**Leksioni 10**

**Database Connectivity with JDBC**

**Java Seciurity**

**10.1 Java Database Connectivity**

Lidhja e bazës së të dhënave me Java (JDBC) është një ndërfaqe e programit të aplikimit (API) e paketës Java SE që bën të mundur standardizimin dhe thjeshtimin e procesit të lidhjes së aplikacioneve Java me sistemet e jashtme, të menaxhimit të bazës së të dhënave (RDBMS). Në thelb, aplikacionet e shkruara në Java kryejnë logjikën. Gjuha Java ofron lehtësi për kryerjen e logjikës përsëritëse nëpërmjet cikleve, logjikë të kushtëzuar me thënie if dhe analiza të orientuara në objekte përmes përdorimit të klasave dhe ndërfaqeve. Por aplikacionet Java nuk i ruajnë të dhënat në mënyrë të vazhdueshme. Qëndrueshmëria e të dhënave zakonisht delegohet në bazat e të dhënave NoSQL të tilla si MongoDB dhe Cassandra, ose në bazat e të dhënave relacionale të tilla si DB2 e IBM ose SQL Server i Microsoft *popular open source database MySQL*.

**10.2 Ndërfaqet, klasat dhe komponentët e JDBC**

JDBC API është i përbërë nga një numër i ndërfaqeve dhe klasave që përfaqësojnë një lidhje me bazën e të dhënave, ofrojnë lehtësira për dërgimin e pyetjeve SQL në një bazë të dhënash dhe ndihmojnë zhvilluesin Java të përpunojë rezultatet e ndërveprimeve relacionale të bazës së të dhënave.

|  |  |
| --- | --- |
| Klasat dhe ndërfaet e zakonshme të JDBC API | |
| java.sql.Connection | Përfaqëson një lidhje nga programi Java në sistemin e bazës së të dhënave të jashtme |
| java.sql.Statement | Përdoret për të lëshuar deklarata të papërpunuara SQL në një bazë të dhënash relacionale |
| java.sql.PreparedStatement | Përdoret për lëshimin e deklaratave të parapërgatitura SQL në një bazë të dhënash relacionale |
| java.sql.ResultSet | Vendos Përfaqëson rezultatet e kthyera nga një bazë të dhënash relacionale pasi të jetë përpunuar një deklaratë SQL |
| java.sql.Blob | Blob përfaqëson një bazë të dhënash të regjistruar që përmban një objekt të madh binar siç është një imazh ose një video |
|  |  |
|  |  |

**10.2.1 Krahasimi mes JDBC dhe SQL**

Structured Query Language (SQL) është një specifikim ISO që përcakton se si aplikacionet mund të kërkojnë, azhornojnë dhe mënyra se si bashkëveprojnë me një bazë të dhënash relacionale. JDBC API nuk kryen ndonjë funksion i cili mund të kryhej përmes një query SQL. Qëllimi i JDBC API është të sigurojë një lidhje me një bazë të dhënash relacionale përmes query SQL që mund të kryhen, dhe rezultatet nga ato pyetje mund të përpunohen brenda një programi Java. JDBC është një API e lidhjes. SQL mbetet gjuha e përdorur për të komunikuar në të vërtetë me bazën e të dhënave.

**10.2.3 Arkitektura JDBC**

JDBC është krijuar për të bërë agnostik bazën e të dhënave të aplikacioneve Java. Ky është, një program i shkruar duke përdorur JDBC i cili do të funksionojë me çdo bazë të dhënash të përputhshme me JDBC. Ky ishte një aplikacion Java që testohej me Apache Derby i cili mund të vendoset me besim në një bazë të dhënash IBM DB2 në prodhim. Sidoqoftë, ka ndryshime midis shitësve të bazave të të dhënave, dhe këto ndryshime duhet të hiqen. Mjeti për heqjen e këtyre ndryshimeve njihet si një administrues JDBC.

Kur një përdorues Java dëshiron të lidhet me një bazë të dhënash, ajo kërkon një ndërfaqe JDBC të njohur si DriverManager, e cila ngarkon një drejtues që është shkruar posaçërisht nga shitësi i bazës së të dhënave me të cilën po lidhet programi Java. Ky drejtues përmban të gjithë informacionin e kërkuar për të lidhur programin Java me bazën e të dhënave. Drejtuesi JDBC është specifik për shitësin, kështu që drejtuesi MySQL JDBC është i ndryshëm nga drejtuesi Apache Derby JDBC. Detyra e këtyre drejtuesve të *mjeteve* është të adresojnë ndryshimet midis bazave të të dhënave në nivelin teknik, t'i heqin ato nga aplikacioni dhe t'u lejojnë zhvilluesve të Java të kenë besim se JDBC API me të cilin bashkëveprojnë dhe do të punojë me çdo bazë të dhënash në përputhje me rregullat dhe kërkesat e JDBC.

