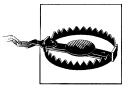
CHAPTER 2
Getting Started

This chapter takes you through the very first steps in using eXist. It handles subjects like downloading and installing, starting and stopping, running the examples and demonstrates some of eXist's capabilities on a "hello world" level. In other words, like the title says, it will get you started.

Feel free to skip this chapter when you think you're beyond this.

Downloading and installing eXist

This section takes you through the steps necessary for getting eXist up and running on a *stand-alone development* system. That is, we keep things simple and don't spend time on more advanced subjects like database security, tuning, performance, running embedded and the likes. These subjects and more are covered in the chapters to come.



Be aware that installing eXist for *production* purposes (e.g. as the engine behind a public website) requires much more thought and planning. Especially security requires attention in those kinds of more public situations. Also: when you plan to use eXist with some really huge datasets, you probably need a different setup than described here. More about this later.

Preconditions

eXist can be installed on most versions of Linux, Windows (XP, Vista and 7 are fine) and Mac OS X. The deciding factor is whether or not your OS support Java version 1.6 or 1.7. If it does, eXist should run.

In order to run the eXist installer, it is necessary to have a working Java Runtime Engine (JRE) or Java Development Kit (JDK), version 1.6 or 1.7. The official recommendation is still to use the 1.6 version but version 1.7 also does the job. Version 1.7 might, in very rare circumstances, cause strange error or warning messages but for normal usage this is very unlikely.

You can download the JDK from *http://www.oracle.com/technetwork/java/javase/downloads/index.html*.

To check whether you have the right Java version (and have installed it correctly), open a command line terminal and type *java -version*. A message telling you which version Java you're running should appear.

Downloading eXist

Downloading eXist is easy. Just go to *http://www.exist-db.org*,, navigate to the download section, pick the right distribution and... download it. For getting started, pick the latest stable distribution. The filename will look like *eXist-db-setup-[version]-revXXXXX.jar*.

This book was based on the 2.0 release of eXist (*eXist-db-setup-2.0-rev18252.jar*), but by the time you read this, a newer version is probably available.

Things to decide upon before installing

Of course you can just go ahead now, run the installer and use the defaults provided. However, there are probably a number of things you want to decide upon *before* firing up the installer:

Installation directory

Where are you going to install eXist? For just a "getting started" installation this is not extremely important and you can use the default provided or any other location you like (provided it is writeable by the installer).

However: There are a number of reasons why the installation directory matters more than usual for a software installation: Firstly, the default for the data directory (where eXist stores its data) is *inside* the installation directory, see below. Secondly, when you're going to write XQuery code on the file system (as opposed to inside the database), the code's location is *inside* the software installation directory too (the subject of where to put your code is handled in "Where to store your application?" on page 155). Thirdly, logging and temporary directories are also inside the installation directory.

Having frequently written locations inside a software installation might be problematic because security sometimes does not allow this or it causes performance degradation. And: when you start to do more serious work, make sure the important locations are included in your backup.

We will refer to the installation directory as *$EXIST_HOME* throughout this book.

Data directory
: This is the directory where eXist stores the content of its database. The installer will propose a default that's *inside* eXist's installation directory. (*$EXIST_HOME/webapp/WEB-INF/data* to be precise). Now if you just plan to play around a bit or do some development work, keep the default. You can always change it later.

  However, if things get serious, like on a production server, make sure that this directory is writeable, sufficiently fast on updates and backed up (which is not always the case for program files directories).

Administrator password
: The installer will ask you to provide an administrator password. This is *not* your operating system's administrator password but the initial password used for eXist's administrator's account (called `admin`). You are strongly encouraged to set an administrator password on all installations of eXist. If you don't, eXist will be open to the world and anybody can get in and hack your system.

