

1. forward 和 redirect 区别

1.从地址栏显示来说

forward 是服务器请求资源,服务器直接访问目标地址的 URL,把那个 URL 的响应内容读取过来,然后把这些内容再发给浏览器.浏览器根本不知道服务器发送的内容从哪里来的,所以它的地址栏还是原来的地址.

redirect 是服务端根据逻辑,发送一个状态码,告诉浏览器重新去请求那个地址.所以地址栏显示的是新的 URL.

2.从数据共享来说

forward:转发页面和转发到的页面可以共享 **request** 里面的数据.

redirect:不能共享数据.

3.从运用地方来说

forward:一般用于用户登陆的时候,根据角色转发到相应的模块.

redirect:一般用于用户注销登陆时返回主页面和跳转到其它的网站等.

4.从效率来说

forward:高.

redirect:低.

2. servlet 的几个方法 doGet doPost 区别

3. ArrayList 里装了 Person 类对象,怎样根据 Person 类的属性年龄来排序

4. HashMap 的键值是否可以任意对象

面试总结

首先自我介绍

1. 面试官拿着你的简历,会挑其中的一个项目,然后让你说出这个框架的流程
2. 画出框架的流程图,会问每一个部分都是干什么用的
3. 针对 2 的问题,我们做的都是 web 项目,在那个框架中, **servlet** 对应的是哪一个部分
4. 由前两个问题牵引出 web.xml 文件配置中都有哪些属性,以及他的作用
5. 对 **spring** 了解吗
6. **spring** 的依赖注入方式有哪几种(注意不是 **spring** 的注入方式是依赖注入)
7. 有关事物的问题,做项目中做到哪些与事物有关的,事物是怎么控制的,怎么去写
8. 触发器,存储过程也说了一点
9. 项目开发中,如果遇到一个问题,你自己也不知道该用什么技术去解决,怎么去解决,该如何去查
10. 你有三年项目开发经验,觉得自己比别人有优势的地方在哪
11. 最后会问对他们公司有什么要了解的,给你个机会,让你去问问题

- 1, 自我介绍,自己做过的项目,擅长的技术。
- 2, 用过的框架,最擅长的那个?
- 3, 所知道的 MVC 框架还有哪些?
- 4, 经常去些什么样的网站,对自己将来有什么样的打算,规划。
- 5, 喜欢技术吗,(喜欢)。举个例子来说明你喜欢技术。
- 6, 以前项目中有没有加班,频率、加班时间长度。对加班的看法。
- 7, 以前的项目主中要做什么内容, SE 级别,还是开发。
- 8, 在项目组怎样做项目,没有详细设计能做吗?
- 9, Struts 用的什么版本。
- 10, Struts2 用过吗? 和 Struts1 有什么区别。
- 11, Spring 的 AOP 了解吗,主要用在项目的那些方面。

12, 以前的项目规模都是多大的。

1. 首先自我介绍
2. 问最熟悉的项目
3. 画出 STRUTS 框架响应 jsp 的流程图.
4. 针对 2 的问题, 我们做的都是 web 项目, 在那个框架中, servlet 对应的是哪一个部分
5. 由前两个问题牵引出 web.xml 文件配置中都有哪些属性, 以及他的作用
6. 对 spring 了解吗
7. spring 的依赖注入方式有哪一种 (注意不是 spring 的注入方式是依赖注入)
8. 有关事物的问题, 做项目中做到哪些与事物有关的, 事物是怎么控制的, 怎么去写
9. Struts 底层的相关知识
10. 项目开发中, 如果遇到一个问题, 你自己也不知道该用什么技术去解决, 怎么去解决, 该如何去查
11. 你有三年项目开发经验, 觉得自己比别人有优势的地方在哪
12. 最后会问对他们公司有什么要了解的, 给你个机会, 让你去问问题

1. 首先自我介绍
2. 在你所用过的框架中你比较喜欢那个
3. 问你做过的这些项目中那个收获最大, 收获到了什么
4. ibatis 和 Hibernate 的区别
5. servlet 的生命周期
6. spring 的两个主要特性(AOP 和 IOC)
7. 说一下你所理解的 J2EE 是什么
8. 为什么说 JBOSS 符合 J2EE 的要求, 而 TOMCAT 不符合 J2EE 的要求
9. Hibernate 的优点和缺点
10. 你认为在项目中最重要的是什么
11. 要是分给你的任务, 你感到完成有困难, 你会怎么办
12. 最后你对支付宝有什么要问的

这次去杭州支付宝面试, 因为我的面试官是个开发主管, 框架方面的技术问的很少, 大部分都是根据问你项目而延伸的一些 Sql, UML 等问题, 简历的项目当中如果有快钱的项目要好好准备下, 对快钱的项目非常感兴趣。主要问题有以下:

