MANUAL EXIST

JAVA + EXIST + ECLIPSE

Barcelona, 4 de Març del 2009

Alpha Release 0.4

(26 de Febrer del 2010)

Contingut

Introducció 1

Idea 1

Objectiu 1

Què és eXist? 1

Abans de començar... 2

Eclipse 2

eXist 2

APIs 2

Instal·lació d'eXist 3

Instal·lació 3

Configuració d'Eclipse 6

Creació del Projecte 6

Instal·lació del Plug-in d'eXist per Eclipse 7

Configuració del Plug-in d'eXist per a Eclipse 10

La primera col·lecció 12

Creant una col·lecció 12

Creant un fitxer dins d'una col·lecció 13

Poblant la base de dades amb una mica d'informació 14

La primera XQuery 14

La interfície de comunicació Java 16

Creant la classe 16

Especificant la ruta de les APIs necessàries 19

Fent la primera crida des de Java 20

Per a aprofundir... 21

Insert, Update i Delete 21

Una mica més enllà 23

Introducció

Idea

Probablement si esteu llegint aquest manual és que ara mateix esteu cursant l'assignatura de PXC (Projecte de Xarxes de Computadors) a la FIB (Facultat d'Informàtica de Barcelona). Si no és així també sou benvinguts, però tot el que conté aquest document està una mica enfocat de cara a l'assignatura en qüestió. A l'assignatura de PXC se us demana dissenyar i implementar un sistema d'informació o aplicació que desenvolupi algun procés de comunicació per xarxa, i normalment qualsevol sistema d'informació acaba requerint un suport i una lògica per a emmagatzemar la informació. Depenent de a quin grup aneu probablement us hauran recomanat eXist coma administrador de bases de dades tot i que molts de vosaltres segurament estigueu molt més acostumats a fer-ne servir d'altres com per exemple MySQL.
La idea d'escriure aquest manual sorgeix després d'un quadrimestre fent servir eXist com a administrador de bases de dades, i després de veure la poca informació que hi ha a la xarxa relacionada amb el tema. Volia que totes les hores invertides en la recerca d'informació, d'exemples i en autoformació en un llenguatge nou (XQuery, XUpdate) de comunicació amb la Base de Dades, servissin per a alguna cosa més que simplement per a aprovar una assignatura, i crec que és una bona idea que deixi aquest document per a la posteritat.

Objectiu

L'objectiu d'aquest manual és intentar guiar a qualsevol persona des de la instal·lació d'eXist al servidor fins a les crides desde Java per fer operacions sobre els fitxers emmagatzemats en ell. Seguint aquest manual pas a pas, acabareu tenint, el que anomenaríem seguint la idea de l'arquitectura en tres capes, un esquelet de la capa de dades, preparat per ser cridada des de domini tant per obtenir com per guardar o actualitzar dades emmagatzemades.

Què és eXist?

eXist no és més que un administrador de bases de dades. El fet que el diferencia dels dba habituals, és que no segueix la idea de base de dades relacional, sinó que segueix un model jeràrquic, el model de l'XML.
Els fitxers on emmagatzemem la informació amb eXist són tot fitxers XML, i per tal d'establir relacions ho hem de fer explícitament d'una manera que, d'altra banda, és bastant senzilla. La comprovació de la integritat de les dades potser és la part més complicada d'un sistema sense relacions.

El fet que les dades estiguin guardades en format XML, és molt útil de cara a manegar les dades obtingudes, ja que una crida a la capa de dades retorna una String XML, que mitjançant les transformacions XSLT podem tranformar molt fàcilment en HTML o fins i tot PDF.

Abans de començar...

Eclipse

El llenguatge de programació que faré servir durant tot el manual és Java, i per tal de poder fer servir certes avantatges d'un IDE, utilitzaré eclipse per als exemples.
Us recomano per tant que us el baixeu d'aquesta adreça http://www.eclipse.org/downloads/ i l'instal·leu al vostre ordinador.

eXist

Necessitareu també l'instal·lador d'eXist. En aquest exemple l'instal·laré a Windows en un ordinador a part on només hi faré córrer eXist. http://exist.sourceforge.net/download.html. Baixeu-vos la versió executable de Windows si voleu seguir aquest exemple al peu de la lletra.

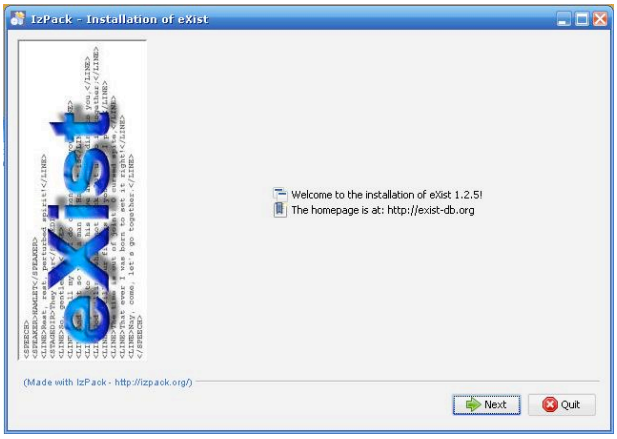
APIs

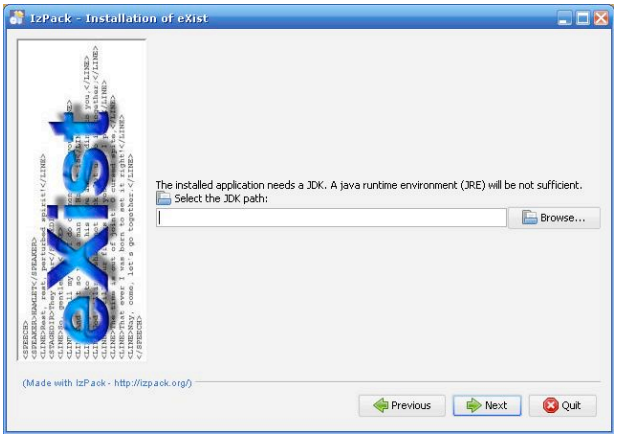
Per tal d'establir la comunicació entre Java i eXist, i també per altres motius es necessiten unes Apis en concret: exist.jar, jdom.jar, log4j-1.2.15.jar, xmldb.jar, xmlrpc-1.2.jar.
Us les podeu descarregar de http://www.llima.net/mutsuda/APIS.zip. Si aquest manual és tant antic que el fitxer no existeix, busqueu les APIS a google o al buscador que estigui de moda actualment i procureu que siguin les mateixes versions.

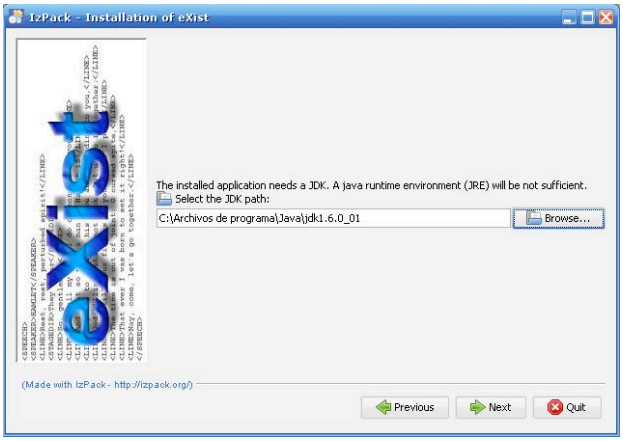
Instal·lació d'eXist

Instal·lació

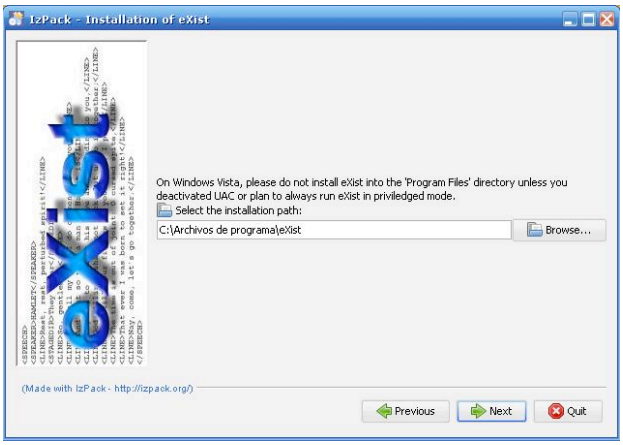
Un cop hagueu descarregat l'instal·lador, executeu-lo i seguiu el procés d'instal·lació. Busqueu on teniu instal·lat el JDK. Generalment el trobareu a **"Archivos de Programa/Java/"**.