Memory settings
: The installer allows you to set the amount of memory reserved for eXist and the internal cache. Common settings are:

  Table 2-1. eXist installation memory settings

Max memory	Cache memory	Remarks
512Mb	64Mb	Don't go any lower than this, eXist will not run properly if you do.
1024Mb	128Mb	This is the default setting and if you don't do anything too outlandish it should be fine
1200Mb	128Mb	This is usually the limit on 32 bit systems
2048Mb	256Mb	If your machine has enough memory to spare, use this

Packages/Apps to install
: For getting started purposes we recommend to keep everything checked (this book assumes that you did!).

Installing eXist

Start the installer in one of the following ways:

For systems with a GUI
: If Java is setup correctly, double clicking the downloaded *eXist-db-setup-[version]-revXXXXX.jar* file will fire up the installer.

On all systems, from the command line
: Open a terminal/command line window and enter the following command: *java –jar eXist-db-setup-[version]-revXXXXX.jar* (where *eXist-db-setup-[version]-revXXXXX.jar* is of course the name of the file you just downloaded).

As usual with installers, follow the instructions on the screen to complete the installation. You're asked to enter the information prepared in the previous section. Let the installer run its course, and that's it!

Post-installation checks

By default, eXist uses two IP ports:

Port 8080
: This port is used for all the normal http communication

Port 8443
: This port is used for the confidential https communication

If one of these ports is used by another application on your system, you either have to make this other application change its ports or change the port settings for eXist.

The easiest way to find out if something is using these ports is, before starting eXist, to surf to *http://localhost:8080/* and *https://localhost:8443/*. When nothing happens the ports are probably free and you can go ahead.

Changing eXist's IP port usage is explained in "Changing Jetty settings: port number and URL prefix" on page 166

This book assumes eXist is running on localhost using the standard IP port numbers 8080 and 8443, so you'll see URL's like *http://localhost:8080/…* sprinkled throughout the book.

Starting and stopping eXist

If you're on a system with a GUI, the installer created a menu entry and/or a desktop icon/shortcut called *eXist database*. If you're on a command line only system, go to "Starting and stopping from the command line" on page 17.

Opening the *eXist database* icon/shortcut starts eXist and also fires up a little control application that should be visible in the system tray (or whatever it's called on your system) as a dotted X. For instance, on my Windows 7 machine it looks like this:

Figure 2-1. The eXist control application in a Windows 7 system tray

Clicking it opens a little menu that gives you further control of eXist (like stopping) and do a few other useful things:

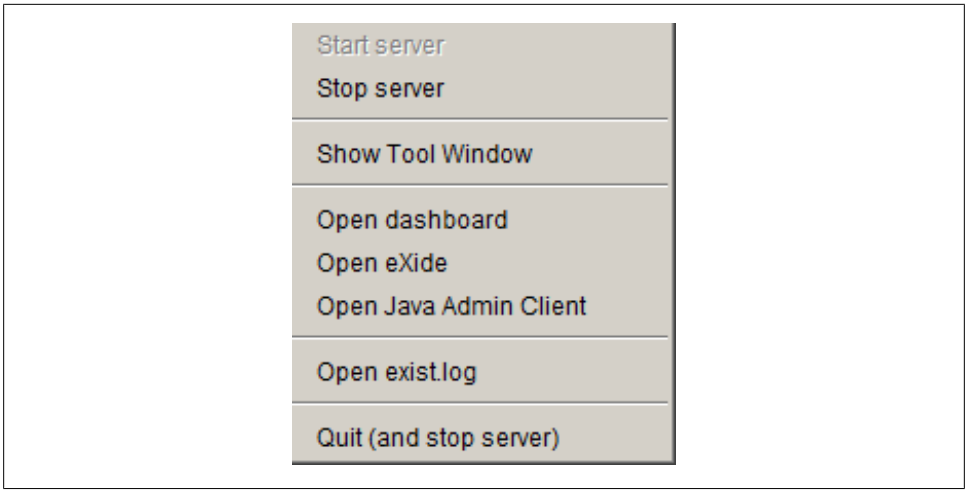


Figure 2-2. The menu of the eXist control application

If for some reason this doesn't work, open a command window in *$EXIST_HOME* and type *java -jar start.jar*. This should fire up the control application and the database. If this works you're probably best of creating a shortcut or menu entry for it manually. If still nothing happens read the next section.