1. 首先面试官自己自我介绍, 然后让你自我介绍。
 2. 哪家公司的? 哪年出生的? 哪年毕业的? 工作几年了? 这些问题都是在看你的回答是不是跟简历不一样, 简历有没有作假。回答一定不要考虑。
 3. 对于协力员工的看法?
 4. 你的人生规划职业规划是怎么样的?
 5. 对于项目加班有什么看法? 你加班的极限是多少?
 6. 熟悉一个新框架需要多长时间? (支付宝自己有个自己的框架)
 7. 说出你认为对你影响最深的项目, 并说出原因。
 8. Oracle 中的分页 Sql 怎么写?
 9. 简单地向一个不懂计算机的人说明一下 java 的多态。
 10. 说一下你知道的 java 设计模式。
 11. struts, spring 中应用了哪写 java 设计模式?
 12. 说下 spring 的代理模式, 画下 spring 代理模式的类图。
 13. 快钱的项目中所担当的模块, 根据你的回答就此展开一些问题。
 14. 宝钢物流的项目的入库那个模块在开发中大致都用了哪些类? 哪些接口? 并画下 UML 图。
- 以上是主要的问题, 还有些问题都是根据你的回答延伸的。

- 1 简单介绍自己
- 2 根据你的介绍提问
- 3 mvc 开发模式有哪些模式
- 4 你的人生规划
- 5 业余爱好
- 6 最近所做的项目中除了你做的模块,还有哪些
- 7 你都去过哪做项目