*Si të lidheni me një bazë të dhënash me JDBC?*

Hapat themelorë për tu lidhur me një bazë të të dhënave JDBC janë:

1. Ngarko drejtuesin e duhur të JDBC
2. Merrni URL-në e bazës së të dhënave
3. Përdorni DriverManger JDBC për t'u lidhur me bazën e të dhënave
4. Krijoni një objekt të bazuar në SQL Statement ose PreparedStatement
5. Ekzekutoni deklaratën kundër bazës së të dhënave
6. Përpunoni rezultatet dhe trajtoni ndonjë përjashtim të SQL
7. Mbyllni çdo lidhje ose burim të bazës së të dhënave që nuk përdoren më

**10.3 JDBC vs ODBC**

Lidhja JDBC dhe Baza e të Dhënave të Hapura (ODBC) janë shumë të ngjashme për nga natyra. Specifikimi ODBC përcakton një sërë kriteresh për lidhjen me bazat e të dhënave në përputhje me ODBC, ashtu si JDBC bën për bazat e të dhënave në përputhje me JDBC. Dallimi i madh midis të dyve është se JDBC është në dispozicion vetëm për gjuhët e shkruara dhe Java dhe aplikacionet e vendosura në JDK, ndërsa ODBC mund të përdoret me një larmi gjuhësh të ndryshme duke përfshirë Visual Basic, C #, FORTRAN dhe Pascal.

Java Development Kit (JDK) ofron diçka të quajtur një drejtues urë JDBC-ODBC që lejon zhvilluesit të shkruajnë aplikacione që përdorin JDBC API por lidhen me bazat e të dhënave në përputhje me ODBC siç është Microsoft Access. Kështu që është e mundur të hyni në një bazë të të dhënave ODBC duke përdorur JDBC.

10.3.1 Llojet e drejtuesve të JDBC

Ura JDBC-ODBC referohet si drejtuesi i JDBC i Tipit 1. Ekzistojnë katër lloje, me dallimet që janë:

1. Drejtuesi Type 4 JDBC është shkruar thjesht në Java dhe funksionon përmes një lidhje rrjeti. Kjo shpesh referohet si drejtues i mirë i JDBC.
2. Drejtues i tipit 3 JDBC i cili së pari ndërvepron me një server të softverit të mesëm siç është IBM WebSphere, i cili pastaj nga ana e tij komunikon me bazën e të dhënave relacionale. Kjo nganjëherë referohet si drejtuesi i përfaqësimit JDBC.
3. Drejtuesi i tipit 2 JDBC i cili mund të shkruhet ose jo në Java. Këto drejtues zakonisht përfshijnë kodin e pronarit të shkruar posaçërisht për një bazë të dhënash të caktuar për të optimizuar performancën e veprimeve. Ky drejtues shpesh quhet drejtues i trashë JDBC.
4. Drejtuesi i tipit 1 JDBC është ura e lartpërmendur JDBC-ODBC.

**10.4 Mospërputhja e rezistencës reale të objektit**

Filozofitë e zhvillimit të orientuar drejt objektit dhe sistemeve të bazës së të dhënave relacionale janë shumë të ndryshme, gjë që shpesh çon në diskutime në lidhje me object-relational dhe mospërputhje e pavaresisë. Sistemet e orientuara drejt objektit janë të dizajnuara shumë më ndryshe sesa sistemet relacionale, dhe tërheqja e të dhënave nga një bazë e të dhënave relacionale dhe sjellja e tyre në një sistem të orientuar drejt objektit nuk është një proces i drejtë përpara. Qëllimi i JDBC API është të përcaktojë klasa, metoda dhe ndërfaqe që minimizojnë mospërputhjen e rezistencës dhe e bëjnë më të lehtë hedhjen e të dhënave mbrapa-prapa midis një programi Java dhe një baze të dhënash të jashtme.

**10.4.1 JDBC vs Hibernate dhe JPA**

Për të ndihmuar në adresimin e mospërputhjes së impedancës relacioneve objek, ekzistojnë një numër veprimesh që thjeshtojnë detyrën e ngarkimit të të dhënave midis një baze të të dhënave relacionale dhe një programi Java. Popular object-relational mapping (ORM) përfshijnë Hibernate, TopLink dhe DataNucleus. Ndërsa secila kornizë ka grupin e vet të aftësive unike, të gjitha ato përputhen me standardin Java Persistence API, i cili tani është pjesë e specifikimeve Java EE / Jakarta EE.

**10.4.2 Bazat e të dhënave JDBC dhe NoSQL**

JDBC API është e përqendruar vetëm në ndërveprimet midis programeve Java dhe bazave të të dhënave relacionale. Ndërsa bazat e të dhënave NoSQL po rriten në popullaritet, nuk ka funksione të integruara në JDBC API për të lehtësuar ndërveprimet me bazat e të dhënave NoSQL siç janë Redis ose CouchDB. JDBC është e përqendruar vetëm në ndërveprimet me sistemet relacionale.

**10.5 Kritikat kundrejt JDBC**

JDBC shpesh kritikohet për rëndimin e zhvilluesit duke u kërkuar atyre që të shkruajnë një sasi të konsiderueshme të kodit për të kryer funksionimin bazë. Përveç kësaj, ndërveprimet me JDBC API hedhin një numër të madh të përjashtimeve të mundshme që duhet të trajtohen, pavarësisht nga fakti se shpesh ka shumë pak që një zhvillues mund të bëjë kur ndodhin këto gabime JDBC. Si rezultat, operacionet që mund të kërkojnë vetëm disa rreshta kodi në gjuhë të tilla si C # ose GoLang mund të jenë pesë ose gjashtë herë në gjatësi kur kryejnë të njëjtin veprim të bazës së të dhënave në Java.

