

bcgit / bc-java Public mirror

mirrored from <https://www.bouncycastle.org/repositories/bc-java>

<> Code Issues 299 Pull requests 40 Discussions Actions Projects

Commit d20cdb8



dghgit committed on Jan 1

refactored out common code from LDAP classes

main · r1rv84

1 parent [e380617](#) commit d20cdb8

3 files changed +154 -158 lines changed

Top

Filter files...

- prov/src/main/java/org/bouncycastle
 - jce/provider
 - X509LDAPCertStoreSpi.java
 - ldap
 - LDAPUtils.java
 - x509/util
 - LDAPStoreHelper.java

3 files changed +154 -158 lines changed

Search within code

```

...stle/jce/provider/X509LDAPCertStoreSpi.java
@@ -34,6 +34,7 @@
34 34 import org.bouncycastle.asn1.ASN1InputStream;
35 35 import org.bouncycastle.asn1.x509.CertificatePair;
36 36 import org.bouncycastle.jce.X509LDAPCertStoreParameters;
37 + import org.bouncycastle.ldap.LDAPUtils;
37 38 import org.bouncycastle.util.Strings;

```

```

38 39
39 40 /**
@@ -50,26 +51,6 @@
50 51 public class X509LDAPCertStoreSpi
51 52     extends CertStoreSpi
52 53 {
53 - private static String[] FILTER_ESCAPE_TABLE = new String['\\' + 1];
54 -
55 - static
56 - {
57 -     // Filter encoding table -----
58 -
59 -     // fill with char itself
60 -     for (char c = 0; c < FILTER_ESCAPE_TABLE.length; c++)
61 -     {
62 -         FILTER_ESCAPE_TABLE[c] = String.valueOf(c);
63 -     }
64 -
65 -     // escapes (RFC2254)
66 -     FILTER_ESCAPE_TABLE['*'] = "\\2a";
67 -     FILTER_ESCAPE_TABLE['('] = "\\28";
68 -     FILTER_ESCAPE_TABLE[')'] = "\\29";
69 -     FILTER_ESCAPE_TABLE['\\'] = "\\5c";
70 -     FILTER_ESCAPE_TABLE[0] = "\\00";
71 - }
72 -
73 54 /**
74 55     * Initial Context Factory.
75 56     */
@@ -124,42 +105,6 @@ private DirContext connectLDAP()
124 105         return ctx;
125 106     }
126 107
127 - private String parseDN(String subject, String subjectAttributeName)
128 - {
129 -     String temp = subject;
130 -     int begin =
131 -         Strings.toLowerCase(temp).indexOf(Strings.toLowerCase(subjectAttributeName));

```

```

132     -         int end = temp.indexOf(',');
133     -         if (end == -1)
134     -         {
135     -             end = temp.length();
136     -         }
137     -         while (temp.charAt(end - 1) == '\\')
138     -         {
139     -             end = temp.indexOf(',', end + 1);
140     -             if (end == -1)
141     -             {
142     -                 end = temp.length();
143     -             }
144     -         }
145     -         temp = temp.substring(0, end);
146     -         begin = temp.indexOf('=');
147     -         temp = temp.substring(begin + 1);
148     -         if (temp.charAt(0) == ' ')
149     -         {
150     -             temp = temp.substring(1);
151     -         }
152     -         if (temp.startsWith("\\"))
153     -         {
154     -             temp = temp.substring(1);
155     -         }
156     -         if (temp.endsWith("\\"))
157     -         {
158     -             temp = temp.substring(0, temp.length() - 1);
159     -         }
160     -         return filterEncode(temp);
161     -     }
162
163     108         public Collection engineGetCertificates(CertSelector selector)
164     109             throws CertStoreException
165     110         {
166     111             @@ -277,7 +222,7 @@ private Set certSubjectSerialSearch(X509CertSelector
167     112             xselector,
168     113             subject = xselector.getSubjectAsString();
169     114         }
170     115     }
171     116     String attrValue = parseDN(subject, subjectAttributeName);

