<http://491569462-qq-com.iteye.com/blog/2054551>

**Java代码  [收藏代码](javascript:void())**

1. Byte和**int**之间的转换
2. /\*\*
3. \*将32位的int值放到4字节的里
4. \* @param num
5. \* @return
6. \*/
7. **public** **static** **byte**[] int2byteArray(**int** num) {
8. **byte**[] result = **new** **byte**[4];
9. result[0] = (**byte**)(num >>> 24);//取最高8位放到0下标
10. result[1] = (**byte**)(num >>> 16);//取次高8为放到1下标
11. result[2] = (**byte**)(num >>> 8); //取次低8位放到2下标
12. result[3] = (**byte**)(num );      //取最低8位放到3下标
13. **return** result;
14. }
16. /\*\*
17. \* 将4字节的byte数组转成一个int值
18. \* @param b
19. \* @return
20. \*/
21. **public** **static** **int** byteArray2int(**byte**[] b){
22. **byte**[] a = **new** **byte**[4];
23. **int** i = a.length - 1,j = b.length - 1;
24. **for** (; i >= 0; i--,j--) {//从b的尾部(即int值的低位)开始copy数据
25. **if**(j >= 0)
26. a[i] = b[j];
27. **else**
28. a[i] = 0;//如果b.length不足4,则将高位补0
29. }
30. **int** v0 = (a[0] & 0xff) << 24;//&0xff将byte值无差异转成int,避免Java自动类型提升后,会保留高位的符号位
31. **int** v1 = (a[1] & 0xff) << 16;
32. **int** v2 = (a[2] & 0xff) << 8;
33. **int** v3 = (a[3] & 0xff) ;
34. **return** v0 + v1 + v2 + v3;
35. }
36. **short**和**byte**的互转
38. /\*\*
39. \* 转换short为byte
40. \*
41. \* @param b
42. \* @param s 需要转换的short
43. \* @param index
44. \*/
45. **public** **static** **void** putShort(**byte** b[], **short** s, **int** index) {
46. b[index + 1] = (**byte**) (s >> 8);
47. b[index + 0] = (**byte**) (s >> 0);
48. }
50. /\*\*
51. \* 通过byte数组取到short
52. \*
53. \* @param b
54. \* @param index 第几位开始取
55. \* @return
56. \*/
57. **public** **static** **short** getShort(**byte**[] b, **int** index) {
58. **return** (**short**) (((b[index + 1] << 8) | b[index + 0] & 0xff));
59. }
60. **byte**和**char**类型的转换
62. /\*\*
63. \* 字符到字节转换
64. \*
65. \* @param ch
66. \* @return
67. \*/
68. **public** **static** **void** putChar(**byte**[] bb, **char** ch, **int** index) {
69. **int** temp = (**int**) ch;
70. // byte[] b = new byte[2];
71. **for** (**int** i = 0; i < 2; i ++ ) {
72. // 将最高位保存在最低位
73. bb[index + i] = **new** Integer(temp & 0xff).byteValue();
74. temp = temp >> 8; // 向右移8位
75. }
76. }
78. /\*\*
79. \* 字节到字符转换
80. \*
81. \* @param b
82. \* @return
83. \*/
84. **public** **static** **char** getChar(**byte**[] b, **int** index) {
85. **int** s = 0;
86. **if** (b[index + 1] > 0)
87. s += b[index + 1];
88. **else**
89. s += 256 + b[index + 0];
90. s \*= 256;
91. **if** (b[index + 0] > 0)
92. s += b[index + 1];
93. **else**
94. s += 256 + b[index + 0];
95. **char** ch = (**char**) s;
96. **return** ch;
97. }
98. **byte**和**float**的转换
100. /\*\*
101. \* float转换byte
102. \*
103. \* @param bb
104. \* @param x
105. \* @param index
106. \*/
107. **public** **static** **void** putFloat(**byte**[] bb, **float** x, **int** index) {
108. // byte[] b = new byte[4];
109. **int** l = Float.floatToIntBits(x);
110. **for** (**int** i = 0; i < 4; i++) {
111. bb[index + i] = **new** Integer(l).byteValue();
112. l = l >> 8;
113. }
114. }
116. /\*\*
117. \* 通过byte数组取得float
118. \*
119. \* @param bb
120. \* @param index
121. \* @return
122. \*/
123. **public** **static** **float** getFloat(**byte**[] b, **int** index) {
124. **int** l;
125. l = b[index + 0];
126. l &= 0xff;
127. l |= ((**long**) b[index + 1] << 8);
128. l &= 0xffff;
129. l |= ((**long**) b[index + 2] << 16);
130. l &= 0xffffff;
131. l |= ((**long**) b[index + 3] << 24);
132. **return** Float.intBitsToFloat(l);
133. }
134. **byte**和**double**转换
136. /\*\*
137. \* double转换byte
138. \*
139. \* @param bb
140. \* @param x
141. \* @param index
142. \*/
143. **public** **static** **void** putDouble(**byte**[] bb, **double** x, **int** index) {
144. // byte[] b = new byte[8];
145. **long** l = Double.doubleToLongBits(x);
146. **for** (**int** i = 0; i < 4; i++) {
147. bb[index + i] = **new** Long(l).byteValue();
148. l = l >> 8;
149. }
150. }
152. /\*\*
153. \* 通过byte数组取得float
154. \*
155. \* @param bb
156. \* @param index
157. \* @return
158. \*/
159. **public** **static** **double** getDouble(**byte**[] b, **int** index) {
160. **long** l;
161. l = b[0];
162. l &= 0xff;
163. l |= ((**long**) b[1] << 8);
164. l &= 0xffff;
165. l |= ((**long**) b[2] << 16);
166. l &= 0xffffff;
167. l |= ((**long**) b[3] << 24);
168. l &= 0xffffffffl;
169. l |= ((**long**) b[4] << 32);
170. l &= 0xffffffffffl;
171. l |= ((**long**) b[5] << 40);
172. l &= 0xffffffffffffl;
173. l |= ((**long**) b[6] << 48);
174. l &= 0xffffffffffffffl;
175. l |= ((**long**) b[7] << 56);
176. **return** Double.longBitsToDouble(l);
177. }
178. }