After starting the database, open your browser and surf to *http://localhost:8080/exist*. If a nicely tiled screen appears *(ike in "The dashboard" on page 18) you've succeeded!

Starting and stopping from the command line

If you don't (want to or can't) work with the GUI niceties, you can also start eXist from the command line. For this, open up a command line window and navigate to *$EXIST_HOME/bin*. There you'll find several command files in both the Windows (**.bat* versions) and Unix/Mac (**.sh* versions) variants. For starting and stopping, do the following:

startup
: This will fire up eXist

shutdown -p {adminpassword}
: This will stop the running eXist instance. It needs the administrator password.

A first tour around town

This section will give you a quick tour around eXist's highlights and attractions, both with the user interface and in terms of what's on your disk.

The dashboard

The home screen of eXist since 2.0, *http://localhost:8080/exist*, is called the dashboard: A set of tiles linking to various applications and utilities. The initial set only shows the basic eXist provided ones, but if you start developing applications of your own and use the package repository (/**TBD: REF NOT YET KNOWN**/), these will appear here too.

Now most of the functionality provided through the Dashboard, stuff like eXide and the function documentation, is important: you will probably use it often. It is therefore well worth your time to familiarize yourself with the smörgåsbord offered.

Figure 2-3. The eXist dashboard

Java Admin Client
: This provides a JNLP (Java Network Launching Protocol) link to eXist's Java Admin Client application. Use this if you want to access an eXist installation remotely, from a system that does not have eXist installed. For local use you're better of starting the Javaa Admin Client directly, e.g. through the control application's

menu as shown in Figure 2-2. More about the Java Admin Client in "The Java Admin Client" on page 22.

Admin Web Application
: This gives access to the original (from pre 2.0) administrator web client application. There is still some functionality there that is not (yet?) turned into the new interface, like profiling queries, index overview, etc.

Collections
: Starts a collection browser that enable you to control the contents of the database.

Package Manager
: A *Package* is a set of related file that together provide some kind of functionality, for instance an application. The Package Manager allows you to manage (view, install, de-install) packages in your eXist database. If you open it you can see that most of the functionality behind the tiles of the dashboard is provided by separate packages.

  Packages can come from the eXist public repository: You can see the packages available there by selecting the *available* option at the top of the Package Manager. Or they can be distributed as a separate Package file with the extension *.xar*.

  It is also possible (and even advised!) to design your own applications for use with the Package Manager and distribute them using *.xar* Package files. How to do this is explained in **/TBD: Link to packaging section/**.

XQuery Function Documentation
: Now this is an application you'll probably use very often: It provides an overview of all the functions available and their documentation (if any) in both the standard eXist extension modules and your own XQuery modules. More about modules in Chapter 6.

eXist-db Documentation App
: This app provides access to the eXist documentation.

eXide
: eXide is a cool, handy and fully integrated editor for working with XQuery, XML and other resources stored in eXist. You can use it from writing complete applications to fiddling around and experimenting. Don't miss it, more information in "eXide" on page 219.

User Manager
: This allows you to control the user population of the eXist database. You can create, edit and delete users, groups, etc.

Shutdown, Backup
: Applications that do what their title suggests.

eXist-db Demo Apps
: A collection of applications that demonstrate some of eXist's capabilities.

betterForm Feature Explorer, betterForm Demos, XSLTForms Demo
: eXist has two ways of doing XForms build in: betterForms and XSLTForms. These applications provide you with demos and overviews.

Since you're going to use some of these very often, consider making bookmarks for them in your browser.