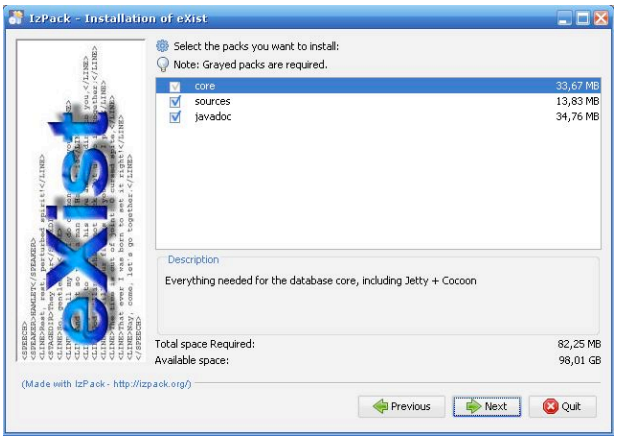




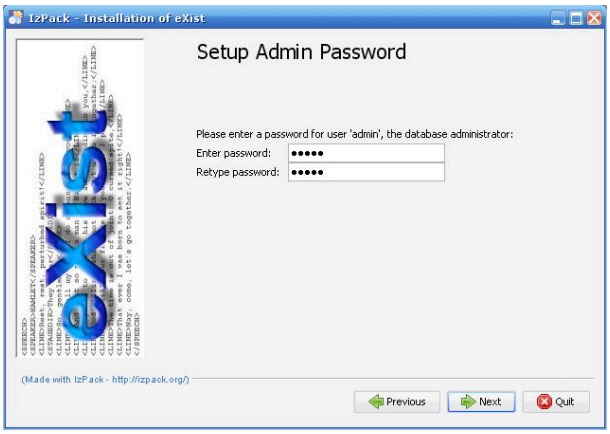
Seleccioneu la ubicació on el voleu instal·lar.



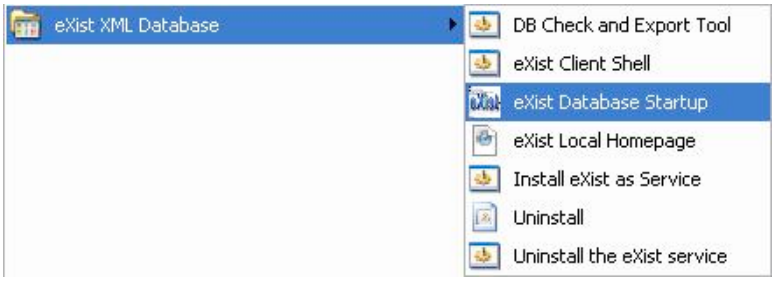
A la següent finestra veureu que us demana si voleu instal·lar també el codi i el javadoc a part del nucli, si aneu sobrats d'espai no està de més, així si teniu qualsevol dubte podreu consultar-lo.



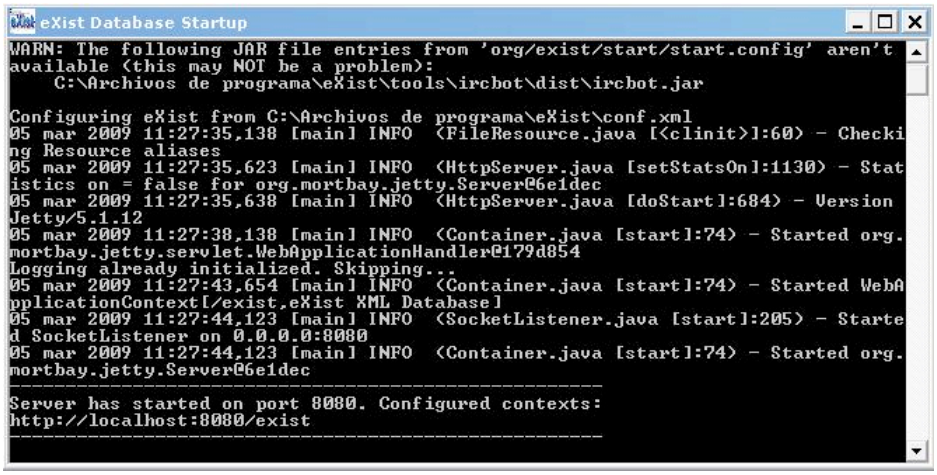
Creeu els accessos directes que necessiteu i escolliu un usuari i contrasenya per tal de restringir l'accés a les dades. Després no l'oblideu.



Un cop finalitzada la instal·lació, engegueu eXist mitjançant el menú d'inici allà on el tingueu instal·lat. Seleccioneu el que diu **"eXist Database Startup"**.



Se us obrirà una finestra de línia de comandes i començarà a escriure coses. Finalment, quan deixi d'escriure, us apareixerà el port pel qual us comunicareu amb eXist, tant si el feu servir des del mateix ordinador, com si us hi connecteu des de la Xina. Aquest port, per defecte, és el 8080.
Penseu que si voleu accedir a eXist des de fora de casa, haureu d'entrar a la configuració del router, i fer que les peticions al port 8080 es dirigeixin cap a la IP interna de l'ordinador on teniu eXist instal·lat. Això és fa amb les opcions de NAT. Si només el voleu fer servir a la xarxa local, vigileu amb els Firewalls que no us bloquegin el port 8080.

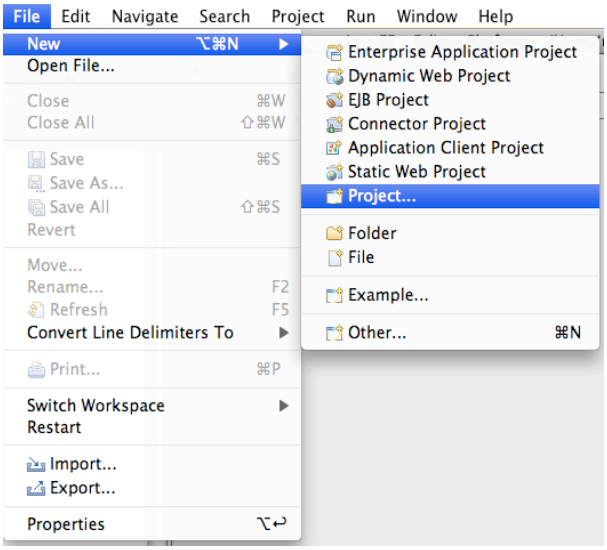


Un cop fet això, anem a comprovar que ens podem comunicar amb eXist, de moment encara des de fora de Java, però fent servir un Plug-in d'eXist per a Eclipse.