总结:根据面试官不同,他可能喜欢的人也不同,这个面试官喜欢做事有计划的

你对 Java 的集合框架了解吗？ 能否说说常用的类？

说说 Hashtable 与 HashMap 的区别： 源代码级别的区别呢？

平时用过的 List 有哪些？ （除了 ArrayList 和 LinkedList），ArrayList 和 LinkedList 的区别？

ArrayList 的特点，内部容器是如何扩充的？

Properties 类的特点？ 线程安全？

=====

平时使用过的框架有哪些？（我提到了 Struts2）

请说一下 Struts2 的初始化？ 和类的创建？（从源代码角度出发）

据你了解，除了反射还有什么方式可以动态的创建对象？（我提到了 CGLIB…… 我以为他会接着问 CGLIB, 揪心中……，结果他没问）

请说一下 Struts2 是如何把 Action 交给 Spring 托管的？它是单例的还是多例？ 你们页面的表单对象是多例还是单例？

请说一下你们业务层对象是单例还是多例的？

请说一下 Struts2 源代码中有哪些设计模式？

=====

请说一下，你觉得你最熟悉的技术特点？（我提到了并发编程）

请说一下线程安全出现的原因？

请说一下线程池的中断策略(4个)？各有什么特点？

请说一下 Tomcat 配置不同应用的不同端口如何配置？如何配置数据源？如何实现动态部署？

请说一下 Java 常用的优化？

你了解最新的 Servlet 规范吗？简单说一下？（我提到了推）

那请你说一下“推”是如何实现的？

线程安全下，StringBuffer 与 StringBuilder 的区别？它们是如何扩充内部数组容量的？（源代码）

请说一下 Tomcat 中的设计模式？（我提到观察者模式）

是否可以说说 Java 反射的相关优化机制？（我说我不太清楚……他说没关系 --！）

请说一些 Mysql 的常用优化策略？

因为我之前有提到过“推”，他可能对我的知识面比较感兴趣，要我说说平时都看些什么书，还了解一些什么其他的技术范畴。

（他首先提到 SOA，我说有了解，并且是未来的趋势，还有提到云计算，我说有过一定了解，但是并未深究）

=====

之后是几个职业方面的问题？

你觉得你的潜力？ 你在团队中的位置？ 你觉得跟团队中最好的还有哪些差距？ 你要花多少时间赶上他们？

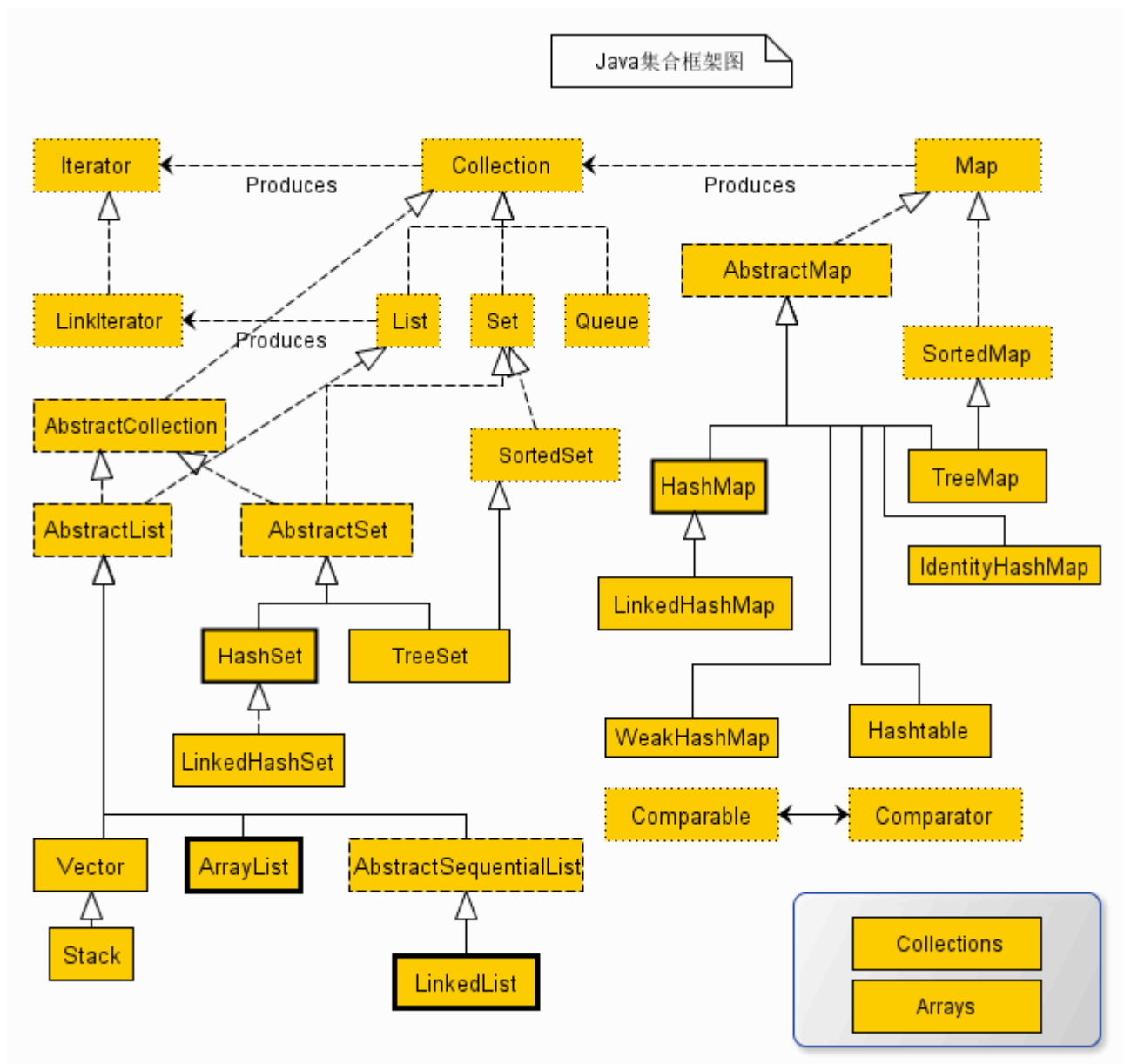
你对阿里巴巴还有什么疑问吗？ （我很囧的问了，“阿里巴巴的牛人平时都跟你们有互动吗？——本意是指培训，但是话没说清楚……”， 囧了……）

PS，下面是时候对问题的整理，里面纯粹仅限于个人浅见，如果有错误，还希望各位能指点一二。

=====

- 你对 Java 的集合框架了解吗？ 能否说说常用的类？

Java 集合框架类图：



我常用的类:

HashMap, Hashtable, HashSet, ArrayList, Vector, LinkedList, Collections, Arrays;


• 说说 Hashtable 与 HashMap 的区别(源代码级别)

1. 最明显的区别在于 Hashtable 是同步的(每个方法都是 synchronized)，而 HashMap 则不是。


2. HashMap 继承至 AbstractMap, Hashtable 继承至 Dictionary, 前者为 Map 的骨干，其内部已经实现了 Map 所需要做的大部分工作，它的子类只需要实现它的少量方法即可具有 Map 的多项特性。而后者内部都为抽象方法，需要它的实现类一一作自己的实现，且该类已过时

3. 两者检测是否含有 key 时, hash 算法不一致, HashMap 内部需要将 key 的 hash 码重新计算一边再检测, 而 Hashtable 则直接利用 key 本身的 hash 码来做验证。

HashMap:


Java 代码 

```
1. int hash = (key == null) ? 0 : hash(key.hashCode());
2. -----
3. static int hash(int h) {
4.     h ^= (h >>> 20) ^ (h >>> 12);
5.     return h ^ (h >>> 7) ^ (h >>> 4);
6. }
```


Java 代码 

```
1. int hash = (key == null) ? 0 : hash(key.hashCode());
2. -----
3. static int hash(int h) {
4.     h ^= (h >>> 20) ^ (h >>> 12);
5.     return h ^ (h >>> 7) ^ (h >>> 4);
6. }
```

Hashtable:

Java 代码 


```
1. int hash = key.hashCode();
```

Java 代码 

```
1. int hash = key.hashCode();
```

4. 两者初始化容量大小不一致, HashMap 内部为 16×0.75 , Hashtable 为 11×0.75

HashMap:

Java 代码 


```
1. static final int DEFAULT_INITIAL_CAPACITY = 16;
```



```

2. static final float DEFAULT_LOAD_FACTOR = 0.75f;
3. public HashMap() {
4.     this.loadFactor = DEFAULT_LOAD_FACTOR;
5.     threshold=(int) (DEFAULT_INITIAL_CAPACITY*DEFAULT_LOAD_FACTOR);
6.     table = new Entry[DEFAULT_INITIAL_CAPACITY];
7.     init();
8. }
9. ....

