表 -1　字符串 API 和正则表达式结构的映射

Table -1　API-Regex-Mapping

[字符串 API 和正则表达式结构的映射]

序号	简要方法签名	APIpair		RegexWrapper				关系
		API 调用 (caller:str | array)	CondExpr	R	L	S	Lmin	
1	<String: boolean isEmpty()>	r=str.isEmpty()	r==T	[\s\S]{0}	-1	NULL	0	≡
2	<String: boolean equals(Object)>	r=str.equals(v)	r==T	v	-1	NULL	Len(v)	≡
3	<String: int compareTo(String)>	r=str.compareTo(v)	r==0	v	-1	NULL	Len(v)	≡
4	<String: int compareTo(Object)>	r=str.compareTo(v)	r==0	v	-1	NULL	Len(v)	≡
5	<String: boolean contentEquals(CharSequence)>	r=str.contentEquals(v)	r==T	v	-1	NULL	Len(v)	≡
6	<String: boolean contentEquals(StringBuffer)>	r=str.contentEquals(v)	r==T	v	-1	NULL	Len(v)	≡
7	<String: int compareToIgnoreCase(String)>	r=str.compareToIgnoreCase(v)	r==0	v	-1	NULL	Len(v)	⊊
8	<String: boolean equalsIgnoreCase(String)>	r=str.equalsIgnoreCase(v)	r==T	v	-1	NULL	Len(v)	⊊
9	<String: boolean contains(CharSequence)>	r=str.contains(v)	r==T	v[\s\S]*	-1	NULL	Len(v)	⊊
10	<String: boolean startsWith(String)>	r=str.startsWith(v)	r==T	v[\s\S]*	-1	NULL	Len(v)	≡
11	<String: boolean startsWith(String,int)>	r=str.startsWith(v,i_1)	r==T	[\s\S]{i_1}v[\s\S]*	-1	NULL	i_1 + Len(v)	≡
12	<String: boolean endsWith(String)>	r=str.endsWith(v)	r==T	[\s\S]*v	-1	NULL	Len(v)	≡
13	<String: int indexOf(String)>	r=str.indexOf(v)	r==i_1	[^v]{i_1}v[\s\S]*	-1	NULL	i_1 + Len(v)	⊊
14	<StringBuilder: int indexOf(String)>	r=str.indexOf(v)	r>=i_1	[^v]{i_1,}v[\s\S]*	-1	NULL	i_1 + Len(v)	⊊
15	<StringBuffer: int indexOf(String)>	r=str.indexOf(v)	r<=i_1	[^v]{0,i_1}v[\s\S]*	-1	NULL	Len(v)	⊊
16	<String: boolean regionMatches (boolean,int,String,int,int)>	r=str.regionMatches(b_1,i_1,v_1,i_2,i_3)	r==T	[\s\S]{i_1}subStr[\s\S]* subStr=v_1.substring(i_2,i_2+i_3)	-1	NULL	i_1 + i_3	⊊
17	<String: boolean regionMatches (int,String,int,int)>	r=str.regionMatches(i_1,v_1,i_2,i_3)	r==T	[\s\S]{i_1}subStr[\s\S]* subStr=v_1.substring(i_2,i_2+i_3)	-1	NULL	i_1 + i_3	≡

字符串 API 和正则表达式结构的映射 (续 1)

API-Regex-Mapping (continued)