### **10.6 Spring JDBC Framework**

Ka pasur shumë përpjekje nga anëtarët e komunitetit Java për të adresuar disa nga mangësitë e JDBC, me më të popullarizuarain Spring JDBC.

Kuadri Spring JDBC është ndërtuar plotësisht në fillim të JDBC API, kështu që mbështetja e saj bazohet plotësisht në standarde. Sidoqoftë, korniza ofron klasa të ndryshme ndihmëse të tilla si klasa Spring JdbcTemplate dhe ndërfaqja SQLExceptionTranslator JDBC, të cilat janë krijuar për të ndihmuar në trajtimin e përjashtimeve JDBC duke minimizuar gjithashtu sasinë e kodit të bojlarit që kërkohet për të kryer operacione të thjeshta të bazës së të dhënave.

Kuptimi i vërtetë i JDBC në një shënim të fundit, vlen të përmendet se në aspektin ligjor, JDBC në të vërtetë nuk qëndron për Lidhjen e të dhënave me Java. Termi JDBC është në vetvete një fjalë e markës tregtare dhe ndërsa vepron si një ombrellë nën të cilën ndodhen të gjitha bazat e të dhënave të ndryshme Java dhe API-të e lidhura me SQL, vetë aplikacioni i markës tregtare të Oracle nuk tregon askund se JDBC do të thotë lidhja e bazës së të dhënave Java. Në fakt, duke e bërë këtë në të vërtetë do të dobësonte të drejtën e Oracle për markën tregtare. Nëse një përdorues do të shkarkonte dhe lexonte specifikimet aktuale të JDBC, askund në atë JSR nuk do të gjendeshin fjalët lidhja e bazës së të dhënave Java.

**10.7 Siguria në Java**

Siguria në Java përfshin një grup të madh të API-ve, mjeteve dhe implementimeve të algoritmeve të sigurisë, mekanizmave dhe protokolleve që përdoren zakonisht. API-të e sigurisë Java shtrihen në një gamë të gjerë fushash, përfshirë kriptografinë, infrastrukturën e çelësit publik, komunikimin e sigurt, vërtetimin dhe kontrollin e qasjes. Teknologjia e sigurisë Java i siguron zhvilluesit një kornizë gjithëpërfshirëse të sigurisë për të shkruar aplikacione dhe gjithashtu i ofron përdoruesit ose administratorit një sërë mjetesh për të menaxhuar në mënyrë të sigurt aplikacionet.

JDK është krijuar me një theks të fortë në siguri. Në thelb të saj, gjuha Java në vetvete është e sigurt për llojin dhe siguron mbledhjen automatike të mbetjeve, duke rritur qëndrueshmërinë e kodit të aplikimit. Një mekanizëm i sigurt i ngarkimit dhe verifikimit të klasës siguron që të ekzekutohet vetëm kodi legjitim i Java-s. Arkitektura e sigurisë Java përfshin një grup të madh të ndërfaqeve programuese të aplikacioneve (API), mjeteve dhe implementimeve të algoritmeve, mekanizmave dhe protokolleve të sigurisë që përdoren zakonisht.

API-të e sigurisë Java shtrihen në një gamë të gjerë fushash. Ndërfaqet kriptografike dhe të infrastrukturës së çelësit publik (PKI) ofrojnë bazën themelore për zhvillimin e aplikacioneve të sigurta. Ndërfaqet për kryerjen e vërtetimit dhe kontrollit të qasjes mundësojnë që aplikacionet të ruhen nga hyrja e paautorizuar në burimet e mbrojtura.

API-të lejojnë zbatime të shumëfishta ndërvepruese të algoritmeve dhe shërbimeve të tjera të sigurisë. Shërbimet zbatohen nga ofruesit, të cilët janë të kyçur në JDK përmes një ndërfaqe standarde që e bën më të lehtë për aplikacionet marrjen e shërbimeve të sigurisë pa pasur nevojë të dinë asgjë për implementimet e tyre. Kjo i lejon zhvilluesit të përqendrohen në mënyrën e integrimit të sigurisë në aplikacionet e tyre, sesa në mënyrën se si të zbatojnë në të vërtetë mekanizmat komplekse të sigurisë.

JDK përfshin një numër ofruesish që implementojnë një grup themelor të shërbimeve të sigurisë. Gjithashtu lejon instalimin e ofruesve shtesë të personalizimit. Kjo u mundëson zhvilluesve të zgjerojnë platformën me mekanizma të rinj të sigurisë.