```


Java 代码 

```

1. static final int DEFAULT_INITIAL_CAPACITY = 16;
2. static final float DEFAULT_LOAD_FACTOR = 0.75f;
3. public HashMap() {
4.     this.loadFactor = DEFAULT_LOAD_FACTOR;
5.     threshold=(int) (DEFAULT_INITIAL_CAPACITY*DEFAULT_LOAD_FACTOR);
6.     table = new Entry[DEFAULT_INITIAL_CAPACITY];
7.     init();
8. }
9. ....

```

Hashtable:

Java 代码 

```

1. public Hashtable() {
2.     this(11, 0.75f);
3. }
4. ----
5. public Hashtable(int initialCapacity, float loadFactor) {
6.     .....
7.     this.loadFactor = loadFactor;
8.     table = new Entry[initialCapacity];
9.     threshold = (int)(initialCapacity * loadFactor);
10. }

```

Java 代码 

```

1. public Hashtable() {
2.     this(11, 0.75f);
3. }
4. ----
5. public Hashtable(int initialCapacity, float loadFactor) {
6.     .....
7.     this.loadFactor = loadFactor;
8.     table = new Entry[initialCapacity];

```

```
9.         threshold = (int)(initialCapacity * loadFactor);
10.    }
```


其实后续的区别应该还有很多， 这里先列出 4 点。

• 平时除了 ArrayList 和 LinkedList 外,还用过的 List 有哪些? ArrayList 和 LinkedList 的区别?


事实上，我用过的 List 主要就是这 2 个， 另外用过 Vector.

ArrayList 和 LinkedList 的区别：

1. 毫无疑问，第一点就是两者的内部数据结构不同， ArrayList 内部元素容器是一个 Object 的数组，
而 LinkedList 内部实际上一个链表的数据结构，其有一个内部类来表示链表.

Java 代码 

```
1. (ArrayList)
2. private transient Object[] elementData;
3.
4. .....
5.
6. (LinkedList)
7. private transient Entry<E> header = new Entry<E>(null, null, null)
   ;/链表头
8.
9. //内部链表类.
10. private static class Entry<E> {
11.     E element; //数据元素
12.     Entry<E> next; // 前驱
13.     Entry<E> previous;//后驱
14.     Entry(E element, Entry<E> next, Entry<E> previous) {
15.         this.element = element;
16.         this.next = next;
17.         this.previous = previous;
18.     }
19. }
```


Java 代码 

```

20. (ArrayList)
21. private transient Object[] elementData;
22.
23. ....
24.
25. (LinkedList)
26. private transient Entry<E> header = new Entry<E>(null, null, null);
    /链表头
27.
28. //内部链表类.
29. private static class Entry<E> {
30.     E element; //数据元素
31.     Entry<E> next; // 前驱
32.     Entry<E> previous; //后驱
33.     Entry(E element, Entry<E> next, Entry<E> previous) {
34.         this.element = element;
35.         this.next = next;
36.         this.previous = previous;
37.     }
38. }

```


2. 两者的父类不同，也就决定了两者的存储形式不同。ArrayList 继承于 AbstractList, 而 LinkedList 继承于 AbstractSequentialList. 两者都实现了 List 的骨干结构，只是前者的访问形式趋向于 “随机访问” 数据存储（如数组），后者趋向于 “连续访问” 数据存储（如链接列表）

Java 代码 

```

1. public class ArrayList<E> extends AbstractList<E>
2. -----
3. public class LinkedList<E> extends AbstractSequentialList<E>

```

Java 代码 

```


4. public class ArrayList<E> extends AbstractList<E>
5. -----
6. public class LinkedList<E> extends AbstractSequentialList<E>

```

3. 再有就是两者的效率问题，ArrayList 基于数组实现，所以毫无疑问可以直接用下标来索引，其索引数据快，插入元素设计到数组元素移动，或者数组扩充，所以插入元素要慢。LinkedList 基于链表结构，插入元素只需要改变插入元素的前后项的指向即可，故插入数据要快，而索引元素需要向前向后遍历，所以索引元素要慢。

- ArrayList 的特点，内部容器是如何扩充的？

上一点谈到了 ArrayList 的特点，这里略，重点来看其内部容器的扩充：

Java 代码 

```
1. public void ensureCapacity(int minCapacity) {
2.     modCount++;
3.     int oldCapacity = elementData.length;
4.     if (minCapacity > oldCapacity) {
5.         Object oldData[] = elementData;
6.         //这里扩充的大小为原大小的大概 60%
7.         int newCapacity = (oldCapacity * 3) / 2 + 1;

8.         if (newCapacity < minCapacity)
9.             newCapacity = minCapacity;
10.        //创建一个指定大小的新数组来覆盖原数组
11.        elementData = Arrays.copyOf(elementData, newCapacity);

12.    }
13. }
```