序号	简要方法签名	APIpair		RegexWrapper				关系
		API 调用 (caller:str | array)	CondExpr	R	L	S	Lmin	
18	<String: boolean matches(String)>	r=str.matches(v)	r==T	v	-1	NULL	$L_{m}in$(v)	≡
19	<String: int indexOf(int)>	r=str.indexOf(v)	r==i_1	[^v]{i_1}v[\s\S]*	-1	NULL	i_1 + Len(v)	≡
		r=str.indexOf(v)	r>=i_1	[^v]{i_1,}v[\s\S]*	-1	NULL	i_1 + Len(v)	≡
		r=str.indexOf(v)	r<=i_1	[^v]{0,i_1}v[\s\S]*	-1	NULL	Len(v)	≡
				首先将 v 转成字符串形式				
20	<String: int indexOf(String,int)>	r=str.indexOf(v,i_1)	r==i_2	[\s\S]{i_1}[^v]{i_2-i_1}v[\s\S]*	-1	NULL	i_2 + Len(v)	$\subsetneq$
21	<StringBuffer: int indexOf(String,int)>	r=str.indexOf(v,i_1)	r>=i_2	[\s\S]{i_1}[^v]{i_2-i_1,}v[\s\S]*	-1	NULL	i_2 + Len(v)	$\subsetneq$
22	<StringBuilder: int indexOf(String,int)>	r=str.indexOf(v,i_1)	r<=i_2	[\s\S]{i_1}[^v]{0,i_2-i_1}v[\s\S]*	-1	NULL	i_1 + Len(v)	$\subsetneq$
23	<String: int indexOf(int,int)>	r=str.indexOf(v,i_1)	r==i_2	[\s\S]{i_1}[^v]{i_2-i_1}v[\s\S]*	-1	NULL	i_2 + Len(v)	≡
		r=str.indexOf(v,i_1)	r>=i_2	[\s\S]{i_1}[^v]{i_2-i_1,}v[\s\S]*	-1	NULL	i_2 + Len(v)	≡
		r=str.indexOf(v,i_1)	r<=i_2	[\s\S]{i_1}[^v]{0,i_2-i_1}v[\s\S]*	-1	NULL	i_1 + Len(v)	≡
				首先将 v 转成字符串形式				
24	<String: int lastIndexOf(String)>	r=str.lastIndexOf(v)	r==i_1	[\s\S]{i_1}v[^v]*	-1	NULL	i_1 + Len(v)	$\subsetneq$
25	<StringBuffer: int lastIndexOf(String)>	r=str.lastIndexOf(v)	r>=i_1	[\s\S]{i_1,}v[^v]*	-1	NULL	i_1 + Len(v)	$\subsetneq$
26	<StringBuilder: int lastIndexOf(String)>	r=str.lastIndexOf(v)	r<=i_1	[\s\S]{0,i_1}v[^v]*	-1	NULL	Len(v)	$\subsetneq$
27	<String: int lastIndexOf(int)>	r=str.lastIndexOf(v)	r==i_1	[\s\S]{i_1}v[^v]*	-1	NULL	i_1 + Len(v)	≡
		r=str.lastIndexOf(v)	r>=i_1	[\s\S]{i_1,}v[^v]*	-1	NULL	i_1 + Len(v)	≡
		r=str.lastIndexOf(v)	r<=i_1	[\s\S]{0,i_1}v[^v]*	-1	NULL	Len(v)	≡
				首先将 v 转成字符串形式				

字符串 API 和正则表达式结构的映射 (续 2)

API-Regex-Mapping(continued)