JDK është e ndarë në module. Modulet që përmbajnë API të sigurisë përfshijnë sa vijon:

|  |  |
| --- | --- |
| Modulet | Pershkrimi |
| [java.base](https://docs.oracle.com/javase/9/docs/api/java.base-summary.html) | Përcakton API-të themelore të Java SE. Paketat e përfshira përfshijnë java.security, *javax.crypto*, *javax.net.ssl* ​​dhe *javax.security.auth*. |
| [java.security.jgss](https://docs.oracle.com/javase/9/docs/api/java.security.jgss-summary.html) | Përcakton lidhjen Java të API-ve të Shërbimeve të Përgjithshme të Sigurisë IETF (GSS-API). Ky modul gjithashtu përmban mekanizma GSS-API përfshirë Kerberos v5 dhe SPNEGO. |
| [java.security.sasl](https://docs.oracle.com/javase/9/docs/api/java.security.sasl-summary.html) | Përcakton mbështetjen Java për shtresën e thjeshtë të vërtetimit dhe sigurisë të IETF (SASL). Ky modul gjithashtu përmban mekanizma SASL përfshirë DIGEST-MD5, CRAM-MD5 dhe NTLM. |
| [java.smartcardio](https://docs.oracle.com/javase/9/docs/api/java.smartcardio-summary.html) | Përcakton API-në e Kartës Smart Java. |
| [java.xml.crypto](https://docs.oracle.com/javase/9/docs/api/java.xml.crypto-summary.html) | Përcakton API për kriptografinë XML. |
| [jdk.security.auth](https://docs.oracle.com/javase/9/docs/api/jdk.security.auth-summary.html) | Siguron implementime të ndërfaqeve javax.security.auth. \* dhe moduleve të ndryshme të vërtetimit. |
| [jdk.security.jgss](https://docs.oracle.com/javase/9/docs/api/jdk.security.jgss-summary.html) | Përcakton shtesat Java në GSS-API dhe një implementim të mekanizmit SASL GSS-API. |

**10.8 Java Language Security dhe Verifikimi Bytecode**

Gjuha Java është krijuar për të qenë e sigurt në llojin dhe e lehtë për t'u përdorur. Ajo siguron menaxhim automatik të kujtesës, mbledhjen e mbetjeve dhe kontrollimin e vargut në vargje. Kjo zvogëlon barrën e përgjithshme të programimit të vendosur mbi zhvilluesit, duke çuar në më pak gabime delikate të programimit dhe në një kod më të sigurt dhe më të fortë.

Një përpilues përkthen programet Java në një përfaqësim bytecode të pavarur nga makina. Një verifikues bytecode thirret për të siguruar që vetëm bytecodes legjitime ekzekutohen në kohën e ekzekutimit të Java. Kontrollon që kodet e bajtëve janë në përputhje me specifikimin e gjuhës Java dhe nuk shkelin rregullat e gjuhës Java ose kufizimet e hapësirës së emrave. Verifikuesi gjithashtu kontrollon për shkelje të menaxhimit të kujtesës, ngarkesën e të dhënave ose mbingarkesën dhe llojet e paligjshme të të dhënave. Pasi të jenë verifikuar bytecodet, koha e ekzekutimit në Java i përgatit ato për ekzekutim.

Përveç kësaj, gjuha Java përcakton modifikues të ndryshëm të hyrjes që mund t'u caktohen klasave, metodave dhe fushave Java, duke u mundësuar zhvilluesve të kufizojnë hyrjen në implementimet e klasave të tyre, siç është e përshtatshme. Gjuha përcakton katër nivele të dallueshme të hyrjes:

*private:* Modifikuesi më kufizues; hyrja nuk lejohet jashtë klasës së veçantë në të cilën përcaktohet anëtari privat (një metodë, për shembull).

*protection:* Lejon hyrjen në çdo nënklasë ose në klasa të tjera brenda së njëjtës paketë.

*Paketa private:* Nëse nuk është specifikuar, atëherë ky është niveli i paracaktuar i hyrjes; lejon hyrjen në klasat brenda së njëjtës paketë.

*publik*: Nuk garanton më që elementi është i arritshëm kudo; aksesueshmëria varet nga fakti nëse pakoja që përmban atë element eksportohet nga moduli i saj përcaktues dhe nëse ai modul është i lexueshëm nga moduli që përmban kodin që po përpiqet të aksesojë atë.

**10.9 Arkitektura Themelore e Sigurisë**

JDK përcakton një sërë API-ve që përfshijnë fusha kryesore të sigurisë, duke përfshirë kriptografinë, infrastrukturën e çelësit publik, vërtetimin, komunikimin e sigurt dhe kontrollin e qasjes. API lejojnë zhvilluesit të integrojnë lehtësisht sigurinë në kodin e tyre të aplikimit.

***API-të janë hartuar rreth parimeve të mëposhtme***:

1. Pavarësia e zbatimit

Aplikimet nuk kanë nevojë të zbatojnë vetë sigurinë. Përkundrazi, ata mund të kërkojnë shërbime të sigurisë nga JDK. Shërbimet e sigurisë zbatohen nga ofruesit të cilët janë të *kyçur* në JDK përmes një ndërfaqe standarde. Një aplikacion mund të mbështetet në shumë ofrues të pavarur për funksionalitetin e sigurisë.

