**答案**

1. D
2. B
3. B
4. D
5. A
6. D
7. A
8. ABD
9. D
   * + 1. 构造器Constructor不能被继承，因此不能重写，但可以被重载
       2. HashMap是以键值对(key-value)的形式存储元素的。HashMap需要一个hash函数，它使用hashCode()和equals()方法来向集合/从集合添加和检索元素。当调用put()方法的时候，HashMap会计算key的hash值，然后把键值对存储在集合中合适的索引上。如果key已经存在了，value会被更新成新值。HashMap的一些重要的特性是它的容量(capacity)，负载因子(load factor)和扩容极限(threshold resizing)
       3. 不能，一个对象的一个synchronized方法只能由一个线程访问
       4. GC是垃圾收集器。Java 程序员不用担心内存管理，因为垃圾收集器会自动进行管理。要请求垃圾收集，可以调用下面的方法之一：

System.gc()

Runtime.getRuntime().gc()

* + - 1. 垃圾回收不会发生在永久代，如果永久代满了或者是超过了临界值，会触发完全垃圾回收(Full GC)。如果你仔细查看垃圾收集器的输出信息，就会发现永久代也是被回收的。这就是为什么正确的永久代大小对避免Full GC是非常重要的原因
      2. init() service() doGet() doPost() getServletConfig() getServletInfo() destroy()
      3. 1、客户端浏览器发出HTTP请求。

2、根据web.xml配置，该请求被FilterDispatcher接收。

3、根据struts.xml配置，找到需要调用的Action类和方法， 并通过IoC方式，将值注入给Aciton。

4、Action调用业务逻辑组件处理业务逻辑，这一步包含表单验证。

5、Action执行完毕，根据struts.xml中的配置找到对应的返回结果result，并跳转到相应页面。

6、返回HTTP响应到客户端浏览器。

11. c3p0、dbcp

12. 在Action里面定义属性或成员变量,名称必须要和页面表单元素name属

性的值一样,区分大小写,然后生成该属性或成员变量的set方法

2. 在Action里面获取HttpServletRequest对象,然后根据

HttpServletRequest. getParameter (“”)方法获取参数值

a. HttpServletRequest request = ServletActionContext.getRequest()

b. ActionContext ct= ActionContext.getContext()

HttpServletRequest request=

(HttpServletRequest)ct.get(ServletActionContext. HTTP\_REQUEST );

3．在Action里通过下列代码获取

ActionContext context =ActionContext.getContext();

Map params = context.getParameters();

String username = (String) params.get("parameter");

1. 1.客户端请求提交到DispatcherServlet  
   2. 由DispatcherServlet控制器查询一个或多个HandlerMapping，找到处理请求的Controller  
   3. DispatcherServlet将请求提交到Controller  
   4. Controller调用业务逻辑处理后，返回ModelAndView  
   5. DispatcherServlet查询一个或多个ViewResoler视图解析器，找到ModelAndView指定的视图  
   6. 视图负责将结果显示到客户端



1. Spring中的 org.springframework.beans 包和 org.springframework.context包构成了Spring框架IoC容器的基础。

BeanFactory 接口提供了一个先进的配置机制，使得任何类型的对象的配置成为可能。ApplicationContex接口对BeanFactory（是一个子接口）进行了扩展，在BeanFactory的基础上添加了其他功能，比如与[Spring的AOP](http://howtodoinjava.com/category/frameworks/java-spring-tutorials/spring-aop/" \t "_blank)更容易集成，也提供了处理[message resource的机制](http://howtodoinjava.com/2015/02/10/spring-mvc-internationalization-i18n-and-localization-i10n-example/)（用于国际化）、事件传播以及应用层的特别配置，比如针对Web应用的WebApplicationContext。

**org.springframework.beans.factory.BeanFactory** 是Spring IoC容器的具体实现，用来包装和管理前面提到的各种bean。BeanFactory接口是Spring IoC 容器的核心接口。

BeanFactory 可以理解为含有bean集合的工厂类。BeanFactory 包含了种bean的定义，以便在接收到客户端请求时将对应的bean实例化。

BeanFactory还能在实例化对象的时生成协作类之间的关系。此举将bean自身与bean客户端的配置中解放出来。BeanFactory还包含了bean生命周期的控制，调用客户端的初始化方法（initialization methods）和销毁方法（destruction methods）。

从表面上看，application context如同bean factory一样具有bean定义、bean关联关系的设置，根据请求分发bean的功能。但application context在此基础上还提供了其他的功能。

1. 提供了支持国际化的文本消息
2. 统一的资源文件读取方式
3. 已在监听器中注册的bean的事件

以下是三种较常见的 ApplicationContext 实现方式：

1、ClassPathXmlApplicationContext：从classpath的XML配置文件中读取上下文，并生成上下文定义。应用程序上下文从程序环境变量中取得。

|  |  |
| --- | --- |
| 1 | ApplicationContext context = new ClassPathXmlApplicationContext(“bean.xml”); |

2、FileSystemXmlApplicationContext ：由文件系统中的XML配置文件读取上下文。

|  |  |
| --- | --- |
| 1 | ApplicationContext context = new FileSystemXmlApplicationContext(“bean.xml”); |

3、XmlWebApplicationContext：由Web应用的XML文件读取上下文。

15. Spring Bean的生命周期简单易懂。在一个bean实例被初始化时，需要执行一系列的初始化操作以达到可用的状态。同样的，当一个bean不在被调用时需要进行相关的析构操作，并从bean容器中移除。

Spring bean factory 负责管理在spring容器中被创建的bean的生命周期。Bean的生命周期由两组回调（call back）方法组成。

1. 初始化之后调用的回调方法。
2. 销毁之前调用的回调方法。

Spring框架提供了以下四种方式来管理bean的生命周期事件：

* InitializingBean和DisposableBean回调接口
* 针对特殊行为的其他Aware接口
* Bean配置文件中的Custom init()方法和destroy()方法
* @PostConstruct和@PreDestroy注解方式

1. Spring框架中使用到了大量的设计模式，下面列举了比较有代表性的：

* 代理模式—在AOP和remoting中被用的比较多。
* 单例模式—在spring配置文件中定义的bean默认为单例模式。
* 模板方法—用来解决代码重复的问题。比如. [RestTemplate](http://howtodoinjava.com/2015/02/20/spring-restful-client-resttemplate-example/" \o "Spring RESTFul Client – RestTemplate Example" \t "_blank), JmsTemplate, JpaTemplate。
* 前端控制器—Spring提供了DispatcherServlet来对请求进行分发。
* 视图帮助(View Helper )—Spring提供了一系列的JSP标签，高效宏来辅助将分散的代码整合在视图里。
* 依赖注入—贯穿于BeanFactory / ApplicationContext接口的核心理念。
* 工厂模式—BeanFactory用来创建对象的实例。