```

225	+	String attrValue = LDAPUtils.parseDN(subject, subjectAttributeName);
281	226	set.addAll(search(attrName, "*" + attrValue + "*", attrs));
282	227	if (serial != null
283	228	&& params.getSearchForSerialNumberIn() != null)
⋮		@@ -374,13 +319,13 @@ public Collection engineGetCRLs(CRLSelector selector)
374	319	{
375	320	String issuerAttributeName = params
376	321	.getCertificateRevocationListIssuerAttributeName();
377	-	attrValue = parseDN((String)o, issuerAttributeName);
322	+	attrValue = LDAPUtils.parseDN((String)o, issuerAttributeName);
378	323	}
379	324	else
380	325	{
381	326	String issuerAttributeName = params
382	327	.getCertificateRevocationListIssuerAttributeName();
383	-	attrValue = parseDN(new X500Principal((byte[])o)
328	+	attrValue = LDAPUtils.parseDN(new X500Principal((byte[])o)
384	329	.getName("RFC1779"), issuerAttributeName);
385	330	}
386	331	set.addAll(search(attrName, "*" + attrValue + "*", attrs));
⋮		@@ -415,43 +360,7 @@ public Collection engineGetCRLs(CRLSelector selector)
415	360	
416	361	return crlSet;
417	362	}
418	-	
419	-	/**
420	-	* Escape a value for use in a filter.
421	-	*
422	-	* @param value the value to escape.
423	-	* @return a properly escaped representation of the supplied value.
424	-	*/
425	-	private String filterEncode(String value)
426	-	{
427	-	if (value == null)
428	-	{
429	-	return null;

```

430     -     }
431     -
432     -     // make buffer roomy
433     -     StringBuilder encodedValue = new StringBuilder(value.length() * 2);
434     -
435     -     int length = value.length();
436     -
437     -     for (int i = 0; i < length; i++)
438     -     {
439     -         char c = value.charAt(i);
440     -
441     -         if (c < FILTER_ESCAPE_TABLE.length)
442     -         {
443     -             encodedValue.append(FILTER_ESCAPE_TABLE[c]);
444     -         }
445     -         else
446     -         {
447     -             // default: add the char
448     -             encodedValue.append(c);
449     -         }
450     -     }
451     -
452     -     return encodedValue.toString();
453     - }
454     -

```

363

+

```

455 364     /**
456 365     * Returns a Set of byte arrays with the certificate or CRL encodings.
457 366     *

```



▼ ...n/java/org/bouncycastle/ldap/LDAPUtils.java

...

...

@@ -0,0 +1,112 @@

```

1 + package org.bouncycastle.ldap;
2 +
3 + import org.bouncycastle.util.Strings;
4 +
5 + /**
6 + * General utility methods for assisting with preparation of LDAP queries.
7 + */