1. Ndërveprimi i zbatimit

Ofruesit janë të ndërveprueshëm në të gjithë aplikacionet. Në mënyrë të veçantë, një aplikacion nuk është i lidhur me një ofrues specifik nëse nuk mbështetet në vlerat e paracaktuara nga ofruesi. Zgjatësia e algoritmit JDK përfshin një numër ofruesish të integruar që implementojnë një grup themelor të shërbimeve të sigurisë që përdoren gjerësisht sot. Sidoqoftë, disa aplikacione mund të mbështeten në standardet e reja që nuk janë zbatuar ende, ose në shërbimet e pronarit. JDK mbështet instalimin e ofruesve të personalizuar që implementojnë shërbime të tilla.

1. Ofruesit e Sigurisë

Klasa *java.security.Provider* përmbledh nocionin e një ofruesi të sigurisë në platformën Java. Ajo specifikon emrin e ofruesit dhe rendit shërbimet e sigurisë që zbaton. Ofrues të shumtë mund të konfigurohen në të njëjtën kohë dhe renditen sipas radhës së preferencës. Kur kërkohet një shërbim sigurie, zgjidhet ofruesi i përparësisë më të lartë që zbaton atë shërbim. Aplikimet mbështeten në metodën përkatëse *getInstance*() për të kërkuar një shërbim sigurie nga një ofrues themelor. Për shembull, krijimi i përmbledhjes së mesazheve përfaqëson një lloj shërbimi të disponueshëm nga ofruesit. Për të kërkuar zbatimin e një algoritmi specifik të tretjes së mesazheve, thërrisni metodën *java.security.MessageDigest.getInstance*

**10.9.1 Kriptografia**

Arkitektura e kriptografisë Java është një kornizë për të hyrë dhe zhvilluar funksionalitetin kriptografik për platformën Java.

Ai përfshin API-të për një larmi të madhe të shërbimeve kriptografike, duke përfshirë sa vijon:

1. Algoritmet e tretjes së mesazheve
2. Algoritmet e nënshkrimit dixhital
3. Kriptimi simetrik i pjesës më të madhe dhe transmetimit
4. Kriptimi asimetrik
5. Kriptimi i bazuar në fjalëkalim (PBE)
6. Kriptografia e kurbës eliptike (ECC)
7. Algoritmet kryesore të marrëveshjes
8. Gjeneratorët kryesorë
9. Kodet e vërtetimit të mesazheve (MAC)
10. Gjeneratorë të Sigurt të Numrave të Rastit

Për arsye historike (kontrolli i eksportit), API-të e kriptografisë janë të organizuara në dy paketa të dallueshme:

Paketat *java.security* dhe *java.security. \** këto paketa përmbajnë klasa që nuk i nënshtrohen kontrolleve të eksportit (si Nënshkrimi dhe MessageDigest)

Paketa *javax.crypto* përmban klasa që i nënshtrohen kontrolleve të eksportit (si Cipher dhe KeyAg marrëveshje)

Ndërfaqet kriptografike janë të bazuara në ofrues, duke lejuar zbatime të shumëfishta dhe ndërvepruese të kriptografisë. Disa ofrues mund të kryejnë operacione kriptografike në softuer; të tjerët mund të kryejnë operacione në një harduer (për shembull, në një pajisje me kartë inteligjente ose në një përshpejtues kriptografik të harduerit). Ofruesit që zbatojnë shërbime të kontrolluara nga eksporti duhet të nënshkruhen dixhitalisht nga një certifikatë e lëshuar nga Autoriteti i Certifikimit Oracle JCE.

Platforma Java përfshin ofrues të integruar për shumë prej algoritmeve kriptografikë më të përdorur, duke përfshirë algoritmet e nënshkrimit RSA, DSA dhe ECDSA, algoritmin e kriptimit AES, algoritmet e tretjes së mesazheve SHA-2 dhe Diffie-Hellman (DH) dhe algoritmet e marrëveshjes kryesore të Kurbës Eliptike Diffie-Hellman (ECDH). Shumica e ofruesve të integruar implementojnë algoritme kriptografike në kodin Java.

Platforma Java gjithashtu përfshin një ofrues të integruar që vepron si një urë për një shenjë vendase PKCS # 11 (v2.x). Ky ofrues, i quajtur SunPKCS11, lejon aplikacionet Java të hyjnë në mënyrë të përsosur në shërbimet kriptografike të vendosura në argumentet e përputhshme me PKCS # 11.

Në Windows, platforma Java përfshin një ofrues të integruar që vepron si një urë për Microsoft CryptoAPI vendas. Ky ofrues, i quajtur SunMSCAPI, lejon që aplikacionet Java të qasen në mënyrë të përsosur në shërbimet kriptografike në Windows përmes CryptoAPI.

**10.9.2 Infrastruktura e Çelësit Publik**

Infrastruktura e Çelësit Publik (PKI) është një term i përdorur për një kornizë që mundëson shkëmbimin e sigurt të informacionit bazuar në kriptografinë e çelësit publik. Ai lejon që identitetet (e njerëzve, organizatave, etj.) Të jenë të lidhura me certifikatat dixhitale dhe siguron një mjet për të verifikuar vërtetësinë e certifikatave. PKI përfshin çelësat, çertifikatat, kriptimin e çelësit publik dhe autoritetet e besuara të çertifikimit (AK) që gjenerojnë dhe nënshkruajnë dixhital çertifikatat.