Java 代码 


```
1. public void ensureCapacity(int minCapacity) {
2.     modCount++;
3.     int oldCapacity = elementData.length;
4.     if (minCapacity > oldCapacity) {
5.         Object oldData[] = elementData;
6.         //这里扩充的大小为原大小的大概 60%
7.         int newCapacity = (oldCapacity * 3) / 2 + 1;

8.         if (newCapacity < minCapacity)
9.             newCapacity = minCapacity;
10.        //创建一个指定大小的新数组来覆盖原数组
11.        elementData = Arrays.copyOf(elementData, newCapacity);

12.    }
13. }
```

- Properties 类的特点？ 线程安全吗？

Properties 继承于 Hashtable, 所以它是线程安全的。其特点是：它表示的是一个持久的属性集，它可以保存在流中或者从流中加载，属性列表的每一个键和它所对应的值都是一个“字符串” 其中，常用的方法是 load() 方法，从流中加载属性：

Java 代码 

```

<SPAN style="FONT-WEIGHT: normal">public synchronized void load(InputStream inStream) throws
    IOException {

    // 将输入流转换成 LineReader

    load0(new LineReader(inStream));

}

private void load0(LineReader lr) throws IOException {

    char[] convtBuf = new char[1024];

    int limit;

    int keyLen;

    int valueStart;

    char c;

    boolean hasSep;

    boolean precedingBackslash;

    // 一行一行处理

    while ((limit = lr.readLine()) >= 0) {

        c = 0;

        keyLen = 0;

        valueStart = limit;

        hasSep = false;

        precedingBackslash = false;

        // 下面用 2 个循环来处理 key, value

        while (keyLen < limit) {

            c = lr.lineBuf[keyLen];

            // need check if escaped.

            if ((c == '=' || c == ':') && !precedingBackslash) {

```

```

        valueStart = keyLen + 1;

        hasSep = true;

        break;

    } else if ((c == ' ' || c == '\t' || c == '\f')

                && !precedingBackslash) {

        valueStart = keyLen + 1;

        break;

    }

    if (c == '\\') {

        precedingBackslash = !precedingBackslash;

    } else {

        precedingBackslash = false;

    }

    keyLen++;

}

while (valueStart < limit) {

    c = lr.lineBuf[valueStart];

    if (c != ' ' && c != '\t' && c != '\f') {

        if (!hasSep && (c == '=' || c == ':')) {

            hasSep = true;

        } else {

            break;

        }

    }

    valueStart++;

```

```

    }

    String key = loadConvert(lr.lineBuf, 0, keyLen, convtBuf);

    String value = loadConvert(lr.lineBuf, valueStart, limit
                                - valueStart, convtBuf);


    // 存入内部容器中，这里用的是 Hashtable 内部的方法.

    put(key, value);

}

} </SPAN>

```

Java 代码 

```

<SPAN style="FONT-WEIGHT: normal">public synchronized void load(InputStream inStream) throws
    IOException {

    // 将输入流转换成 LineReader

    load0(new LineReader(inStream));

}

private void load0(LineReader lr) throws IOException {

    char[] convtBuf = new char[1024];

    int limit;

    int keyLen;

    int valueStart;

    char c;

    boolean hasSep;

    boolean precedingBackslash;

    // 一行一行处理

    while ((limit = lr.readLine()) >= 0) {

```

```

c = 0;

keyLen = 0;

valueStart = limit;

hasSep = false;

precedingBackslash = false;

// 下面用 2 个循环来处理 key, value

while (keyLen < limit) {

    c = lr.lineBuf[keyLen];

    // need check if escaped.

    if ((c == '=' || c == ':') && !precedingBackslash) {

        valueStart = keyLen + 1;

        hasSep = true;

        break;

    } else if ((c == ' ' || c == '\t' || c == '\f')

        && !precedingBackslash) {

        valueStart = keyLen + 1;

        break;

    }

    if (c == '\\') {

        precedingBackslash = !precedingBackslash;

    } else {

        precedingBackslash = false;

    }

    keyLen++;

}

```



```

while (valueStart < limit) {

    c = lr.lineBuf[valueStart];

    if (c != ' ' && c != '\t' && c != '\f') {

        if (!hasSep && (c == '=' || c == ':')) {

            hasSep = true;

        } else {

            break;

        }

    }

    valueStart++;

}

```

```

String key = loadConvert(lr.lineBuf, 0, keyLen, convtBuf);

String value = loadConvert(lr.lineBuf, valueStart, limit
                            - valueStart, convtBuf);

// 存入内部容器中，这里用的是 Hashtable 内部的方法。


put(key, value);

}

```

}

LineNumber 类，是 Properties 内部的类：

Java 代码 

```

<SPAN style="FONT-WEIGHT: normal">class LineReader {

    public LineReader(InputStream inStream) {

        this.inStream = inStream;

        inByteBuf = new byte[8192];
    }
}

```

```

    }

    public LineReader(Reader reader) {

        this.reader = reader;

        inCharBuf = new char[8192];

    }

    byte[] inByteBuf;

    char[] inCharBuf;

    char[] lineBuf = new char[1024];

    int inLimit = 0;

    int inOff = 0;

    InputStream inStream;

    Reader reader;

    /**
     * 读取一行
     *
     * @return
     * @throws IOException
     */

    int readLine() throws IOException {

        int len = 0;

        char c = 0;

        boolean skipWhiteSpace = true;// 空白

        boolean isCommentLine = false;// 注释