```

```
8 + public class LDAPUtils
9 + {
10 +     private static String[] FILTER_ESCAPE_TABLE = new String['\\' + 1];
11 +
12 +     static
13 +     {
14 +         // Filter encoding table -----
15 +
16 +         // fill with char itself
17 +         for (char c = 0; c < FILTER_ESCAPE_TABLE.length; c++)
18 +         {
19 +             FILTER_ESCAPE_TABLE[c] = String.valueOf(c);
20 +         }
21 +
22 +         // escapes (RFC2254)
23 +         FILTER_ESCAPE_TABLE['*'] = "\\2a";
24 +         FILTER_ESCAPE_TABLE['('] = "\\28";
25 +         FILTER_ESCAPE_TABLE[')'] = "\\29";
26 +         FILTER_ESCAPE_TABLE['\\'] = "\\5c";
27 +         FILTER_ESCAPE_TABLE[0] = "\\00";
28 +     }
29 +
30 +     /**
31 +      * Parse out the contents of a particular subject attribute name from the
32 +      * string form of an X.500 DN.
33 +      *
34 +      * @param subject string form of an X.500 DN.
35 +      * @param subjectAttributeName the RDN attribute name of interest.
36 +      * @return an escaped string suitable for use in an LDAP query.
37 +      */
38 +     public static String parseDN(String subject, String subjectAttributeName)
39 +     {
40 +         String temp = subject;
41 +         int begin =
42 +             Strings.toLowerCase(temp).indexOf(Strings.toLowerCase(subjectAttributeName));
43 +         if (begin == -1)
44 +         {
45 +             return "";
46 +         }
47 +         temp = temp.substring(begin + subjectAttributeName.length());
```

```
46 +     int end = temp.indexOf(',');
47 +     if (end == -1)
48 +     {
49 +         end = temp.length();
50 +     }
51 +     while (temp.charAt(end - 1) == '\\')
52 +     {
53 +         end = temp.indexOf(',', end + 1);
54 +         if (end == -1)
55 +         {
56 +             end = temp.length();
57 +         }
58 +     }
59 +     temp = temp.substring(0, end);
60 +     begin = temp.indexOf('=');
61 +     temp = temp.substring(begin + 1);
62 +     if (temp.charAt(0) == ' ')
63 +     {
64 +         temp = temp.substring(1);
65 +     }
66 +     if (temp.startsWith("\\"))
67 +     {
68 +         temp = temp.substring(1);
69 +     }
70 +     if (temp.endsWith("\\"))
71 +     {
72 +         temp = temp.substring(0, temp.length() - 1);
73 +     }
74 +     return filterEncode(temp);
75 + }
76 +
77 + /**
78 +  * Escape a value for use in a filter.
79 +  *
80 +  * @param value the value to escape.
81 +  * @return a properly escaped representation of the supplied value.
82 +  */
83 + private static String filterEncode(String value)
84 + {
85 +     if (value == null)
```

```
86 +     {
87 +         return null;
88 +     }
89 +
90 +     // make buffer roomy
91 +     StringBuilder encodedValue = new StringBuilder(value.length() * 2);
92 +
93 +     int length = value.length();
94 +
95 +     for (int i = 0; i < length; i++)
96 +     {
97 +         char c = value.charAt(i);
98 +
99 +         if (c < FILTER_ESCAPE_TABLE.length)
100 +         {
101 +             encodedValue.append(FILTER_ESCAPE_TABLE[c]);
102 +         }
103 +         else
104 +         {
105 +             // default: add the char
106 +             encodedValue.append(c);
107 +         }
108 +     }
109 +
110 +     return encodedValue.toString();
111 + }
112 + }
```

▼ ...bouncycastle/x509/util/LDAPStoreHelper.java ...



@@ -35,6 +35,7 @@

```
35 35 import org.bouncycastle.jce.provider.X509CRLParser;
36 36 import org.bouncycastle.jce.provider.X509CertPairParser;
37 37 import org.bouncycastle.jce.provider.X509CertParser;
38 + import org.bouncycastle.ldap.LDAPUtils;
38 39 import org.bouncycastle.util.StoreException;
39 40 import org.bouncycastle.util.Strings;
40 41 import org.bouncycastle.x509.X509AttributeCertStoreSelector;
```



@@ -65,7 +66,6 @@



```
65 66 */
```

```

66 67 public class LDAPStoreHelper
67 68 {
68 -
69 69 // TODO: cache results
70 70
71 71 private X509LDAPCertStoreParameters params;
@@ -95,7 +95,8 @@ public LDAPStoreHelper(X509LDAPCertStoreParameters
params)
95 95 */
96 96 private static final String URL_CONTEXT_PREFIX = "com.sun.jndi.url";
97 97
98 - private DirContext connectLDAP() throws NamingException
98 + private DirContext connectLDAP()
99 + throws NamingException
99 100 {
100 101 Properties props = new Properties();
101 102 props.setProperty(Context.INITIAL_CONTEXT_FACTORY, LDAP_PROVIDER);
@@ -111,47 +112,6 @@ private DirContext connectLDAP() throws
NamingException
111 112 return ctx;
112 113 }
113 114
114 - private String parseDN(String subject, String dnAttributeName)
115 - {
116 - String temp = subject;
117 - int begin = Strings.toLowerCase(temp).indexOf(
118 - Strings.toLowerCase(dnAttributeName) + "=");
119 - if (begin == -1)
120 - {
121 - return "";
122 - }
123 - temp = temp.substring(begin + dnAttributeName.length());
124 - int end = temp.indexOf(',');
125 - if (end == -1)
126 - {
127 - end = temp.length();
128 - }
129 - while (temp.charAt(end - 1) == '\\')
130 - {
131 - end = temp.indexOf(',', end + 1);