Platforma Java përfshin API-të dhe mbështetjen e ofruesit të certifikatave dixhitale X.509 dhe Listave të Revokimit të Certifikatave (CRLs), si dhe ndërtimin dhe vlerësimin e rrugës së certifikimit në përputhje me PKIX. Klasat që lidhen me PKI gjenden në paketat java.security dhe java.security.cert.

**10.9.3 Ruajtja e çelësit dhe certifikatës**

Platforma Java siguron ruajtje afatgjatë të vazhdueshme të çelësave dhe çertifikatave kriptografike përmes dyqaneve të çelësave dhe çertifikatave. Në mënyrë të veçantë, klasa *java.security.KeyStore* përfaqëson një dyqan çelësash, një depo të sigurt të çelësave kriptografikë dhe / ose çertifikatave të besuara (që do të përdoren, për shembull, gjatë vlerësimit të rrugës së çertifikimit), dhe klasa *java.security.cert.CertStore* përfaqëson një dyqan certifikatash, një depo publike dhe potencialisht e gjerë e certifikatave të palidhura dhe tipikisht të pabesueshme. Një CertStore gjithashtu mund të ruajë CRL.

Implementimet KeyStore dhe CertStore dallohen sipas llojeve. Platforma Java përfshin llojet standarde të dyqaneve kryesore PKCS11 dhe PKCS12 (implementimet e të cilave janë në përputhje me specifikimet përkatëse PKCS nga RSA Security). Ai gjithashtu përmban një tip të dyqanit të çelësave të bazuar në skedarë të quajtur JKS (që qëndron për Java Key Store), dhe një lloj të quajtur DKS (Domain Key Store) i cili është një koleksion i dyqaneve të tasteve që paraqiten si një dyqan i vetëm logjik. Platforma Java përfshin një dyqan të veçantë të integruar të çelësave, cacerts, që përmban një numër certifikatash për AK të njohura, të besuara. Shërbimi keytool është përgjegjës për rendisinë e certifikatave të përfshira në cacerts. Kjo do të thotë që çelësat dhe çertifikatat që qëndrojnë në pajisje të sigurta (të tilla si një kartë inteligjente) mund të arrihen dhe të përdoren nga aplikacionet Java përmes KeyStore API. Vini re se çelësat e kartave inteligjente mund të mos lejohen të largohen nga pajisja. Ky objekt i kthyer nga KeyStore API mund të jetë thjesht një referencë në çelësin (domethënë, nuk do të përmbante materialin aktual të çelësit). Një objekt i tillë Kyç mund të përdoret vetëm për të kryer operacione kriptografike në pajisjen ku ndodhet çelësi aktual. Platforma Java gjithashtu përfshin një lloj dyqani të certifikatës LDAP (për të hyrë në çertifikatat e ruajtura në një direktori LDAP), si dhe një lloj dyqani të çertifikatës së Koleksionit në memorje (për të hyrë në certifikatat e menaxhuara në një objekt java.util.Collection).

Mjetet e Infrastrukturës së Çelësit Publik

Ekzistojnë dy mjete të integruara për të punuar me çelësat, çertifikatat dhe dyqanet e çelësave: keytool krijon dhe administron dyqanet kryesore. Përdoreni atë për të kryer detyrat e mëposhtme: Krijoni çifte çelësash publik / privat, shfaq importin dhe eksportoni certifikatat X.509 v1, v2 dhe v3 të ruajtura si skedarë ,krijoni certifikata X.509 Çertifikata e lëshimit (PKCS # 10) kërkon që të dërgohen në AK, krijoni certifikata bazuar në kërkesat e certifikatave ,i përgjigjet e certifikatës së importit (të marra nga kërkesat e certifikatave të dërguara nga AK) , caktoni çertifikatat e çelësit publik si të besuara, pranoni një fjalëkalim dhe ruajeni në mënyrë të sigurt si një çelës sekret jarsigner nënshkruan skedarët JAR dhe verifikon nënshkrimet në skedarët e nënshkruar JAR. Formati i skedarit Java ARchive (JAR) mundëson bashkimin e shumë skedarëve në një skedar të vetëm. Në mënyrë tipike, një skedar JAR përmban skedarët e klasës dhe burimet ndihmëse të shoqëruara me applet dhe aplikacione. Për të nënshkruar kodin dixhital, kryeni sa vijon:   
1. Përdorni keytool për të gjeneruar ose importuar çelësa dhe çertifikata të përshtatshme në dyqanin tuaj të çelësave (nëse nuk janë atje tashmë).

2. Përdorni mjetin jar për të paketuar kodin në një skedar JAR.

3. Përdorni mjetin jarsigner për të nënshkruar skedarin JAR. Mjeti jarsigner hyn në një dyqan çelësash për të gjetur çelësat dhe çertifikatat e nevojshme për të nënshkruar një skedar JAR ose për të verifikuar nënshkrimin e një skedari të nënshkruar JAR.

***Kontrolli i Hyrjes***

Arkitektura e kontrollit të qasjes në platformën Java mbron hyrjen në burimet e ndjeshme (për shembull, skedarët lokalë) ose kodin e ndjeshëm të aplikimit (për shembull, metodat në një klasë). Të gjitha vendimet e kontrollit të qasjes ndërmjetësohen nga një menaxher sigurie, i përfaqësuar nga klasa java.lang.SecurityManager. Një SecurityManager duhet të instalohet në kohën e ekzekutimit të Java-së në mënyrë që të aktivizojë kontrollet e kontrollit të hyrjes.