Play around

Now if you're like us, at this point you'll want to play around, try some XQuery, store some XML and more of these familiarization rituals. Get your feet wet and splash around (without going in too deep water). Here is the quick recipe:

- Open the dashboard: *http://localhost:8080/exist*
- Click on the eXide tile
- Click *Login* and log in as `admin` (don't bother with permissions now...)
- Directly type some XQuery and run it
- If you want to see what's in the database, click *File*, *Manage* (or *ctrl-shift-M*)
- If you want to save your work or put some related files together, create a collection for this underneath */db* or */db/apps*.

What's in your database

You can look inside the database using, among others, the Collections app in the Dashboard or the eXist Java Admin Client ("The Java Admin Client" on page 22). You'll see something that looks like a disk directory structure (but of course isn't). To get used to the terminology: What you think inside your database is a "directory", is called a *collection* in XML Database geekspeak (more about this in "Terminology" on page 34). The most important ones:

/db
: The root collection in the database is always */db*. You can't change this.

/db/system
: This is where eXist stores important configuration information, e.g. about users, groups, versioning, etc. You shouldn't change any of this information by hand or programmatically, with the exception of what's inside */db/system/config*.

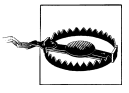
/db/system/config
: This collection is used to store the collection specific configuration for eXist, like validation, indexes, triggers, etc. If you look underneath, you'll find a (partial) copy of the existing database structure with *collection.xconf* files here and there. These

(XML) files contain the collection configuration. More about this in for instance "Implicit validation" on page 182 and "Configuring indexes" on page 193.

*/db/apps/**
: These are the root directories for the packages, installed during installation and manually later. Underneath these is their code and data. If you're ever going to write applications yourself (Chapter 8) you'll create your own sub-collection here.

What's on your disk

Here are some interesting and/or important locations on your disk for eXist:



There are whispers through the grapevine that the basic file structure will change in future versions, so be aware if you use this book with a later version than 2.0.

$EXIST_HOME/
: This is eXist's home directory

$EXIST_HOME/conf.xml
: This is eXist's main configuration file. If you peek inside (its well documented), you'll find entries for, for instance, all kinds of default behaviour, the location of the database (in `db-connection/@files`), cache sizes, the indexer, the built in XQuery modules, etc.

$EXIST_HOME/tools/jetty/etc/jetty.xml
: This is the Jetty web server's configuration file (eXist uses Jetty to communicate with the world). There are several interesting things you might want to change using this file, like the IP port numbers and the default URL prefix (which is now `exist/`).

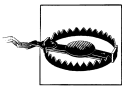
$EXIST_HOME/webapp/
: This is eXist's web application directory. If you are going to write code on the file system (and not within the database), this is where you'll put it. More information about where to put your code in "Where to store your application?" on page 155.

$EXIST_HOME/webapp/WEB-INF/
: This location defines the eXist web application. It holds several important configuration files and is the default base location for the database and the log files.

$EXIST_HOME/webapp/WEB-INF/controller-config.xml
: This tells eXist what to do when a request with a certain URL is entered. More information in "The controller-config.xml configuration file" on page 167.

$EXIST_HOME/webapp/WEB-INF/data/
: This is the default location for eXist's database (you might have installed it elsewhere during the installation process).

If you peek inside this directory, you'll find underneath the *fs* sub-directory all the non-XML files stored in the database. However, your XML files are not there and have seemingly disappeared. Don't despair, they're absorbed in the `*.dbx` files you see in the root of the database directory. More information about this in "Help: Where is my XML?" on page 33.



You might be tempted to change the non-XML content underneath the *fs* sub-directory directly. However *don't do this*. It will ruin the database's internal administration. Only use the normal mechanisms for this, like for instance WebDav, the Dashboard or the Client Tool.

$EXIST_HOME/webapp/WEB-INF/logs/
: Here you'll find several log files that can help you in finding out what's going on underneath eXist's hood.