```

```

132     -         if (end == -1)
133     -         {
134     -             end = temp.length();
135     -         }
136     -     }
137     -     temp = temp.substring(0, end);
138     -     begin = temp.indexOf('=');
139     -     temp = temp.substring(begin + 1);
140     -     if (temp.charAt(0) == ' ')
141     -     {
142     -         temp = temp.substring(1);
143     -     }
144     -     if (temp.startsWith("\\"))
145     -     {
146     -         temp = temp.substring(1);
147     -     }
148     -     if (temp.endsWith("\\"))
149     -     {
150     -         temp = temp.substring(0, temp.length() - 1);
151     -     }
152     -     return temp;
153     - }
154     -
155     115     private Set createCerts(List list, X509CertStoreSelector xselector)
156     116         throws StoreException
157     117     {
158     118         @@ -224,7 +184,7 @@ private List
159     119         certSubjectSerialSearch(X509CertStoreSelector xselector,
160     120
161     121         {
162     122             for (int i = 0; i < subjectAttributeNames.length; i++)
163     123             {
164     124                 attrValue = parseDN(subject, subjectAttributeNames[i]);
165     125                 + attrValue = LDAPUtils.parseDN(subject,
166     126                 subjectAttributeNames[i]);
167     127                 list
168     128                 .addAll(search(attrNames, "*" + attrValue + "*",
169     129                 attrs));
170     130
171     131         @@ -235,7 +195,7 @@ private List
172     132         certSubjectSerialSearch(X509CertStoreSelector xselector,
173     133
174     134         attrValue = serial;

```

236	196	list.addAll(search(
237	197	splitString(params.getSearchForSerialNumberIn()),
238	-	attrValue, attrs));
198	+	attrValue, attrs));
239	199	}
240	200	if (serial == null && subject == null)
241	201	{
↕		@@ -246,7 +206,6 @@ private List certSubjectSerialSearch(X509CertStoreSelector xselector,
246	206	}
247	207	
248	208	
249	-	
250	209	/**
251	210	* Can use the subject of the forward certificate of the set certificate
252	211	* pair or the subject of the forward
↕		@@ -290,7 +249,7 @@ private List crossCertificatePairSubjectSearch(
290	249	{
291	250	for (int i = 0; i < subjectAttributeNames.length; i++)
292	251	{
293	-	attrValue = parseDN(subject, subjectAttributeNames[i]);
252	+	attrValue = LDAPUtils.parseDN(subject, subjectAttributeNames[i]);
294	253	list
295	254	.addAll(search(attrNames, "\"" + attrValue + "\",
296	255	attrs));
↕		@@ -385,7 +344,7 @@ private List attrCertSubjectSerialSearch(
385	344	{
386	345	for (int i = 0; i < subjectAttributeNames.length; i++)
387	346	{
388	-	attrValue = parseDN(subject, subjectAttributeNames[i]);
347	+	attrValue = LDAPUtils.parseDN(subject, subjectAttributeNames[i]);
389	348	list
390	349	.addAll(search(attrNames, "\"" + attrValue + "\",
391	350	attrs));
↕		@@ -441,11 +400,11 @@ private List cRLIssuerSearch(X509CRLStoreSelector xselector,

```

441 400     if (xselector.getAttrCertificateChecking() != null)
442 401     {
443 402         Principal principals[] =
xselector.getAttrCertificateChecking().getIssuer().getPrincipals();
444 -         for (int i=0; i<principals.length; i++)
403 +         for (int i = 0; i < principals.length; i++)
445 404     {
446 405         if (principals[i] instanceof X500Principal)
447 406     {
448 -         issuers.add(principals[i]);
407 +         issuers.add(principals[i]);
449 408     }
450 409     }
451 410     }
@@ -457,7 +416,7 @@ private List cRLIssuerSearch(X509CRLStoreSelector
xselector,
457 416
458 417         for (int i = 0; i < issuerAttributeNames.length; i++)
459 418     {
460 -         attrValue = parseDN(issuer, issuerAttributeNames[i]);
419 +         attrValue = LDAPUtils.parseDN(issuer,
issuerAttributeNames[i]);
461 420         list
462 421             .addAll(search(attrNames, "*" + attrValue + "*",
463 422                 attrs));
@@ -485,7 +444,8 @@ private List cRLIssuerSearch(X509CRLStoreSelector
xselector,
485 444         *
directory.
486 445         */
487 446     private List search(String attributeNames[], String attributeValue,
488 -         String[] attrs) throws StoreException
447 +         String[] attrs)
448 +         throws StoreException
489 449     {
490 450         String filter = null;
491 451         if (attributeNames == null)
@@ -599,7 +559,8 @@ private Set createCRLs(List list,
X509CRLStoreSelector xselector)
599 559     }
600 560