序号	简要方法签名	APIpair		RegexWrapper				关系
		API 调用 (caller:str | array)	CondExpr	R	L	S	Lmin	
28	<String: int lastIndexOf(String,int)>	r=str.lastIndexOf(v,i_1)	r==i_2	[\s\S]{i_2}v[^v]{i_1+1-i_2-Len(v)}[\s\S]*	-1	NULL	$i_1 + 1$	$\subsetneq$
29	<StringBuffer: int lastIndexOf(String,int)>	r=str.lastIndexOf(v,i_1)	r>=i_2	[\s\S]{i_2,}v[^v]{i_1+1-i_2-Len(v)}[\s\S]*	-1	NULL	$i_1 + 1$	$\subsetneq$
30	<StringBuilder: int lastIndexOf(String,int)>	r=str.lastIndexOf(v,i_1)	r<=i_2	[\s\S]{0,i_2}v[^v]{i_1+1-i_2-Len(v)}[\s\S]*	-1	NULL	$i_1 - i_2 + 1$	$\subsetneq$
31	<String: int lastIndexOf(int,int)>	r=str.lastIndexOf(v,i_1)	r==i_2	[\s\S]{i_2}v[^v]{i_1+1-i_2-Len(v)}[\s\S]*	-1	NULL	$i_1 + 1$	≡
		r=str.lastIndexOf(v,i_1)	r>=i_2	[\s\S]{i_2,}v[^v]{i_1+1-i_2-Len(v)}[\s\S]*	-1	NULL	$i_1 + 1$	$\subsetneq$
		r=str.lastIndexOf(v,i_1)	r<=i_2	[\s\S]{0,i_2}v[^v]{i_1+1-i_2-Len(v)}[\s\S]*	-1	NULL	$i_1 - i_2 + 1$	$\subsetneq$
32	<String: char charAt(int)>	r=str.charAt(i_1)	r==i_2	[\s\S]{i_1}i_2[\s\S]*	-1	NULL	$i_1 + 1$	≡
33	<StringBuffer: char charAt(int)>	r=str.charAt(i_1)	r>=i_2	[\s\S]{i_1}[i_2+1][\s\S]*	-1	NULL	$i_1 + 1$	$\subsetneq$
34	<StringBuilder: char charAt(int)>	r=str.charAt(i_1)	r<=i_2	[\s\S]{i_1}[i_2-1][\s\S]*	-1	NULL	$i_1 + 1$	$\subsetneq$
35	<String: char codePointAt(int)>	r=str.codePointAt(i_1)	r==i_2	[\s\S]{i_1}i_2[\s\S]*	-1	NULL	$i_1 + 1$	≡
36	<StringBuffer: char codePointAt(int)>	r=str.codePointAt(i_1)	r>=i_2	[\s\S]{i_1}[i_2+1][\s\S]*	-1	NULL	$i_1 + 1$	$\subsetneq$
37	<StringBuilder: char codePointAt(int)>	r=str.codePointAt(i_1)	r<=i_2	[\s\S]{i_1}[i_2-1][\s\S]*	-1	NULL	$i_1 + 1$	$\subsetneq$
38	<String: int codePointCount(int,int)>	r=str.codePointCount(i_1,i_2)	r==i_3	[\s\S]{i_2,}	-1	NULL	i_2	≡
39	<StringBuffer: int codePointCount(int,int)>	r=str.codePointCount(i_1,i_2)	r>=i_3	[\s\S]{i_2,}	-1	NULL	i_2	$\subsetneq$
40	<StringBuilder: int codePointCount(int,int)>	r=str.codePointCount(i_1,i_2)	r<=i_3	[\s\S]{i_2,}	-1	NULL	i_2	$\subsetneq$
41	<String: int codePointBefore(int)>	r=str.codePointBefore(i_1)	r==i_2	[\s\S]{i_1-1}i_2[\s\S]*	-1	NULL	i_1	≡
42	<StringBuffer: int codePointBefore(int)>	r=str.codePointBefore(i_1)	r>=i_2	[\s\S]{i_1-1}[i_2+1][\s\S]*	-1	NULL	i_1	$\subsetneq$
43	<StringBuilder: int codePointBefore(int)>	r=str.codePointBefore(i_1)	r<=i_2	[\s\S]{i_1-1}[i_2-1][\s\S]*	-1	NULL	i_2	$\subsetneq$
				首先将 i_2-1,i_2+1,i_2 转成字符串				

字符串 API 和正则表达式结构的映射 (续 3)
API-Regex-Mapping(continued)

序号	简要方法签名	APIpair		RegexWrapper				关系
		API 调用 (caller:str | array)	CondExpr	R	L	S	Lmin	
44	<String: int offsetByCodePoints(int,int)>	r=str.offsetByCodePoints(i_1,i_2)	r==i_3 (>=,<=,>,<)	[\s\S]{i_1+i_2,}	-1	NULL	i_1+i_2	≡
45	<StringBuffer: int offsetByCodePoints(int,int)>							
46	<StringBuilder: int offsetByCodePoints(int,int)>							
47	<String: int length()>	r=str.length()	r==i_1	[\s\S]{i_1}	-1	NULL	i_1	≡
48	<StringBuffer: int length()>	r=str.length()	r>=i_1	[\s\S]{i_1,}	-1	NULL	i_1	≡
49	<StringBuilder: int length()>	r=str.length()	r<=i_1	[\s\S]{0,i_1}	-1	NULL	0	≡
50	<String: char[] toCharArray()>	r=str.toCharArray()	r[i]==v	[\s\S]{i}v[\s\S]*	-1	NULL	i+1	≡
		r=str.toCharArray()	r[i]>v	[\s\S]{i}[v+1][\s\S]*	-1	NULL	i+1	⊊
		r=str.toCharArray()	r[i]<v	[\s\S]{i}[v-1][\s\S]*	-1	NULL	i+1	⊊
				首先将 v+1,v-1 转成字符串				
51	<String: byte[] getBytes(String)>	r=str.getBytes(v1)	r[i]==v	[\s\S]{i}v[\s\S]*	-1	NULL	i+1	≡
		r=str.getBytes(v1)	r[i]>v	[\s\S]{i}[v+1][\s\S]*	-1	NULL	i+1	⊊
52	<String: byte[] getBytes(Charset)>	r=str.getBytes(v1)	r[i]<v	[\s\S]{i}[v-1][\s\S]*	-1	NULL	i+1	⊊
				首先将 v+1,v-1 转成字符串				
53	<String: byte[] getBytes()>	r=str.getBytes()	r[i]==v	[\s\S]{i}v[\s\S]*	-1	NULL	i+1	≡
		r=str.getBytes()	r[i]>v	[\s\S]{i}[v+1][\s\S]*	-1	NULL	i+1	⊊
		r=str.getBytes()	r[i]<v	[\s\S]{i}[v-1][\s\S]*	-1	NULL	i+1	⊊
				首先将 v+1,v-1 转成字符串				
54	<String: String trim()>	r=str.trim()	NULL	NULL	-1	[^]]	0	⊊