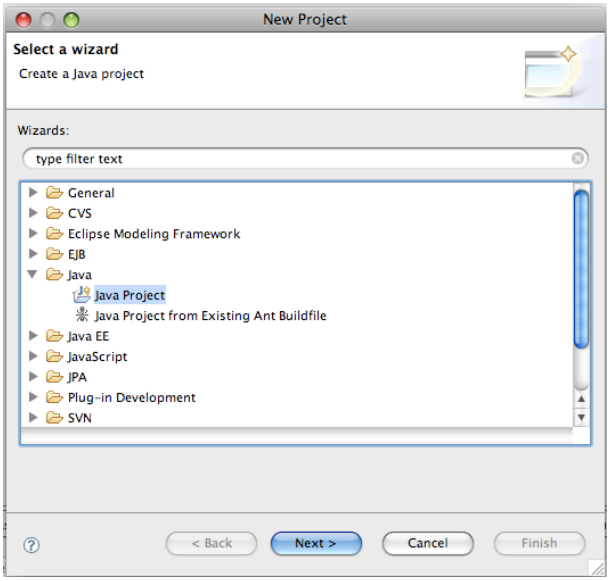
Configuració d'Eclipse

Creació del Projecte

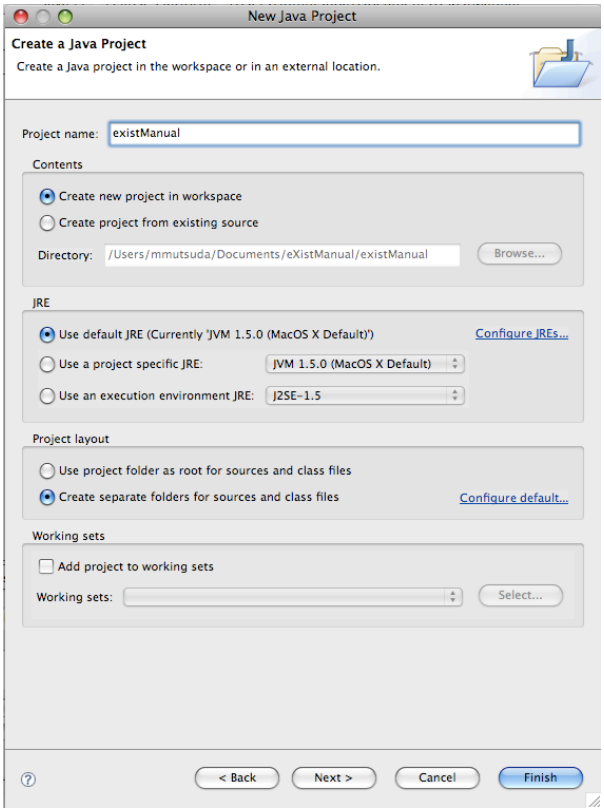
Per tal de crear la interfície de comunicació de Java amb eXist, necessitem crear un nou projecte.



El projecte que crearem serà de tipus "Java Project".



En aquest exemple li he dit **"existManual"**, i he deixat la resta d'opcions per defecte.

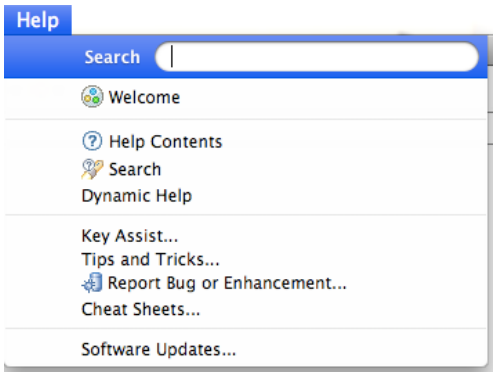


Finalitzem, i ens apareixerà el nou projecte creat.

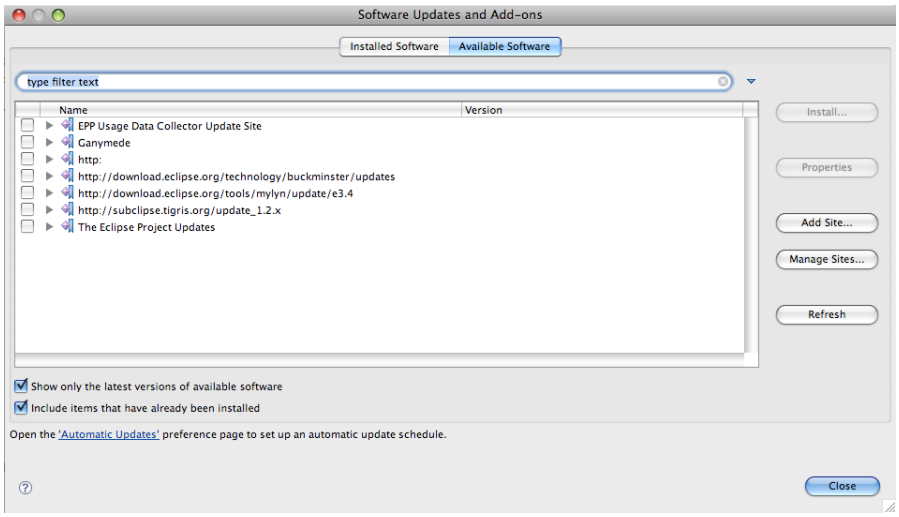
Instal·lació del Plug-in d'eXist per Eclipse

Una eina que em va ser molt útil durant el desenvolupament del projecte, va ser un plug-in d'eXist per a eclipse, que permet fer crides a la base de dades, creacions de fitxers, modificacions, etc, d'una manera una mica intuïtiva. És molt útil de cara a provar els **"Selects"** o **"Inserts"** més complicats, de manera externa a Java.

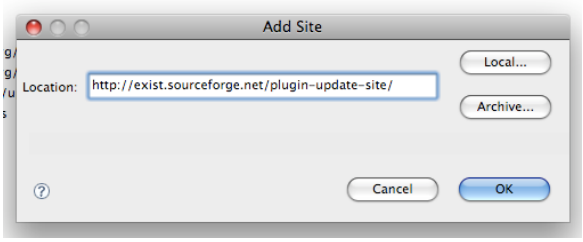
Per tal d'instal·lar el Plug-in, anirem al menú **"Help"** i seleccionarem **"Software Updates"**.



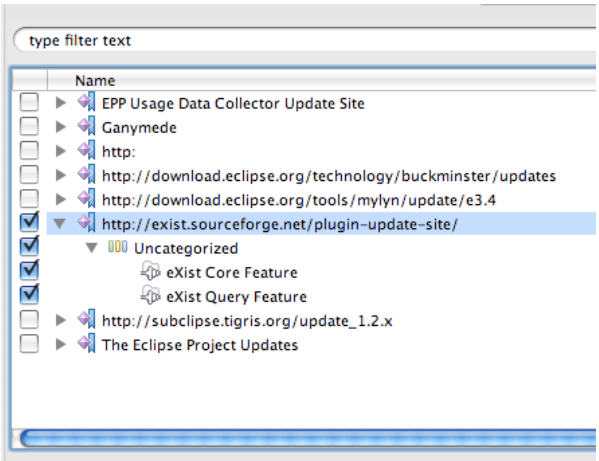
Seleccionem la pestanya **"Available Software"** i fem click a **"Add Site..."**



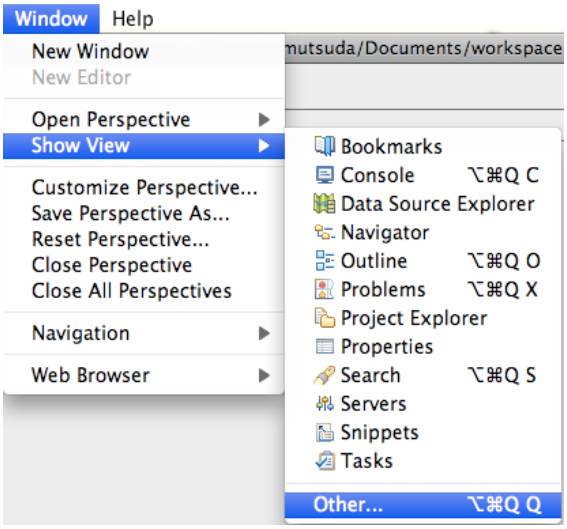
Afegim el site: http://exist.sourceforge.net/plugin-update-site/ i fem click a **"Ok"**.



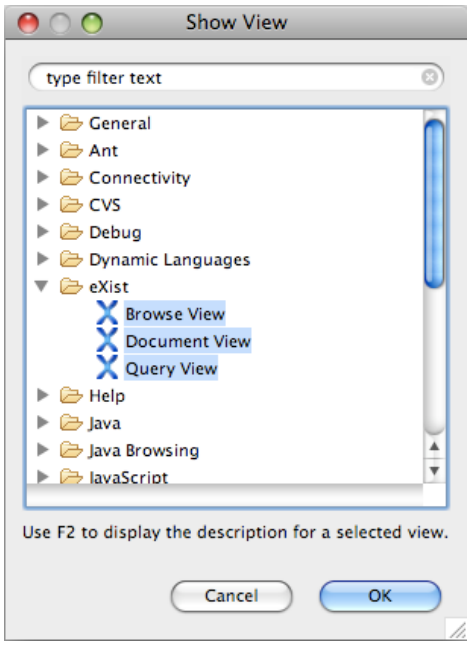
Apareixerà a la llista una cosa similar a al que podeu veure a la imatge. Seleccioneu tot el que sigui nou relatiu a eXist, i feu click al botó **"Install"** d'adalt a la dreta.