```

```

601 561     private Set createCrossCertificatePairs(List list,
602 -      X509CertPairStoreSelector
      xselector) throws StoreException
603 +      X509CertPairStoreSelector
      xselector)
604 +      throws StoreException
603 564     {
604 565         Set certPairSet = new HashSet();
605 566
@@ -626,7 +587,7 @@ private Set createCrossCertificatePairs(List list,
626 587         pair = new X509CertificatePair(new CertificatePair(
627 588             Certificate
628 589                 .getInstance(new ASN1InputStream(
629 -                 forward).readObject()),
630 +                 forward).readObject()),
631 +                 Certificate
632 591                 .getInstance(new ASN1InputStream(
633 592                     .getInstance(new ASN1InputStream(
634 593                         reverse).readObject()))));
@@ -652,7 +613,8 @@ private Set createCrossCertificatePairs(List list,
652 613     }
653 614
654 615     private Set createAttributeCertificates(List list,
655 -      X509AttributeCertStoreSelector
      xselector) throws StoreException
656 +      X509AttributeCertStoreSelector
      xselector)
657 +      throws StoreException
656 618     {
657 619         Set certSet = new HashSet();
658 620
@@ -719,12 +681,14 @@ public Collection
@@ -719,12 +681,14 @@ public Collection
getAuthorityRevocationLists(X509CRLStoreSelector selector)
719 681     * The attributeCertificateRevocationList holds a list of attribute
720 682     * certificates that have been revoked.
721 683     * </p>
722 684     + *
723 685     * @param selector The CRL selector to use to find the CRLs.
724 686     * @return A possible empty collection with CRLs.
725 687     * @throws StoreException
726 688     */