```

```

boolean isNewLine = true;// 是否新行.

boolean appendedLineBegin = false;// 加 至行开始

boolean precedingBackslash = false;// 反斜杠

boolean skipLF = false;

while (true) {

    if (inOff >= inLimit) {

        // 从输入流中读取一定数量的字节并将其存储在缓冲区数组
inCharBuf/inByteBuf 中, 这里区分字节流和字符流

        inLimit = (inStream == null) ? reader.read(inCharBuf)

            : inStream.read(inByteBuf);

        inOff = 0;

        // 读取到的为空.

        if (inLimit <= 0) {

            if (len == 0 || isCommentLine) {

                return -1;

            }

            return len;

        }

    }

    if (inStream != null) {

        // 由于是字节流, 需要使用 ISO8859-1 来解码

        c = (char) (0xff & inByteBuf[inOff++]);

    } else {

        c = inCharBuf[inOff++];

    }
}

```

```

    if (skipLF) {

        skipLF = false;

        if (c == '\n') {

            continue;

        }

    }

    if (skipWhiteSpace) {

        if (c == ' ' || c == '\t' || c == '\f') {

            continue;

        }

        if (!appendedLineBegin && (c == '\r' || c == '\n'))

            continue;

        }

        skipWhiteSpace = false;

        appendedLineBegin = false;

    }

    if (isNewLine) {

        isNewLine = false;

        if (c == '#' || c == '!') {

            // 注释行，忽略。

            isCommentLine = true;

            continue;

        }

    }

    // 读取真正的属性内容

```

取的内容.

ngth);

```
if (c != '\n' && c != '\r') {

    // 这里类似于 ArrayList 内部的容量扩充, 使用字符数组来保存读

    lineBuf[len++] = c;

    if (len == lineBuf.length) {

        int newLength = lineBuf.length * 2;

        if (newLength < 0) {

            newLength = Integer.MAX_VALUE;

        }

        char[] buf = new char[newLength];

        System.arraycopy(lineBuf, 0, buf, 0, lineBuf.le

ngth);

        lineBuf = buf;

    }

    if (c == '\\') {

        precedingBackslash = !precedingBackslash;

    } else {

        precedingBackslash = false;

    }

} else {

    // reached EOL 文件结束

    if (isCommentLine || len == 0) {

        isCommentLine = false;

        isNewLine = true;

        skipWhiteSpace = true;

        len = 0;

        continue;

    }

}
```

CharBuf)

```
    }

    if (inOff >= inLimit) {

        inLimit = (inStream == null) ? reader.read(in

                                : inStream.read(inByteBuf);

        inOff = 0;

        if (inLimit <= 0) {

            return len;

        }

    }

    if (precedingBackslash) {

        len -= 1;

        skipWhiteSpace = true;

        appendedLineBegin = true;

        precedingBackslash = false;

        if (c == '\\r') {

            skipLF = true;

        }

    } else {

        return len;

    }

}

}

}
```



```
<SPAN style="FONT-WEIGHT: normal">class LineReader {

    public LineReader(InputStream inStream) {

        this.inStream = inStream;

        inByteBuf = new byte[8192];

    }

    public LineReader(Reader reader) {

        this.reader = reader;

        inCharBuf = new char[8192];

    }

    byte[] inByteBuf;

    char[] inCharBuf;

    char[] lineBuf = new char[1024];

    int inLimit = 0;

    int inOff = 0;

    InputStream inStream;

    Reader reader;

    /**

     * 读取一行

     *

     * @return

     * @throws IOException

     */

    int readLine() throws IOException {
```

```

int len = 0;

char c = 0;

boolean skipWhiteSpace = true; // 空白

boolean isCommentLine = false; // 注释

boolean isNewLine = true; // 是否新行.

boolean appendedLineBegin = false; // 加 至行开始

boolean precedingBackslash = false; // 反斜杠

boolean skipLF = false;

while (true) {

    if (inOff >= inLimit) {

        // 从输入流中读取一定数量的字节并将其存储在缓冲区数组
inCharBuf/inByteBuf 中, 这里区分字节流和字符流

        inLimit = (inStream == null) ? reader.read(inCharBuf)

            : inStream.read(inByteBuf);

        inOff = 0;

        // 读取到的为空.

        if (inLimit <= 0) {

            if (len == 0 || isCommentLine) {

                return -1;

            }

            return len;

        }

    }

    if (inStream != null) {

        // 由于是字节流, 需要使用 ISO8859-1 来解码

        c = (char) (0xff & inByteBuf[inOff++]);

```



```

    } else {

        c = inCharBuf[inOff++];

    }

    if (skipLF) {

        skipLF = false;

        if (c == '\n') {

            continue;

        }

    }

    if (skipWhiteSpace) {

        if (c == ' ' || c == '\t' || c == '\f') {

            continue;

        }

        if (!appendedLineBegin && (c == '\r' || c == '\n'))

            continue;

        }

        skipWhiteSpace = false;

        appendedLineBegin = false;

    }

    if (isNewLine) {

        isNewLine = false;

        if (c == '#' || c == '!') {

            // 注释行，忽略.

            isCommentLine = true;