Apletet Java dhe aplikacionet Java Web Start ekzekutohen automatikisht me një SecurityManager të instaluar. Sidoqoftë, aplikacionet lokale të ekzekutuara përmes komandës java nuk përdoren si parazgjedhje me një SecurityManager të instaluar. Në mënyrë që të ekzekutohen aplikacione lokale me një SecurityManager, ose vetë aplikacioni duhet të vendosë një program përmes metodës setSecurityManager (në klasën java.lang.System), ose java duhet të thirret me një argument -Djava.security.manager me komandë në linjë.

***Lejet***

Një leje përfaqëson hyrjen në një burim të sistemit. Në mënyrë që një hyrje në burime të lejohet për një applet (ose një aplikacion që ekzekutohet me një menaxher sigurie), leja përkatëse duhet t'i jepet shprehimisht kodit që përpiqet të ketë qasje.

Kur kodi Java ngarkohet nga një ngarkues klase në kohën e ekzekutimit të Java, ngarkuesi i klasës automatikisht shoqëron informacionin e mëposhtëm me atë kod:

*Nga ishte ngarkuar kodi*

*Kush e nënshkroi kodin (nëse dikush)*

*Lejet e parazgjedhura të dhëna kodit*

Ky informacion shoqërohet me kodin, pavarësisht nëse kodi është shkarkuar përmes një rrjeti jo të besueshëm (p.sh., një applet) ose është ngarkuar nga sistemi i skedarëve (p.sh., një aplikacion lokal). Vendndodhja nga e cila u ngarkua kodi përfaqësohet nga një URL, nënshkruesi i kodit përfaqësohet nga zinxhiri i certifikatave të nënshkruesit dhe lejet e parazgjedhura përfaqësohen nga java.security.Objekte të lejes.

Lejet e parazgjedhura, të dhëna automatikisht për kodin e shkarkuar, përfshijnë mundësinë për të rikthyer lidhjet e rrjetit në hostin nga ka ardhur. Lejet e parazgjedhura që i jepen automatikisht kodit të ngarkuar nga sistemi lokal i skedarëve përfshijnë aftësinë për të lexuar skedarë nga drejtoria nga erdhi, dhe gjithashtu nga nëndirektoritë e këtij direktori.

Vini re se identiteti i përdoruesit që ekzekuton kodin nuk është i disponueshëm në kohën e ngarkimit të klasës. Është përgjegjësi e kodit të aplikimit të vërtetojë përdoruesin përfundimtar nëse është e nevojshme (shih seksionin Vërtetimi). Pasi përdoruesi të jetë vërtetuar, aplikacioni mund ta shoqërojë atë përdorues dinamikisht me kodin ekzekutues duke thirrur metodën *doAs* në klasën *javax.security.auth.Subject*.

**10.10 Politika e Sigurisë**

Një grup i kufizuar i lejeve parazgjedhur i jepen kodit nga ngarkuesit e klasave. Administratorët kanë aftësinë për të menaxhuar në mënyrë fleksibile lejet shtesë të kodit përmes një politike të sigurisë. Java SE përmbledh nocionin e një politike të sigurisë në klasën java.security.Policy. Ekziston vetëm një objekt i Politikës i instaluar në kohën e duhur të Java-s në çdo kohë. Përgjegjësia themelore e objektit Policy është të përcaktojë nëse lejohet kodimi i hyrjes në një burim të mbrojtur (karakterizuar nga vendi ku është ngarkuar, kush e ka nënshkruar atë dhe kush po e ekzekuton atë). Se si një objekt i politikës e bën këtë përcaktim varet nga zbatimi. Për shembull, mund të konsultohet me një bazë të dhënash që përmban të dhëna autorizimi, ose mund të kontaktojë një shërbim tjetër.

Java SE përfshin një zbatim të paracaktuar të Politikës që lexon të dhënat e tij të autorizimit nga një ose më shumë skedarë ASCII (UTF-8) të konfiguruar në skedarin e vetive të sigurisë. Këto skedarë politikash përmbajnë grupe të sakta të lejeve të dhëna të kodit: posaçërisht, grupet e sakta së lejeve të dhëna të kodit të ngarkuar nga vende të veçanta, të nënshkruar nga njësi të veçanta dhe që ekzekutohen si përdorues të veçantë. Regjistrimet e politikave në secilën skedar duhet të jenë në përputhje me një sintaksë të dokumentuar të pronarit dhe mund të kompozohen përmes një redaktuesi të thjeshtë teksti ose programit politik të mjeteve grafike.

**10.10.1 Zbatimi i Kontrollit të Hyrjes**

Koha e ekzekutimit Java mban gjurmët e sekuencës së thirrjeve Java që bëhen kur ekzekuton një program. Kur kërkohet qasja në një burim të mbrojtur, i gjithë pirja e thirrjeve, si parazgjedhje, vlerësohet për të përcaktuar nëse lejohet hyrja e kërkuar.