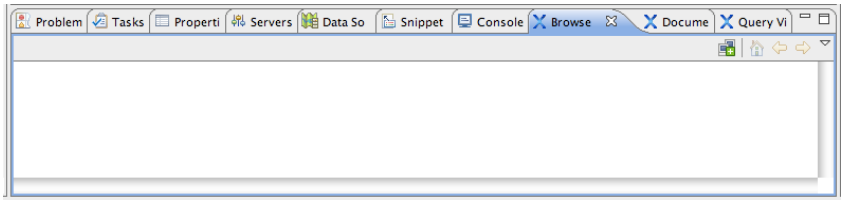
Un cop instal·lat, haurem de fer que les noves pestanyes d'eXist es mostrin a la vista de l'Eclipse per tal de poder-les fer servir. Aneu a **"Window" -> "Show View" -> "Other..."**



Seleccioneu les tres vistes que englobla la carpeta eXist.

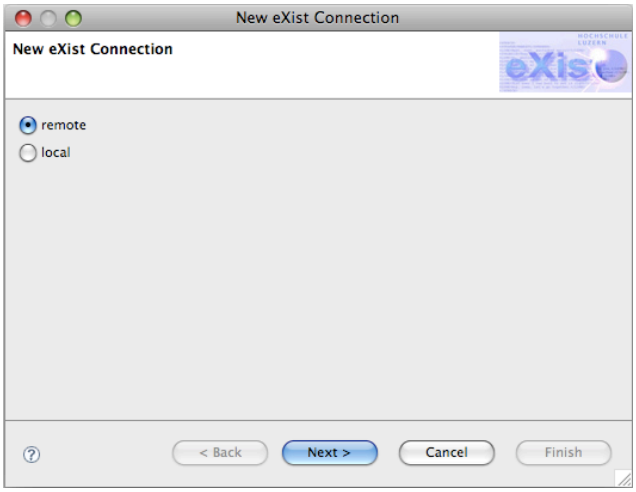


Us apareixeran abaix tres pestanyes amb una X blava a l'esquerra. Browse, Document i Query view.

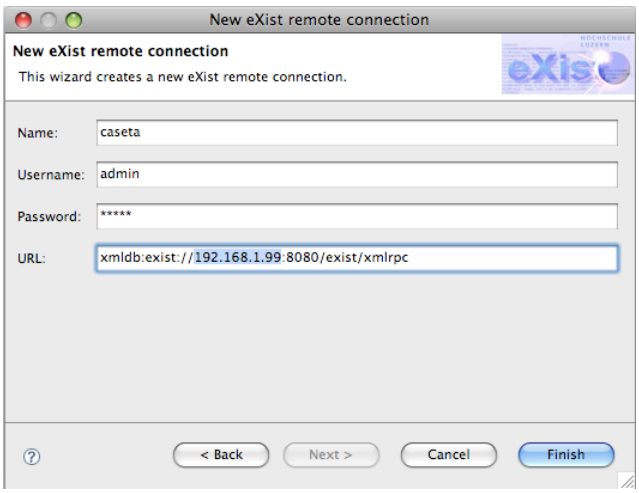


Configuració del Plug-in d'eXist per a Eclipse

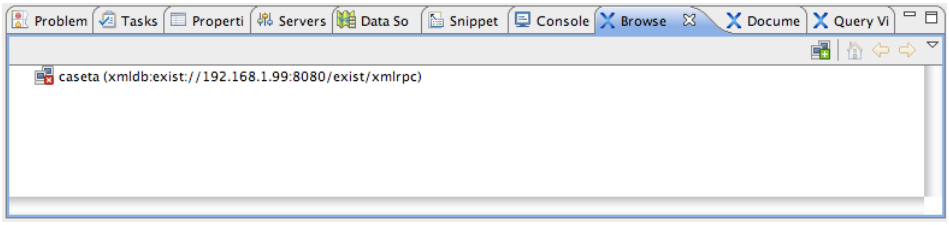
Per tal d'establir la comunicació entre el Plug-in d'eXist i el mateix eXist hem d'afegir un perfil de connexió. Això és fa anant a la **"Browse View"** que recentment hem fet que es mostri, i fent click a la icona d'adalt a la dreta que mostra dos ordinadors amb un **"+"** entremig. Se us demanarà si la connexió a eXist serà local o remota. Això dependrà de si l'eclipse el feu servir des del mateix ordinador on teniu instal·lat eXist, o no.
En el meu cas tinc eXist instal·lat a un altre ordinador, i per tant seleccionaré l'opció remote.



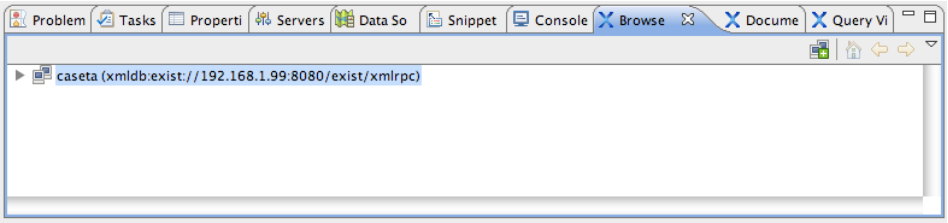
Haureu d'especificar un nom pel perfil, i també el nom d'usuari i contrasenya amb què prèviament heu configurat eXist durant la instal·lació. La URL serà de la forma **xmldb:exist://xxx.xxx.xxx.xxx:8080/exist/xmlrpc** on **xxx.xxx.xxx.xxx** és la IP de l'ordinador on teniu eXist instal·lat. Si esteu connectats localment, heu de fer servir la IP interna, probablement de l'estil 192.168.1.x ó 192.168.0.x ó 10.10.x.x ó 172.26.0.x ó similar. En canvi si us hi esteu connectant des de l'exterior, heu de fer servir la vostra IP pública, que si és estàtica sempre serà la mateixa. En aquest cas concret m'estic connectant a l'ordinador que hi ha a la sala d'estar, així que la IP que faig servir és interna, però durant el desenvolupament del projecte sempre tenia creats dos perfils, el **"caseta"** per quan m'hi connectava des de casa, i l'**"extern"** per quan em connectava a l'ordinador de la sala des de la biblioteca o la universitat.



Un cop creeu el perfil, us apareixerà a la llista de connexions.



Per connectar-vos-hi només cal que feu doble click, o botó dret **"Connect"**. Veureu que la icona canviarà i la creu vermella desapareixerà un cop estigueu connectats.

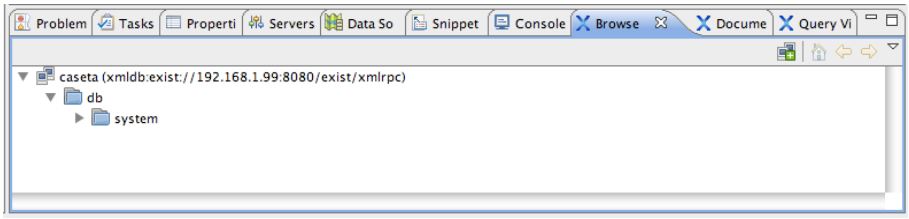


Si heu aconseguit arribar fins aquí, ja heu pogut establir una connexió externa a eXist. El següent pas és comprovar que efectivament estem ben connectats i podem crear un fitxer on emmagatzemarem les dades. Això ho farem des del mateix Plug-in.