```

取的内容.

ngth);

```
        continue;

    }

}

// 读取真正的属性内容

if (c != '\n' && c != '\r') {

    // 这里类似于 ArrayList 内部的容量扩充, 使用字符数组来保存读

    lineBuf[len++] = c;

    if (len == lineBuf.length) {

        int newLength = lineBuf.length * 2;

        if (newLength < 0) {

            newLength = Integer.MAX_VALUE;

        }

        char[] buf = new char[newLength];

        System.arraycopy(lineBuf, 0, buf, 0, lineBuf.le

ngth);

        lineBuf = buf;

    }

    if (c == '\\') {

        precedingBackslash = !precedingBackslash;

    } else {

        precedingBackslash = false;

    }

} else {

    // reached EOL 文件结束

    if (isCommentLine || len == 0) {

        isCommentLine = false;
```

CharBuf)

```
        isNewLine = true;

        skipWhiteSpace = true;

        len = 0;

        continue;
    }

    if (inOff >= inLimit) {

        inLimit = (inStream == null) ? reader.read(in

                                : inStream.read(inByteBuf);

        inOff = 0;

        if (inLimit <= 0) {

            return len;

        }
    }

    if (precedingBackslash) {

        len -= 1;

        skipWhiteSpace = true;

        appendedLineBegin = true;

        precedingBackslash = false;

        if (c == '\\r') {

            skipLF = true;

        }
    } else {

        return len;

    }
}
```

```

    }

}

} </SPAN>


```

这里特别的是，实际上，Properties 从流中加载属性集合，是通过将流中的字符或者字节分成一行行来处理的。

• 请说一下 Struts2 的初始化？和类的创建？（从源代码角度出发）

（我当时回答这个问题的思路我想应该对了，我说是通过反射加配置文件来做的）

由于这个问题研究起来可以另外写一篇专门的模块，这里只列出相对简单的流程，后续会希望有时间整理出具体的细节：首先，Struts2 是基于 Xwork 框架的，如果你有仔细看过 Xwork 的文档，你会发现，它的初始化过程基于以下几个类：Configuring XWork2 centers around the following classes:- 1. ConfigurationManager 2. ConfigurationProvider 3. Configuration 而在 ConfigurationProvider 的实现类 XmlConfigurationProvider 的内部，你可以看到下面的代码

Java 代码 


```

<SPAN style="FONT-WEIGHT: normal"> public XmlConfigurationProvider() {

    this("xwork.xml", true);

} </SPAN>

```

Java 代码 

```


<SPAN style="FONT-WEIGHT: normal"> public XmlConfigurationProvider() {

    this("xwork.xml", true);

} </SPAN>

```

同样的，Struts2 的初始化也是这样的一个类，只不过它继承于 Xwork 原有的类，并针对 Struts2 做了一些特别的定制。

Java 代码 

```

1. <SPAN style="FONT-WEIGHT: normal">public class StrutsXmlConfigurationProvider
2.     extends XmlConfigurationProvider {
3.     public StrutsXmlConfigurationProvi

```

1. 在一次歌唱竞争中，每一名参赛选手都有评委投了优秀票。如果上述断定为真，则以下哪项不可能为真？1) 有的评委投了所有参赛选手优秀票。2) 有的评委没有给任何参赛选手投优秀票。3) 有的参赛选手没有得到一张优秀票。

- A. 只有 1)。
- B. 只有 1)。
- C. 只有 3)。
- D. 只有 1) 和 2)。
- E. 只有 1) 和 3)。

2. 所有通过英语六级考试的学生都参加了学校的英语俱乐部，王进参加了英语俱乐部，所以他一定通过了英语六级考试。以下哪项最好的指出了上述论证的逻辑错误？

- A. 部分通过英语六级考试的学生没有参加英语俱乐部
- B. 王进能够参加英语俱乐部是因为它符合加入俱乐部的基本条件。
- C. 王进曾经获得过年级英语演讲比赛第一名。
- D. 凡愿意每学期缴纳 50 元会费，并且愿意积极参加俱乐部活动的学生都可以成为俱乐部的成员。
- E. 有些参加俱乐部的学生还没有通过英语六级考试。

3. 一架飞机在满油的情况下可以绕地球飞 0.5 圈，假设飞机与飞机之间可以互相加油，请问在确保所有飞机够油飞回起点的情况下，最少需要几架飞机才可以让其中一架飞机成功绕地球飞行一圈？

- A. 3
- B. 4
- C. 5
- D. 6
- E. 7

4. 如果所有的妇女都有大衣，那么漂亮的妇女会有？

- A. 更多的大衣
- B. 时髦的大衣
- C. 大衣
- D. 昂贵的大衣

5. 100 张多米诺骨牌整齐的排成一列，顺序编号依次为 1, 2, 3, ……，99, 100。第一次拿走所有奇数位置上的骨牌，第二次再从剩余骨牌中拿走所有奇数位置上的骨牌，依次类推。请问最后剩下的一张骨牌的编号是多少？