```

726	689		<code>public Collection</code>	<code>getAttributeCertificateRevocationLists(</code>
727	-		<code>X509CRLStoreSelector selector)</code>	<code>throws StoreException</code>
	690	+	<code>X509CRLStoreSelector selector)</code>	
	691	+	<code>throws StoreException</code>	
728	692		{	
729	693		String[] attrs = <code>splitString(params</code>	
730	694		<code>.getAttributeCertificateRevocationListAttribute());</code>	
			@@ -754,12 +718,14 @@	<code>public Collection</code>
				<code>getAttributeCertificateRevocationLists(</code>
754	718		* The attributeAuthorityList holds a list of AA certificates that have	
			been	
755	719		* revoked.	
756	720		* </p>	
	721	+	*	
757	722		* @param selector The CRL selector to use to find the CRLs.	
758	723		* @return A possible empty collection with CRLs	
759	724		* @throws StoreException	
760	725		*/	
761	726		<code>public Collection</code>	<code>getAttributeAuthorityRevocationLists(</code>
762	-		<code>X509CRLStoreSelector selector)</code>	<code>throws StoreException</code>
	727	+	<code>X509CRLStoreSelector selector)</code>	
	728	+	<code>throws StoreException</code>	
763	729		{	
764	730		String[] attrs =	
			<code>splitString(params.getAttributeAuthorityRevocationListAttribute());</code>	
765	731		String attrNames[] = <code>splitString(params</code>	
			@@ -789,7 +755,8 @@	<code>public Collection</code>
				<code>getAttributeAuthorityRevocationLists(</code>
789	755		* @throws StoreException	
790	756		*/	
791	757		<code>public Collection</code>	<code>getCrossCertificatePairs(</code>
792	-		<code>X509CertPairStoreSelector selector)</code>	<code>throws StoreException</code>
	758	+	<code>X509CertPairStoreSelector selector)</code>	
	759	+	<code>throws StoreException</code>	
793	760		{	
794	761		String[] attrs = <code>splitString(params.getAttributeAuthorityRevocationListAttribute());</code>	
795	762		String attrNames[] =	
			<code>splitString(params.getLdapCrossCertificateAttributeName());</code>	
			@@ -850,6 +817,7 @@	<code>public Collection</code>
				<code>getUserCertificates(X509CertStoreSelector selector)</code>

```

850 817 * <p>
851 818 * The aCertificate holds the privileges of an attribute authority.
852 819 * </p>
820 + *
853 821 * @param selector The selector to find the attribute certificates.
854 822 * @return A possible empty collection with attribute certificates.
855 823 * @throws StoreException
⌵
@@ -882,12 +850,14 @@ public Collection
⌴
getAACertificates(X509AttributeCertStoreSelector selector)
882 850 * authority and holds a description of the privilege and its delegation
883 851 * rules.
884 852 * </p>
853 + *
885 854 * @param selector The selector to find the attribute certificates.
886 855 * @return A possible empty collection with attribute certificates.
887 856 * @throws StoreException
888 857 */
889 858 public Collection getAttributeDescriptorCertificates(
890 - X509AttributeCertStoreSelector selector) throws StoreException
859 + X509AttributeCertStoreSelector selector)
860 + throws StoreException
891 861 {
892 862     String[] attrs =
splitString(params.getAttributeDescriptorCertificateAttribute());
893 863     String attrNames[] = splitString(params
⌵
@@ -916,6 +886,7 @@ public Collection getAttributeDescriptorCertificates(
⌴
916 886 * store self-issued certificates (if any) and certificates issued to
this
917 887 * CA by CAs in the same realm as this CA.
918 888 * </p>
889 + *
919 890 * @param selector The selector to find the certificates.
920 891 * @return A possible empty collection with certificates.
921 892 * @throws StoreException
⌵
@@ -948,7 +919,8 @@ public Collection
⌴
getCACertificates(X509CertStoreSelector selector)
948 919 * @throws StoreException
949 920 */
950 921 public Collection getDeltaCertificateRevocationLists(

```

951	-	X509CRLStoreSelector selector) throws StoreException
922	+	X509CRLStoreSelector selector)
923	+	throws StoreException
952	924	{
953	925	String[] attrs =
		splitString(params.getDeltaRevocationListAttribute());
954	926	String attrNames[] =
		splitString(params.getLdapDeltaRevocationListAttributeName());
		@@ -973,12 +945,14 @@ public Collection
		getDeltaCertificateRevocationLists(
973	945	* <p>
974	946	* The attributeCertificateAttribute holds the privileges of a user
975	947	* </p>
948	+	*
976	949	* @param selector The selector to find the attribute certificates.
977	950	* @return A possible empty collection with attribute certificates.
978	951	* @throws StoreException
979	952	*/
980	953	public Collection getAttributeCertificateAttributes(
981	-	X509AttributeCertStoreSelector selector) throws StoreException
954	+	X509AttributeCertStoreSelector selector)
955	+	throws StoreException
982	956	{
983	957	String[] attrs =
		splitString(params.getAttributeCertificateAttributeAttribute());
984	958	String attrNames[] = splitString(params
		@@ -1007,7 +981,8 @@ public Collection getAttributeCertificateAttributes(
1007	981	* @throws StoreException
1008	982	*/
1009	983	public Collection getCertificateRevocationLists(
1010	-	X509CRLStoreSelector selector) throws StoreException
984	+	X509CRLStoreSelector selector)
985	+	throws StoreException
1011	986	{
1012	987	String[] attrs =
		splitString(params.getCertificateRevocationListAttribute());
1013	988	String attrNames[] = splitString(params

Comments 0



Please [sign in](#) to comment.