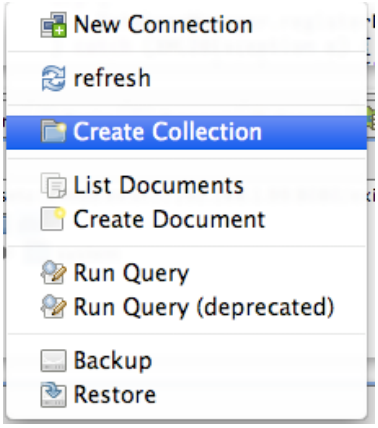
La primera col·lecció

Creant una col·lecció

Una col·lecció amb eXist no és més que un conjunt de fitxers que conformen la nostra base de dades. Poden ser per exemple un fitxer amb informació de tots els països del món, i un altre fitxer amb els noms dels usuaris del nostre sistema. Com a exemple fàcil de seguir, crearem una col·lecció anomenada **"dbExemple"** on ficarem un sol fitxer amb informació de persones.



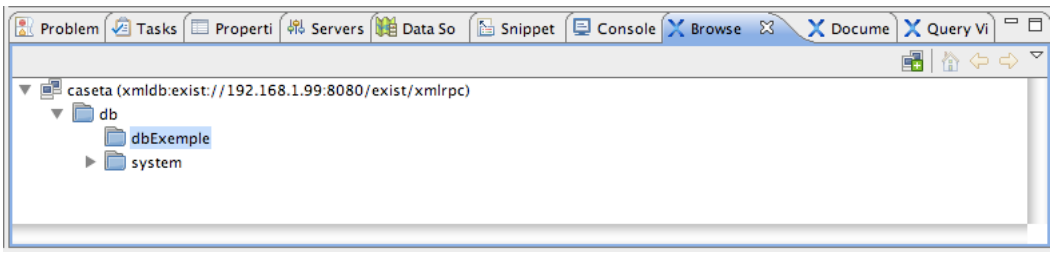
El primer pas és exandir l'arbre de continguts del perfil un cop hi estem connectats, fent click a la fletxa de l'esquerra. Veurem que apareix una carpeta **"db"** amb una subcarpeta **"system"**. Nosaltres crearem la nostra col·lecció al mateix nivell que la carpeta **"system"**, és a dir, com a subcarpeta de **"db"**.
Per fer-ho fem click dret sobre la carpeta **"db"** i seleccionem l'opció **"Create Collection"**.



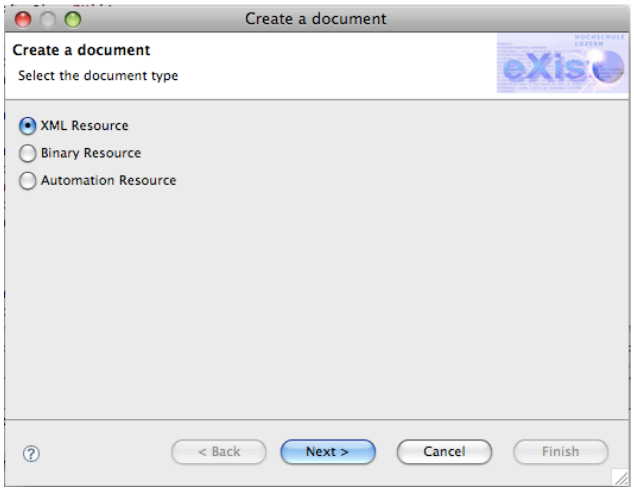
Això ens obrirà una altra finestra on se'ns demanarà el nom de la nova col·lecció. Jo li he dit **"dbExemple"**.

Creant un fitxer dins d'una col·lecció

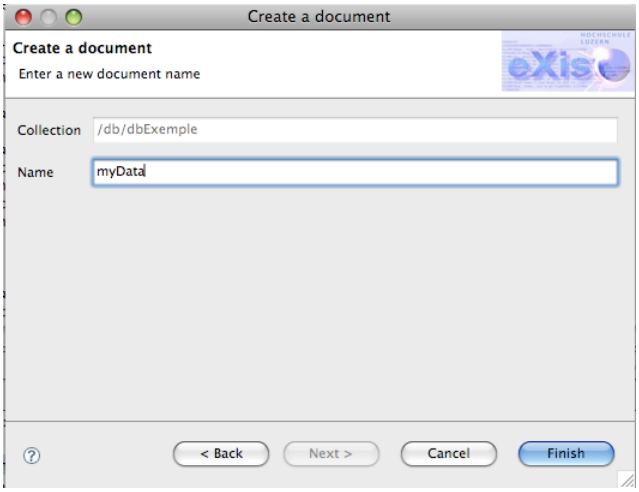
Un cop creada la col·lecció, crearem el fitxer que contindrà les dades a dins d'aquesta col·lecció, fent click dret sobre la nova col·lecció i seleccionant l'opció **"Create Document"**.



Seleccionarem l'opció **"XML Resource"**.



Li direm per exemple, **"myData".**

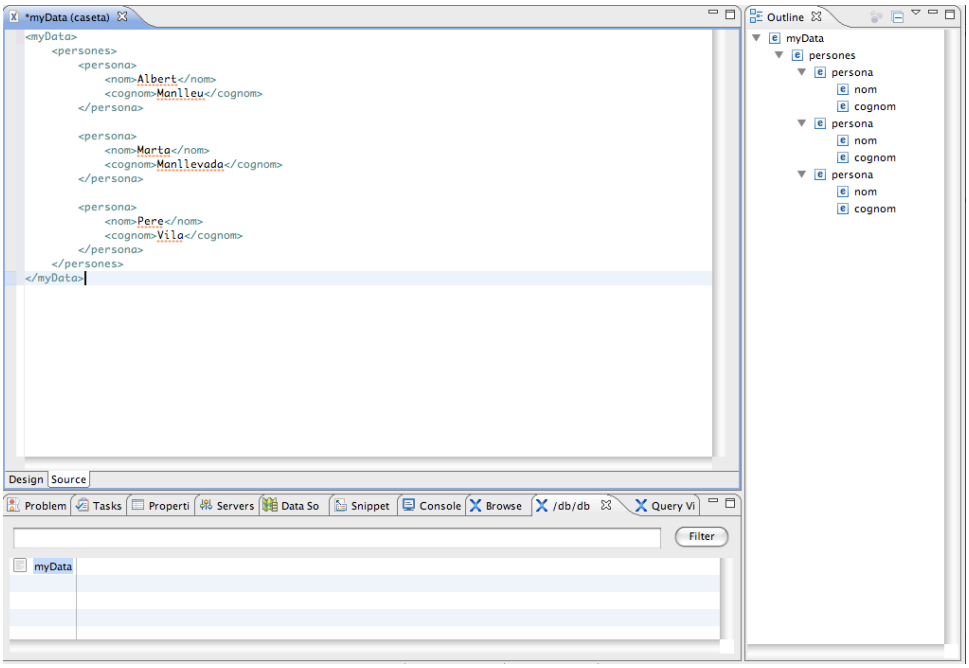


I un cop creat, si fem doble click a la carpeta **"dbExemple"**, ens apareixerà el seu contingut en una altra pestanya, a la pestanya que abans es deia **"Browse View"** però que ara tindrà el nom i la ruta de la carpeta que conté el fitxer nou.

Poblant la base de dades amb una mica d'informació

Si fem doble click al fitxer **"myData"**, se'ns obrirà un editor en blanc, només amb la paraula **"<template/>"**. Aquesta és la fila que fa la base de dades quan encara està buida del tot.

La idea és que a mode d'exemple, omplim aquest fitxer per exemple de la següent forma. Les dades d'aquest fitxer són bàsicament el nom i primer cognom de tres persones, el Pere, la Marta i l'Albert.