字符串 API 和正则表达式结构的映射 (续 4)

API-Regex-Mapping(continued)

序号	简要方法签名	**APIpair**		**RegexWrapper**				关系
		API 调用 (caller:str ∣ array)	CondExpr	**R**	**L**	**S**	**Lmin**	
55	<String: String substring(int)>	r=str.substring(i_1)	NULL	[\s\S]{i_1}	-1	NULL	i_1	≡
56	<StringBuffer: String substring(int)>							
57	<StringBuilder: String substring(int)>							
58	<String: String substring(int,int)>	r=str.substring(i_1,i_2)	NULL	[\s\S]{i_1}	i_2-i_1	NULL	i_1	⊊
59	<StringBuffer: String substring(int,int)>							
60	<StringBuilder: String substring(int,int)>							
61	<String: CharSequence subSequence(int,int)>	r=str.subSequence(i_1,i_2)	NULL	[\s\S]{i_1}	i_2-i_1	NULL	i_1	⊊
62	<StringBuffer: CharSequence subSequence(int,int)>							
63	<StringBuilder: CharSequence subSequence(int,int)>							
64	<String: String[] split(String)>	r=str.split(v)	r[i]	v{i}	-1	[^v]]	i	⊊
65	<String: String[] split(String,int)>	r=str.split(v,i_1)	r[i]	v{i}	-1	[^v]]	i	⊊
66	<String: String toLowerCase(Locale)>	r=str.toLowerCase(local)	NULL	NULL	-1	NULL	0	⊊
67	<String: String toLowerCase()>	r=str.toLowerCase()	NULL	NULL	-1	NULL	0	⊊
68	<String: String toUpperCase(Locale)>	r=str.toUpperCase(local)	NULL	NULL	-1	NULL	0	⊊
69	<String: String toUpperCase()>	r=str.toUpperCase()	NULL	NULL	-1	NULL	0	⊊
70	<String: String toString()>	r=str.toString()	NULL	NULL	-1	NULL	0	⊊
71	<StringBuffer: String toString()>							
72	<StringBuilder: String toString()>							
73	<String: String replace(char,char)>	r=str.replace(v_1,v_2)	NULL	NULL	-1	[^v_1]	0	⊊

字符串 API 和正则表达式结构的映射 (续 5)

API-Regex-Mapping(continued)

序号	简要方法签名	APIpair		RegexWrapper				关系
		API 调用 (caller:str ǀ array)	CondExpr	**R**	**L**	**S**	**Lmin**	
74	<String: String replace(CharSequence,CharSequence)>	r=str.replace(v_1,v_2)	NULL	NULL	-1	[^v_1]	0	$\subsetneq$
75	<String: String replaceAll(String,String)>	r=str.replaceAll(v_1,v_2)	NULL	NULL	-1	[^v_1]	0	$\subsetneq$
76	<String: String replaceFirst(String,String)>	r=str.replaceFirst(v_1,v_2)	NULL	NULL	-1	[^v_1]	0	$\subsetneq$
77 78	<StringBuffer: StringBuffer replace(int,int,String)> <StringBuilder: StringBuilder replace(int,int,String)>	r=str.replace(i_1,i_2,v_1)	NULL	NULL	-1	[\^{}v_2] v_2=str.substring(i_1,i_2)	0	$\subsetneq$
79	<String: String concat(String)>	r=str.concat(v_1)	NULL	NULL	-1	NULL	0	$\subsetneq$
80-93	<StringBuffer: StringBuffer append(...)>	r=str.append(v_1...v_n)	NULL	NULL	-1	NULL	0	$\subsetneq$
94-107	<StringBuilder: StringBuilder append(...)>	r=str.append(v_1...v_n)	NULL	NULL	-1	NULL	0	$\subsetneq$
108 (new add)	<Array: int length>	r=str.length	r==i	[\s\S]{i}	-1	NULL	i	≡
		r=str.length	r>=i	[\s\S]{i,}	-1	NULL	i	≡
		r=str.length	r<=i	[\s\S]{0,i}	-1	NULL	0	≡