Book {  
 @WebMethod  
 float getPrice(String bookName);  
}

**2)**. @WebMethod  
@SOAPBinding(style = Style.RPC)  
public interface Book {  
 @WebService  
 float getPrice(String bookName);  
}

**3)**. **@WebService  
@SOAPBinding(style = Style.RPC)  
public interface Book {  
 @WebMethod  
 float getPrice(String bookName);  
}**

**4)**. public interface Book {  
 @WebMethod  
 float getPrice(String bookName);  
}

**5)**. @WebService  
public interface Book {  
 float getPrice(String bookName);  
}

**Q.** Given the code snippet below:   
  
public class LogAdvice {  
    public void aroundMethod() {  
          System.out.println("Around Method Call");  
    }  
}      
  
Consider the partial code in Spring configuration file:  
  
1.<beans>  
2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
3. <bean id="employeeService" class="com.cg.EmployeeService" />  
4.   <aop:config>  
5.      <aop:aspect \_\_\_\_\_\_\_\_\_\_\_\_>  
6.      <aop:around pointcut="execution(\* com.cg.EmployeeService.addEmployee())"   method="aroundMethod" />  
7.      </aop:aspect>  
8.     </aop:config>  
9.</beans>  
  
What changes need to be done on line 2 and line 5 in the above configuration file to make   
the above code as aspect?

**1)**. **In line 1, register LogAdvice bean as a bean like any other class configuration.**

**2)**. In line 1, register LogAdvice bean with the use of <aop:advice> config element.

**3)**. **In Line 5, refer the registered advice bean as shown below:  
<aop: aspect ref="logAdvice">**

**4)**. In Line 5, refer the registered advice bean as shown below:  
<aop: aspect refer="logAdvice">

**Q.**Which of the following method/s available in jQuery object to works with CSS class attribute ?

**1)**. addClass()

**2)**. hasClass()

**3)**. **removeClass()**

**4)**. **All of the above**

**Q.**Consider there is a class named as "Student.java" which is annotated with @Component and   
the class exists in "com.cg.beans" package. Which code inclusion will detect this class?

**1)**. **Include <context:component-scan base-package="com.cg.\*" /> in Spring configuration file**

**2)**. **Include <context:component-scan base-package="com.cg.beans" /> in Spring configuration file**

**3)**. Annotated classes will be detected automatically.

**4)**. Include <context:component-scan base-package="com.cg.Student" /> in web.xml

**Q.** Examine the below code:  
  
package com.cg;  
public class ProductService  
{  
 private ProductDAO productDAO;  
 .........  
}  
  
Select the option(s) which suits to configure the above class in spring configuration file as beans.

**1)**. <bean id="productService" class="com.cg.ProductService">  
          <bean class="com.cg.ProductDAO"/>  
</bean>

**2)**. **<bean id="productDAO" class="com.cg.ProductDAO"/>  
<bean id="productService" class="com.cg.ProductService">  
 <property name="productDAO">  
            <ref bean="productDAO"/>  
 </property>  
</bean>**

**3)**. **<bean id="productDAO" class="com.cg.ProductDAO"/>  
<bean id="productService" class="com.cg.ProductService">  
 <property name="productDAO">  
            <refer bean="productDAO"/>  
 </property>  
</bean>**

**4)**. **<bean id="productDAO" class="com.cg.ProductDAO"/>  
<bean id="productService" class="com.cg.ProductService" autowire="byName"/>**

**Q.**Given below following URL "http://localhost:8080/SpringRESTFULMVC/movie/ironMan"   
which of the method will call ?

**1)**. **@Controller  
@RequestMapping("/movie")  
public class MovieController {  
   
 @RequestMapping(value = "/{name}", method = RequestMethod.GET)  
 public String getMovie(@PathVariable String name, ModelMap model) {  
   
  model.addAttribute("movie", name);  
  return "list";  
   
 }**

**2)**. **@Controller  
@RequestMapping("/movie")  
public class MovieController {  
   
 @RequestMapping(value = "/{uname}{name}", method = RequestMethod.GET)  
 public String getMovie(@PathVariable String name, ModelMap model) {  
   
  model.addAttribute("movie", name);  
  return "list";  
   
 }**

**3)**. @Controller  
@RequestMapping("/movie")  
public class MovieController {  
   
@RequestMapping(value = "/", method = RequestMethod.GET)  
 public String getDefaultMovie(ModelMap model) {  
   
  model.addAttribute("movie", "this is default movie");  
  return "list";  
   
 }}

**4)**. @Controller  
@RequestMapping("/id")  
public class MovieController {  
   
 @RequestMapping(value = "/{name}", method = RequestMethod.GET)  
 public String getMovie(@PathVariable String name, ModelMap model) {  
   
  model.addAttribute("movie", name);  
  return "list";  
   
 }

**Q.** Why Restful services is easy to work with?

**1)**. Restful web services are easier to work with , as resources can be accessed via SOAP.

**2)**. Restful web services are easier to work with , as resources can be accessed via Http.

**3)**. **Restful web services are easier to work with , as resources can be accessed by user- defined   
messaging formats .**

**4)**. **Restful web services are easier to work with , as resources can be identified by URIs.**

**Q.** Consider the below code snippet exists in Spring Configuration file:  
  
<beans ...>  
<context:component-scan base-package="com.cg.bean" />  
<util:properties id="custprops" location="classpath:customer.properties" />  
</beans>  
  
Contents in "customer.properties" file looks as shown below:  
  
custid=1001  
name=Gita  
  
Identify the bean code snippet in which value should be injected through annotations   
and using spring SPEL.