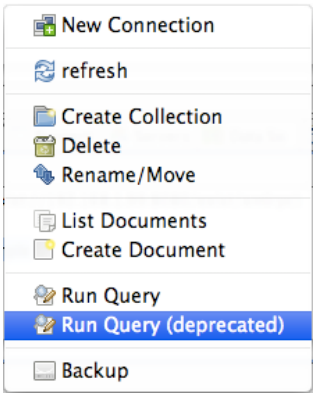


Podeu veure com a la dreta, a la zona **"Outline"** veiem l'estructura jeràrquica que van agafant les nostres dades. Això és molt útil quan la base de dades comença a créixer molt, i volem fer comprovacions ràpides. Si copiant el contingut d'aquesta captura us equivoqueu en algun tag, probablement veureu coses que no tenen gaire sentit al **"Outline",** i fins i tot, si cometeu algun error de sintaxi XML és possible que el Plug-in no us permeti guardar el fitxer. De moment això no té més solució que mirar línia per línia on pot ser que ens haguem equivocat, amb ajuda és clar de la vista **"Outline".**

La primera XQuery

Ara que ja tenim certa informació a la base de dades, ja li podem fer preguntes, per exemple, vull saber el cognom de totes les persones que es diguin "Albert". Això, traduït a XQuery, seria alguna cosa com:

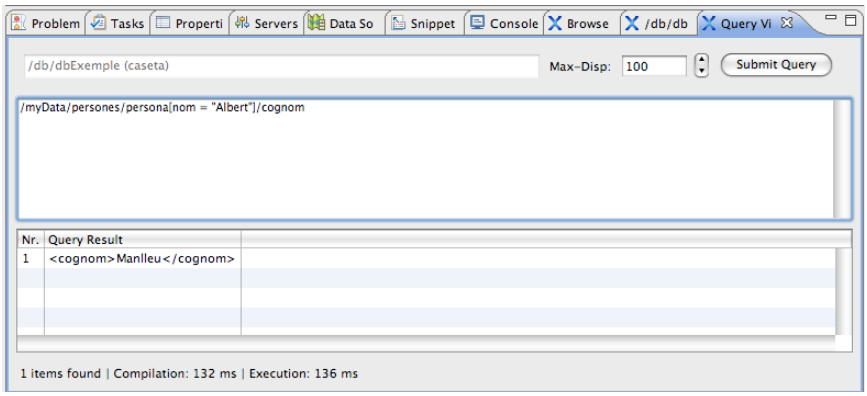
```
/myData/persones/persona[nom = "Albert"]/cognom
```



Amb *"myData/persones/persona"*, ens endinsem en la jerarquia fins a arribar al nivell de les persones. Amb el *corxet*, especifiquem que ens volem centrar en les persones que segueixen una condició, en aquest cas, que el seu camp *"nom"* és exactament igual a *"Albert"*, i finalment amb el *"/cognom"* especifiquem que volem saber-ne el cognom. Això que sembla tant extrany, després acaba sent molt fàcil d'entendre, però requereix pràctica. Al final d'aquest manual trobareu exemples més complexos.

Per tal de poder fer aquest crida des del Plug-in, hem d'especificar sobre quina col·lecció volem llançar l'XQuery. Això es fent click dret sobre la col·lecció **"dbExemple"**, i seleccionant l'opció **"Run Query (Deprecated)".**

Un cop fet això, des de la pestanya **"Query View"** ja podeu començar a llançar crides contra la base de dades. Per provar, intenteu fer la crida que us he posat d'exmple. Només l'heu d'escriure a l'espai en blanc, i clicar el botó de **"Submit Query".** En qüestió de dècimes de segon veureu com eXist, des d'allà on sigui que el tingueu instal·lat, us retorna la solució a la vostra crida: **"Manlleu"**.



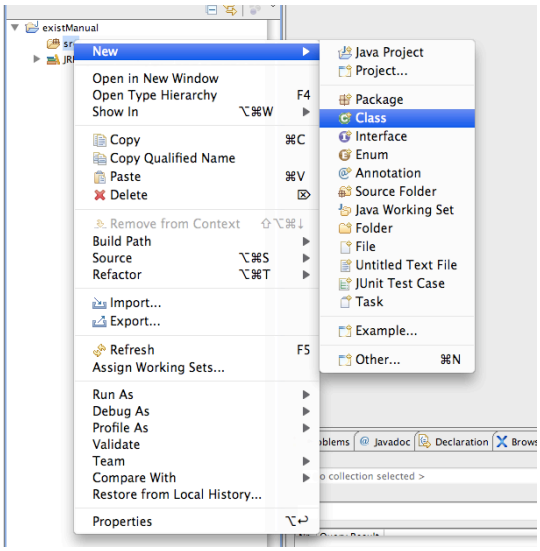
El següent pas serà aconseguir això mateix, però des de Java, és a dir aconseguint, per exemple, guardar aquest resultat en una variable de tipus String, per després poder-ne fer el que sigui.

La interfície de comunicació Java

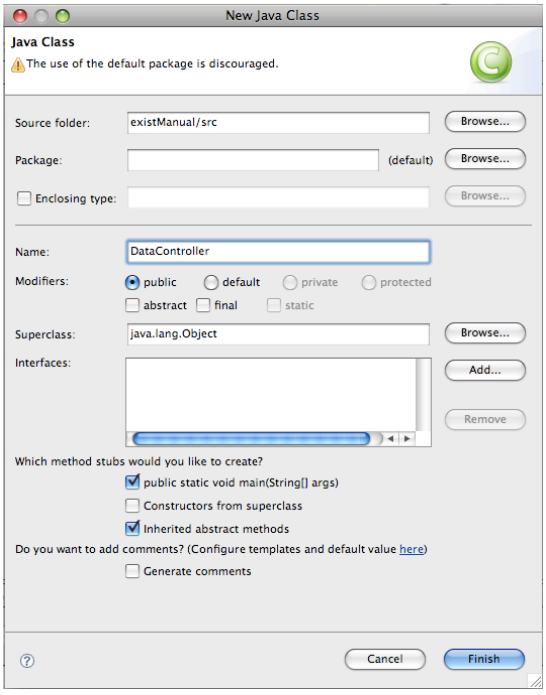
Creant la classe

Per tal de comunicar el nostre programa amb la base de dades, crearem una classe que anomenarem **"Data Controller".** Aquesta classe s'encarregarà de tenir les funcions necessàries per tal d'establir la connexió amb la base de dades i també de fer-li les crides pertinents.

Per tal de crear la classe, farem botó dret sobre el projecte i seleccionarem l'opció **"New"->"Class"**.



En la finestra que se'ns obre especificarem el nom que li volem donar a la classe i farem click a **"Finish"**.



Aquesta classe que acabem de crear contindrà tres variables estàtiques: l'**URI** on es troba la col·lecció, una **col·lecció**, i un **servei.** Aquestes classes són necessàries tant per establir la connexió com per comunicar-se.

Com a operacions, la classe en contindrà dues: **connect,** que s'encarregarà de carregar la instància de la connexió per tal de poder fer les diverses crides a la col·lecció, i **doQuery,** una operació que bàsicament s'encarrega d'executar l'operació que li passem per paràmetre (com a String) a la base de dades eXist.

Creant aquestes dues operacions tindrem un codi més fàcil de llegir i molt més net. Per tant, des de l'exterior, l'únic que hauria de fer la capa de domini per escriure o llegir alguna cosa de la base de dades seria crear una instància del **DataController**, cridar la operació **connect()** de la instància creada, i finalment anar cridant totes les operacions d'actualització o selecció que volguem fer sobre les dades mitjançant l'operació **doQuery(String).** Per tal d'aïllar una mica més el domini de la capa de dades, el que farem serà implementar totes les operacions que hagi de necessitar el domini a la mateixa classe **DataController**. D'aquesta manera, si des de domini volen obtenir el cognom de les persones anomenades Albert, només hauran de cridar l'operació **obtenirCognomAlbert()** que haurem implementat a la classe **DataController** i que internament farà una crida a **doQuery()** passant-li l'XQuery que hem vist anteriorment per paràmetre.

Per tal de fer fàcil de seguir aquest manual faré totes les crides que suposadament es farien desde domini, des de la mateixa classe **DataController** fent servir un **main**. Si esteu fent el projecte i sou els encarregats de la capa de dades, probablement voldreu fer totes les proves del correcte funcionament de la classe des del **main** de la mateixa classe, de la mateixa manera que veurem en aquest manual.