Siç u përmend më parë, burimet mbrohen nga SecurityManager. Kodi i ndjeshëm ndaj sigurisë në JDK dhe në aplikacione mbron qasjen në burime përmes kodit si më poshtë:

SecurityManager sm = System.getSecurityManager ();

if (sm! = null) {

sm.chekPremission (perm);

}

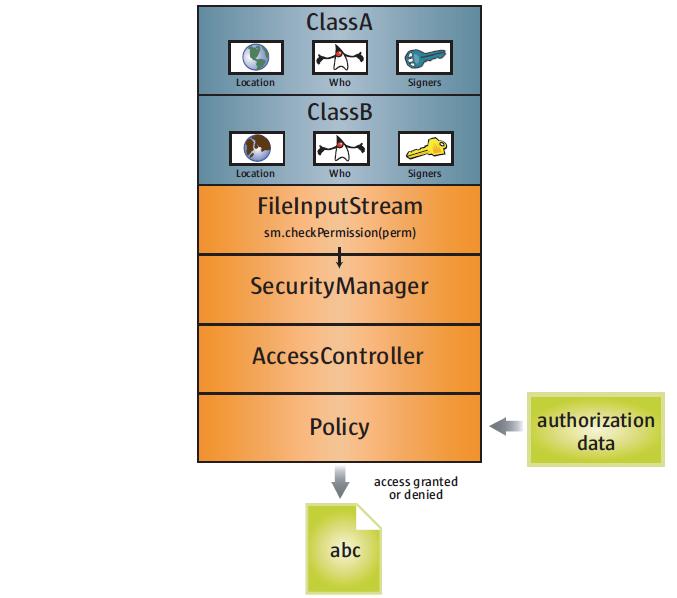
Perm e objektit të Lejes korrespondon me hyrjen e kërkuar. Për shembull, nëse bëhet një përpjekje për të lexuar skedarin / tmp / abc, leja mund të ndërtohet si më poshtë:

*Permission Perm = new java.io.FilePermission ("/ tmp / abc", "read");*

Zbatimi i paracaktuar i SecurityManager ia delegon vendimin e tij zbatimit të *java.security.AccessController.* **AccessController** përshkon stokun e thirrjeve, duke kaluar te politika e instaluar e sigurisë çdo element kodi në stok, së bashku me lejen e kërkuar (për shembull, FilePermission në shembullin e mëparshëm). Politika përcakton nëse jepet hyrja e kërkuar, bazuar në lejet e konfiguruara nga administratori. Nëse hyrja nuk jepet, AccessController hedh një *java.lang.SecurityException*.

Figura 1-1 ilustron zbatimin e kontrollit të qasjes. Në këtë shembull të veçantë, fillimisht ka dy elementë në stekun e thirrjeve, ClassA dhe ClassB. ClassA thërret një metodë në ClassB, e cila më pas përpiqet të ketë qasje në skedar / tmp / abc duke krijuar një shembull të *java.io.FileInputStream*. Konstruktori *FileInputStream* krijon një *FilePermission*, perm, siç tregohet më sipër, dhe më pas kalon perm në metodën checkPermission të klasës SecurityManager. Në këtë rast të veçantë, duhet të kontrollohen vetëm lejet për ClassA dhe ClassB, sepse të gjitha klasat në modulin java.base, përfshirë FileInputStream, SecurityManager dhe AccessController, automatikisht i merr të gjitha lejet. Në këtë shembull, ClassA dhe ClassB kanë karakteristika të ndryshme të kodit - ato vijnë nga vende të ndryshme dhe kanë nënshkrues të ndryshëm. Secili mund të jetë dhënë një grup tjetër të lejeve. AccessController lejon qasje në skedarin e kërkuar vetëm nëse Politika tregon që të dy klasave u është dhënë Leja e kërkuar e Skedarit.

***Figure 1- 1 Access to Resources***



**10.11 Nënshkrimi XML Java XML**

Nënshkrimi Dixhital API është një standard Java API për gjenerimin dhe vërtetimin e Nënshkrimeve XML. Nënshkrimet XML mund të zbatohen për të dhëna të çdo lloji, XML ose binare (shih Sintaksën dhe Përpunimin e Nënshkrimit XML). Nënshkrimi që rezulton përfaqësohet në XML. Një Nënshkrim XML mund të përdoret për të siguruar të dhënat tuaja dhe për të siguruar integritetin e të dhënave, vërtetimin e mesazheve dhe vërtetimin e nënshkruesve. API është krijuar për të mbështetur të gjitha tiparet e kërkuara ose të rekomanduara të Rekomandimit W3C për Sintaksën dhe Përpunimin e Nënshkrimit XML. API është e zgjerueshme dhe e ndërlidhshme dhe bazohet në Arkitekturën e Java Krijuesit të Shërbimit të Kriptografisë. Java XML Nënshkrimi Dixhital API, i cili është në modulin java.xml.crypto, përbëhet nga gjashtë paketa:

* javax.xml.crypto
* javax.xml.crypto.dsig
* javax.xml.crypto.dsig.keyinfo
* javax.xml.crypto.dsig.spec
* javax.xml.crypto.dom
* javax.xml.crypto.dsig.dom