**1)**. @Component("customer")  
public class Customer {  
   
 @Value("${custprops.custid}")  
 private int custId;  
   
 @Value("${custprops.name}")  
 private String name;  
 ......  
}

**2)**. @Component("customer")  
public class Customer {  
   
 @Value("%{custprops.custid}")  
 private int custId;  
   
 @Value("%{custprops.name}")  
 private String name;  
 ......  
}

**3)**. **@Component("customer")  
public class Customer {  
   
 #{custprops.custid}  
 private int custId;  
   
 #{custprops.name}  
 private String name;  
 ......  
}**

**4)**. **@Component("customer")  
public class Customer {  
   
 @Value("#{custprops.custid}")  
 private int custId;  
   
 @Value("#{custprops.name}")  
 private String name;  
 ......  
}**

**Q.**Which of the following statement(s) is/are true about @RequestMapping annotation?

**1)**. @RequestMapping annotation can be used only at Method Level.

**2)**. @RequestMapping annotation can be used at Class Level only.

**3)**. **@RequestMapping annotation can be used at both class and method Level.**

**4)**. @RequestMapping can also be used to specify the acceptable request method.

**Q.** Consider the following code:  
  
@Pointcut("execution(\* com.cg.service.EmployeeService.\*(..))")  
  
Which of the statements are correct about the above code?

**1)**. **Pointcut expression is used to apply advice whenever any method of EmployeeService class   
in com.cg.service package is executed.**

**2)**. **Pointcut expression is used to apply advice on a method with void return type on EmployeeService class   
in com.cg.service package is executed.**

**3)**. Pointcut expression is used to apply advice on method with 2 arguments in EmployeeService class   
in com.cg.service package is executed.

**4)**. All the statements are true

**Q.** Which of the following API avoid runtime cast exception in JPA?

**1)**. Query

**2)**. **TypedQuery<T>**

**3)**. **EntityManager**

**4)**. None of the above

**Q.** Which of the following statement(s) is/are true regarding Spring AOP?  
  
i. Weaving is the process of inserting aspects into application.  
ii. Proxy is the class being advised.  
iii. Pointcut is a point in the execution of the application where an aspect can be plugged in.  
iv. Aspect is the cross cutting functionality being implemented  
v. Spring's AOP supports both, method interceptor as well as constructor interceptor

**1)**. Statements ii, iii and v are true

**2)**. Statements ii and iv are true

**3)**. **Statements i and iv are true**

**4)**. Statements i and iii are true

**Q.** Select the correct statement about spring annotations?

**1)**. **@Service: This annotation marks classes that implement a part of the business logic of the application.**

**2)**. **@ComponentScan : An equivalent for Spring XML's <context:component-scan/> is provided with   
the @ComponentScan annotation.**

**3)**. **@EnableAutoConfiguration annotation  will trigger automatic loading of all the beans the application requires.**

**4)**. **@Configuration  annotation tells Spring that a method annotated with @Bean will return an object   
that should be registered as a bean in the Spring application context.**

**Q.**Assume a collection of employee objects exists  in a Spring application and employee entity   
contains fields such as id, name, designation and salary. Mittal wants to retrieve name and   
designation details of employees whose salary is greater than 50000 .  
  
Identify the code to implement the above said requirements.

**1)**. **<property name="result" value="#{emp.?[salary gt 50000].![name + ','+designation]}"/>**

**2)**. **<property name="result" value="${emp.?[salary gt 50000].![name + ','+designation]}"/>**

**3)**. <property name="result" value="%{emp.![salary gt 50000].$[name + ','+designation]}"/>

**4)**. <property name="result" value="#{emp.![salary gt 50000].?[name + ','+designation]}"/>

**Q.**In spring MVC, \_\_\_\_\_\_\_ queries a view resolver with logical name to find actual view page.

**1)**. HandlerMapping

**2)**. **DispatcherServlet**

**3)**. ModelAndView

**4)**. Controller

**Q.** Examine the below code in Spring configuration file:  
  
<beans>  
<bean id="testBean" class="com.cg.beans.Test" >  
 <property name="testId" value="101"></property>  
 <property name="testName" value="Java"></property>  
 <property name="invigilators">  
   <list>  
  <ref bean="staff1"/>  
  <ref bean="staff2"/>  
   </list>  
 </property>  
</bean>  
<bean id="staff1" class="com.cg.beans.Staff">  
 <property name="staffId" value="4001"></property>  
 <property name="name" value="Rama"></property>  
</bean>  
  
<bean id="staff2" class="com.cg.beans.Staff">  
 <property name="staffId" value="4002"></property>  
 <property name="name" value="Sita"></property>  
</bean>  
</beans>  
  
Choose the appropriate spring beans code which will match with the given configuration code.