En aquest manual no perdré el temps explicant com funcionen les operacions **connect()** o **doQuery()**, ni les variables de la classe, si no que em limitaré a parlar d'elles com si fossin caixes negres.

El codi font de la classe **DataController** el podeu descarregar de http://www.llima.net/mutsuda/DataController.java. De totes formes, per si de cas aquest servidor ja no existeix quan estigueu llegint aquest manual, escriuré tot el codi a continuació i només haureu de copiar-lo i enganxar-lo a la classe que acabeu de crear.

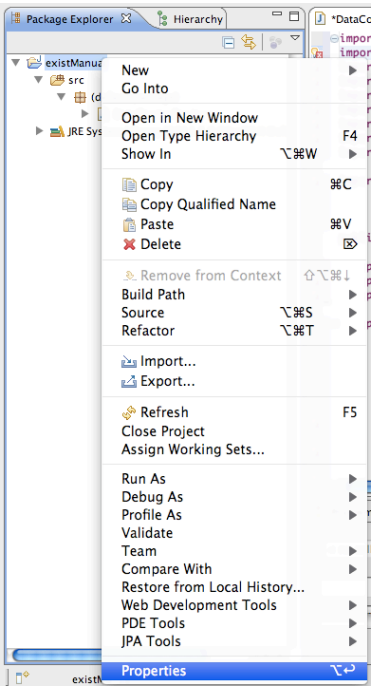
Com podreu veure al codi, heu de substituir les *"x"* de la String URI per les de la IP on tingueu ubicat eXist. Exactament la mateixa IP que heu introduït quan configuràvem el Plug-in.

Veureu que el codi conté el que he mencionat anteriorment, un main, i dues operacions.

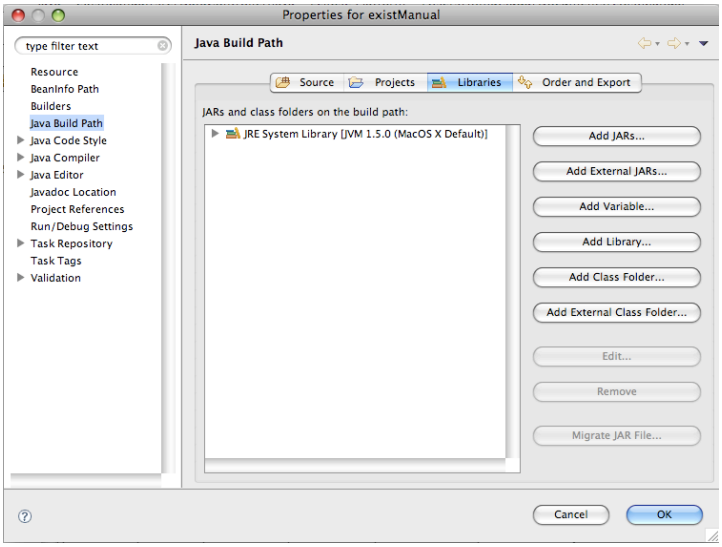
Un cop hagueu enganxat aquest codi a la classe que heu creat, veureu que us dóna un munt d'errors perquè no troba les APIs necessàries. El següent pas és especificar la ruta cap a elles perquè el codi pugui funcionar.

```
import javax.xml.transform.OutputKeys;
import org.xmldb.api.DatabaseManager;
import org.xmldb.api.base.Collection;
import org.xmldb.api.base.Database;
import org.xmldb.api.base.Resource;
import org.xmldb.api.base.ResourceIterator;
import org.xmldb.api.base.ResourceSet;
import org.xmldb.api.base.XMLDBException;
import org.xmldb.api.modules.XPathQueryService;


public class DataController {

	protected static String URI = "xmldb:exist://xxx.xxx.x.xx:8080/exist/xmlrpc/dbExemple";
	public static Collection col;
	public static XPathQueryService service;


	public static void main(String[] args) throws Exception {
		connect();
	}


	public static void connect() throws Exception {
		String driver = "org.exist.xmldb.DatabaseImpl";
		// initialize database driver
		Class<?> cl = null;
		try {
			cl = Class.forName(driver);
		} catch (Exception e) {
			throw new Exception("Error en inicialitzacio del driver1");
		}
		Database database = null;
		try {
			database = (Database) cl.newInstance();
		} catch (InstantiationException e) {
			throw new Exception("Error en inicialitzacio del driver2");
		} catch (IllegalAccessException e) {
			throw new Exception("Error en inicialitzacio del driver3");
		}

		try {
			DatabaseManager.registerDatabase(database);
		} catch (XMLDBException e) {
			throw new Exception("Error en inicialitzacio del driver4");
		}

		try {
			col = DatabaseManager.getCollection(URI);
			col.setProperty(OutputKeys.INDENT, "no");
			service = (XPathQueryService) col.getService("XPathQueryService",
					"1.0");
			service.setProperty("indent", "yes");
		} catch (XMLDBException e) {
			throw new Exception("Error en XMLDB");
		}
	}


	private static String doQuery(String query) throws Exception {
		String res = "";
		try {
			ResourceSet result = service.query(query);
			ResourceIterator i;
			i = result.getIterator();
			while (i.hasMoreResources()) {
				Resource r = i.nextResource();
				res = res.concat((String) r.getContent());
			}
		} catch (Exception e) {
			throw new Exception("Query Error");
		}
		return res;
	}
}
```

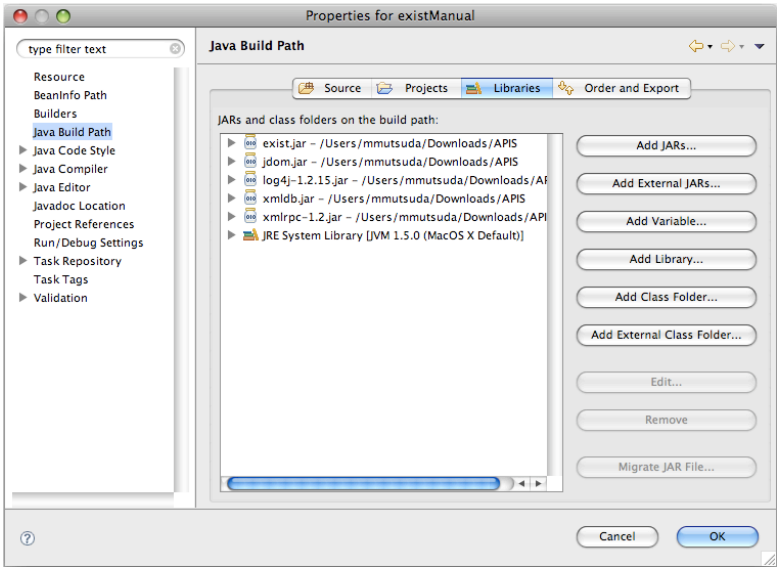
Especificant la ruta de les APIs necessàries



Per tal d'especificar la ubicació de les APIs primer les haureu de descarregar de l'adreça que es menciona a l'apartat del manual **"Abans de començar..."**. Un cop les tingueu, guardeu la carpeta en una ubicació que sapigueu que no canviarà mai. Tot seguit obriu les propietats del projecte fent click al botó dret a sobre del projecte, i seleccionant l'opció **"Properties".** Heu d'ubicar-vos a la pestanya **"Libraries"**, del menú **"Java Build Path"**.



Seleccioneu l'opció **"Add External JARs..."** i busqueu els 5 jars que heu descarregat anteriorment. Un cop afegits us haurien d'aparèixer a la llista que anteriomentnomés contenia la llibreria JRE. Feu click a **"OK".**



Si tot ha anat bé, ja no hauríeu de tenir cap error al codi. Potser teniu algun *"warning"* ja que l'operació **"doQuery"** no està essent feta servir per ningú encara.

Fent la primera crida des de Java

Ara que ja tenim la classe preparada, ja podem fer la nostra primera crida des de Java. Per tal de seguir amb l'exemple, provarem de fer la mateixa crida que abans, que ens retornava el cognom de les persones que es deien Albert. En la nostra població de la base de dades només existeix un Albert, així que només obtenim com a resultat un cognom.