- A. 32
- B. 64
- C. 88
- D. 96

6. 小王在商店买衬衫，售货员问她想要哪种颜色的，小王幽默的说：“我不像讨厌黄色那样讨厌红色，我不像讨厌白色那样讨厌蓝色，我不像喜欢粉红那样喜欢红色，我对蓝色不如对黄色那样喜欢。”小王最后会选择的颜色是：

- A. 粉色
- B. 蓝色
- C. 红色
- D. 黄色

7. 在我国北方严寒冬季的夜晚，车辆前挡风玻璃会因低温而结冰霜。第二天对车辆发动预热后玻璃上的冰霜会很快融化。何宁对此不解，李军解释道：因为车辆仅有除霜孔位于前挡风玻璃，而车辆预热后除霜孔完全开启，因此，是开启除霜孔是车辆玻璃冰霜融化。以下哪项为真，最能质疑李军对车辆玻璃迅速融化的解释？

- A. 车辆一侧玻璃窗没有出现冰霜现象。
- B. 尽管车位玻璃窗没有除霜孔，其玻璃上的冰霜融化速度与挡风玻璃没有差别
- C. 当吹在车辆玻璃上的空气气温增加，其冰霜的融化速度也会增加
- D. 车辆前挡风玻璃除霜孔排出的暖气流排除后可能很快冷却
- E. 即使启用车内空调暖风功能，除霜孔的功能也不能被取代

8. 小张承诺：如果天不下雨，我一定去听音乐会。以下哪项为真，说明小张没有兑现承诺？

- 1) 天没下雨，小张没去听音乐会。2) 天下雨，小张去听了音乐会。3) 天下雨，小张没去

听音乐会。

- A. 仅 1)。
- B. 仅 2)。
- C. 仅 3)。
- D. 仅 1) 和 2)。
- E. 1)、2) 和 3)。

9. 某零件加工厂按工人完成的合格零件和不合格零件支付工资。公认每做一个合格零件得工资 10 元，每做一个不合格零件被扣除 5 元。已知某人一天工作了 12 个零件得工资 90 元。那么他在这一天做了多少个不合格零件？

- A. 2
- B. 3
- C. 4
- D. 6

10. 给你 8 颗小石头和一架天平，其中有 7 颗石头重量一样，另外一个比这 7 颗略重。请问在最坏情况下，最少要称重几次，才能把这颗较重的石头找出来？

- A. 3
- B. 2
- C. 1
- D. 4

11. 如果你有两个大小一样的桶，分别装了半桶红颜料和半桶蓝颜料。如果我们从蓝色颜料桶里舀一杯，倒入红色颜料桶里，搅拌均匀，然后再从红色颜料桶里舀一杯倒入蓝色颜料桶。请问以下说法哪种正确？

- A. 红色桶中蓝颜色的比例大。
- B. 蓝色桶中红颜色的比例大。
- C. 红色桶中蓝颜色的比例和蓝色桶中红颜色的比例一样大。

12. 甲乙丙三人居一学生宿舍。甲报案遗失 2000 元。保安人员经过周密调查，得出结论是丙作的案。班主任说：“这是最不可能的。”保安人员说：“当所有其他的可能性都被排除了，剩下的可能性不管看来多么不可能，都一定是事实。”以下哪项如果是真，将最为有力的动摇保安人员的结论？

- A. 保安人员事实上不可能比班主任更了解学生。
- B. 对违法行为惩处的根据，不能是逻辑推理，而只能是证据。
- C. 保安人员无法穷尽的把握所有的可能性。
- D. 丙是班上公认的品学兼优的学生。
- E. 乙有作案的前科。

13. 为了将当前目录下的归档文件 myftp.tgz 解压缩到/tmp 目录下, 用户可以使用命令

- A. tar xvzf myftp.tgz -C/tmp
- B. tar xvzf myftp.tgz -R/tmp
- C. tar vzf myftp.tgz -X/tmp
- D. tar xvzf myftp.tgz /tmp

14. 软件测试的对象包括 ()

- A. 目标程序和相关文档
- B. 源程序、目标程序、数据及相关文档
- C. 目标程序、操作系统和平台软件
- D. 源程序和目标程序

15. 正则表达式 ab? c 匹配的字符串是 ()。

- A. abcd
- B. adc
- C. aFdc
- D. aEbc

主观题

1. 假设有 Alibaba 网站最近一个月的查询日志, 记录了用户的查询行为。每条查询都至少包含有一个产品词, 称之为查询意图。总计有查询记录 3000 万条, 请统计出这 3000 万条……
2. 为了保护我们的地球, 全世界都在倡导绿色环保。在高效能计算和绿色计算方面, 请谈谈你的一些想法。

1 JDK5 支持泛型, 泛型是如何实现的, 应用反射 API 操作泛型变量时, 泛型还起作用吗?

2 Class.forName("oracle.ddd.driver"); 要 newInstance() 吗? new 出来的实例有什么作用? 为什么要 newInstance ()