**1)**. **package com.cg.beans;  
public class Staff {  
 private int staffId;  
 private String name;  
 //getter and setter methods for these properties  
}**

**2)**. package com.cg.beans;  
public class Staff {  
 private int staffId;  
 private String name;  
 private List<Test> tests;  
 //getter and setter methods for these properties  
}

**3)**. **public class Test {  
 private int testId;  
 private String testName;  
 private List<Staff> invigilators;  
 //getter and setter methods for these properties  
}**

**4)**. **public class Test {  
 private int testId;  
 private String testName;  
 private Staff invigilators;  
 //getter and setter methods for these properties  
}**

**Q.** public class Login  
{  
    1. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
    private String username;  
    2. \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
    private String password;  
    //getter and setter method exists for all these properties  
}  
  
Which statement(s) insertion in the above code at Line 1 and 2 will validate the field values to   
accept username of 3 characters only and password value of atleast 6 characters long.

**1)**. **@Size(min=3,max=3,message="Username must be 3 characters");**

**2)**. @Size(min=3,message="Username must be 3 characters");

**3)**. **@Size(min=6,message="Password must be atleast 6 characters long");**

**4)**. @Size(max=6,message="Password must be atleast 6 characters long");

**Q.** Select the list of clients a RESTful web service can support:

**1)**. Java Web client

**2)**. **.NET client**

**3)**. **standalone Java application**

**4)**. None of the above

**Q.** Match the following:  
  
1. @Component       
2. @Controller  
3. @Service   
4. @Repository  
  
a. Business logic implementation class will be marked with this annotation  
b. Represent a data access logic class  
c. Classes annotated with this is referred as spring beans  
d. Controller in Spring MVC will be annotated with this.

**1)**. 1 - b, 2 - d, 3 - a, 4 - c

**2)**. **1 - c, 2 - a, 3 - d, 4 - b**

**3)**. 1 - c, 2 - d, 3 - b, 4 - a

**4)**. **1 - c, 2 - d, 3 - a, 4 - b**

**Q.** Prem wants to invoke static method in a spring configuration file.   
Identify which code suits for this requirement.   
  
Consider a static method named as "getCustomerId" exists in a class "com.cg.Customer".

**1)**. **<property name="result" value="#{T(com.cg.Customer).getCustomerId()}/>**

**2)**. <property name="result" value="${T(com.cg.Customer).getCustomerId()}/>

**3)**. <property name="result" value="${Type(com.cg.Customer).getCustomerId()}/>

**4)**. <property name="result" value="#{com.cg.Customer.getCustomerId()}/>

**Q.** \_\_\_\_\_\_\_ class validations specified in Spring is checked whether input is adhering.

**1)**. Valid

**2)**. **BindingResult**

**3)**. **Validatable**

**4)**. ValidationAware

**Q.** Refer the below code and predict the output:  
  
Var app=angular.module('myapp',[]);  
app.directive('myWorld',function(){  
 return{  
                  restrict: 'E',  
                 replace : true,  
                 template: '<h1>Hello</h1>'  
  
};  
  
});

**1)**. **<my-world></my-world>**

**2)**. **<p class='my-world'></p>**

**3)**. <p my-world></p>

**4)**. <!-- my-world  -->

**Q.**Which of the following option represent the correct sequence steps for Spring JPA integration?

**1)**. 1. Create or lookup for DataSource  
2.  Obtain EntityManager in DAO repositories.  
3.  Configure Persistence Unit using Spring API.  
4.  Setup transaction management on Services

**2)**. **1. Create or lookup for DataSource  
2.  Configure Persistence Unit using Spring API.  
3.  Obtain EntityManager in DAO repositories.  
4.  Setup transaction management on Services**

**3)**. **1. Create or lookup for DataSource  
2.  Setup transaction management on Services   
3.  Obtain EntityManager in DAO repositories.  
4.  Configure Persistence Unit using Spring API.**

**4)**. 1. Setup transaction management on Services.  
2.  Configure Persistence Unit using Spring API.  
3.  Obtain EntityManager in DAO repositories.  
4.  Create or lookup for DataSource.

**Q.** Choose the correct statement about  Spring Boot?

**1)**. **Dynamically wires up beans and settings and applies them to your application context**

**2)**. Provides a productive and stream-lined full-stack web framework by combining the power.

**3)**. An open-source library with which an application can be connected with cloud environment.

**4)**. **A lightweight, comprehensive batch framework designed to enable the development of robust   
batch applications vital for the daily operations of enterprise systems.**