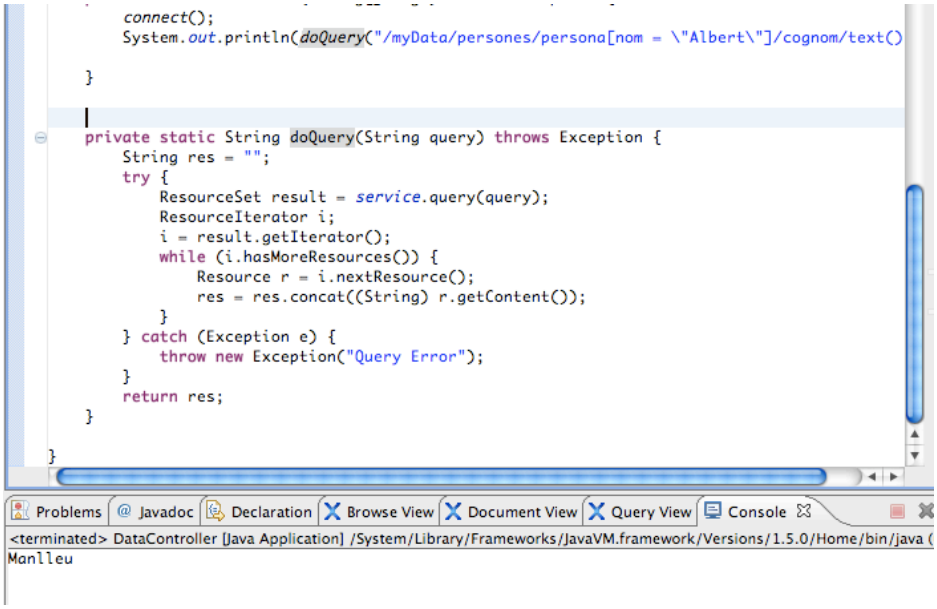
Per tal d'efectuar la crida, farem servir l'operació **"doQuery"** des del main, i escriurem per pantalla el que aquesta ens retorni. Hem de convertir la XQuery que hem fet abans a String, per tant caràcters com per exemple les cometes que hi havia al voltant del nom "Albert", hauran d'estar *"escapades"*.

Afegim aquesta línia de codi al main, tot seguit del **"connect".** L'únic que fa aquesta operació, és escriure per consola el resultat de la crida doQuery(), que retorna una String.

```
System.out.println(doQuery("/myData/persones/persona[nom = \"Albert\"]/cognom/text()"));
```

Observareu que aquest cop he afegit **"/text()"** al final de tot de la XQuery. El que fa això és que enlloc de retornar-me el resultat envoltat dels tags XML, només me'n retorna el text.

Intenteu executar aquest main, amb l'opció **"Run"**. Si tot ha anat bé veureu que per consola escriu exactament el cognom que esperàvem!



Arribats a aquest punt, ja teniu l'esquelet per comunicar-vos amb eXist des de Java, només falta aprofundir una mica en el llenguatge XQuery, XPath i XUpdate.

Per a aprofundir...

Aquests llenguatges ja estan una mica més documentats per la xarxa, però tampoc és per tirar cohets, així que en aquesta secció del manual us donaré alguns consells i posaré alguns exemples de querys per a eXist.

Insert, Update i Delete

Les operacions que més utilitzareu a part de les de consulta, seran les d'actualitzar, esborrar o inserir una dada a la jerarquia. En aquesta secció veurem alguns exemples de com utilitzar aquestes operacions. Seguirem amb l'exemple de bases de dades anterior, myData.

Insert

Posem, per exemple, que volem inserir la persona Haruki Murakami a la nostra base de dades. L'únic que hem de fer doncs, és:

```
update insert <persona><nom>Haruki</nom><cognom>Murakami</cognom></persona> into /myData/persones
```

Com veieu és tan simple com definir la persona en XML amb tots els seus paràmetres, i dir-li on exactament el volem inserir. A la pràctica, probablement els nom i cognom de la persona us vindrien per paràmetre del domini, i per tant, a l'hora de cridar a *doQuery()* hauríeu de concatenar els paràmetres pel mig com a variables.

Per a fer-ho des de Java seria interessant crear una operacio *insereixUsuari* que donats un nom i cognom, cridessin a l'operació *doQuery()* amb els noms que li arriben per paràmetre. Podeu afegir al següent operació al codi per fer la prova.

```
public static void insereixUsuari(String nom, String cognom) throws Exception
{
        doQuery("update  insert  <persona><nom>"+nom+"</nom><cognom>"+cognom+"</cognom></persona>
                 into /myData/persones");
        //*
}
* Aquí està en dues línies perquè no hi cap en una, però al codi feu-ho en una línia o feu servir el
""+"" per concatenar les dues parts de la string, si no la sintaxi serà incorrecta.
```

I substituir la consulta que teniu ara mateix al *main* per:

```
insereixUsuari("Rafael","Sanchez");
```

No esborreu el *connect(),* ja que si no no establirà la connexió amb la base de dades i no hi haurà comunicació.

Ara proveu d'executar el codi, i veureu que us insereix l'usuari Rafael Sanchez a la base de dades.

Update

Imaginem que ara volem substituir el nom de l'usuari que té com a cognom Murakami per Corominas. En el cas d'XQuery, s'utilitza el concepte de *Replace.*

```
update replace /myData/persones/persona[cognom = "Murakami"]/cognom/text()  with "Corominas"
```

Com podeu veure el que fem és especificar exactament a quina persona ens estem referint dient que el cognom és exactament "Murakami", però també hem d'especificar què és el que volem canviar d'aquella persona, que en aquest cas és el cognom, i per tant afegim */cognom* al darrere de la persona en qüestió, i a més la funció */text()* que ens permet

simplement canviar el contingut del camp cognom, si no posessim *text()* hauríem de tornar a escriure *<cognom>Corominas</cognom>*.

Delete

Imaginem que ara volem esborrar l'usuari amb cognom Corominas que hem modificar anteriorment.

```
update delete /myData/persones/persona[cognom = "Corominas"]
```

Amb això l'usuari quedaria completament esborrat. Si només volgúessim esborrar-ne per exemple el node cognom, hauríem de fer:

```
update delete /myData/persones/persona[cognom = "Corominas"]/cognom
```

Una mica més enllà

Ara ja som capaços de consultar, eliminar, modificar i inserir dades a i de la base de dades. Encara no sabem però, fer-ho massivament i condicionalment, és a dir no sabem definir variables o iterar sobre un conjunt obtingut. Si feu servir transformacions XSL també voldreu saber com fer per mostrar els resultats de les querys a la vostra manera, és a dir definir els tags amb què voleu que les dades consultades us siguin retornades.

Llistar els morosos

Per a practicar, afegiu a cada persona de la base de dades un camp *<deute>x</deute>* just a sota del cognom, amb una "x" negativa en alguns casos i positiva. També podeu afegir un nombre més elevat de persones perquè sigui més interessant. El que farem serà llistar els morosos de la base de dades en el següent format:

```
<morosos>
     <moros>
          <nomcognom>Nom Cognom</nomcognom>
          <deu>Deute</deu>
    </moros>
<morosos>
```

I ho farem amb la següent query, que ja és una mica més complexa:

```
<morosos>
{
let $morosos:=/myData/persones/persona[deute>0]
for $moros in $morosos
return
<moros>
      <nomcognom>{$moros/nom/text(), " ", $moros/cognom/text()}</nomcognom>
      <deu>{$moros/deute/text()}</deu>
</moros>
}
</morosos>
```

Com podeu veure hem personalitzat completament el format en què volem obtenir les dades afegint tags que originalment no existeixen a la base de dades. El primer pas és definir la variable *morosos* amb les persones de la base de dades amb un deute superior a 0. Posteriorment, iterem sobre el conjunt *morosos* i per a cada moros n'escrivim el seu nom cognom i deute en el format desitjat. Com veieu això té possibilitats infinites.

Altra vegada, per executar-ho des de Java només ho heu de passar a String anant amb compte amb les cometes i els símbols especials, i veureu que el què us retorna com a String són precisament les persones amb un deute superior a 0.