The Java Admin Client

Through the eXist controller application (visible in the system tray) you can start what is called the *Java Admin Client*. This pops-up a small and, admittedly rather old-fashioned looking, program. It allows you to do maintenance work on the database like backup and restore, imports and exports, checking and setting properties, creating collections, etc. This is how it looks on a freshly installed database:

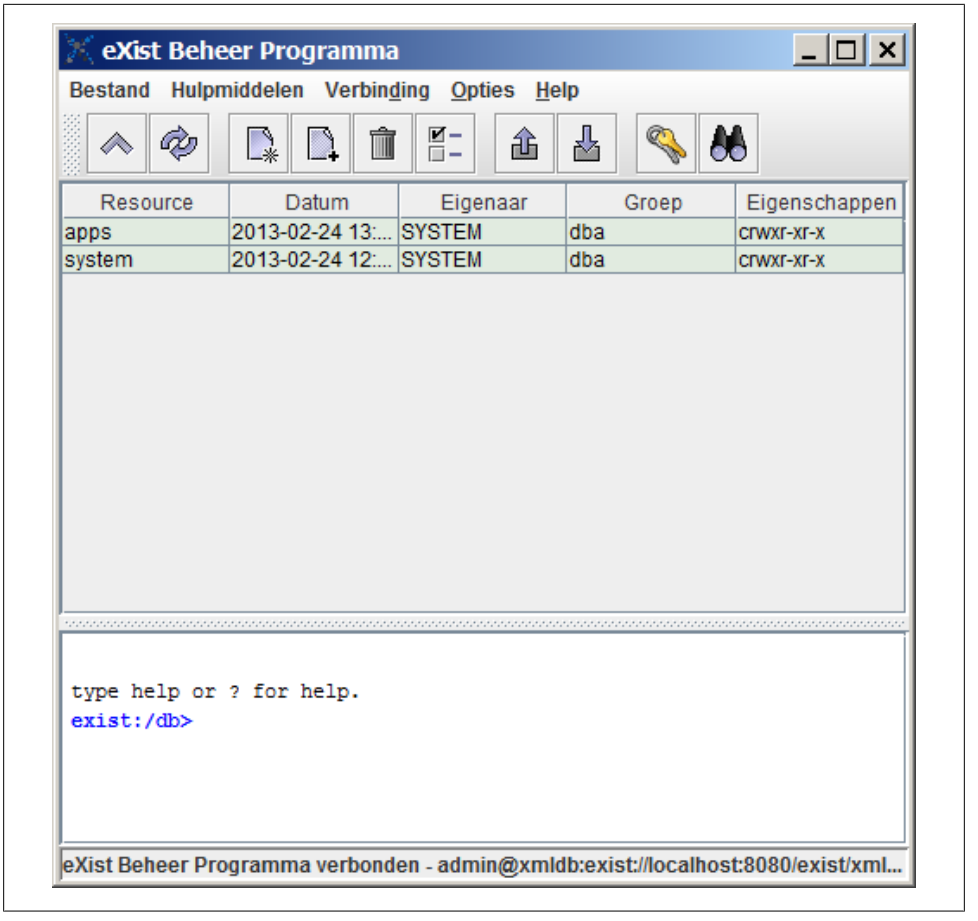


Figure 2-4. The main screen of the eXist Client Tool

The eXist Client tool is a standard GUI application and its functionality speaks for itself.

Most of this tool's functionality is also present in the new Dashboard application, so there's a good chance you'll never need it. However there are circumstances that it can be useful, for instance when the Dashboard application is broken or when you're working on a production server where you don't want the Dashboard to be present.

Installing and using the example code

This book comes with a set of example code. This code is distributed through the O'Reilly website **/TBD: Link?/** in a *.xar* package file. To install the sample code:

- Download the eXist book package file

- Fire up the eXist dashboard and click on the Package Manager tile
- Click on the upload application icon (top left, looks like a cogwheel)
- Select the *.xar* file and press submit

After installation, the sample code is available as another tile in the Dashboard. It runs as a very simple application that allows you quick acces to running the examples.

Getting stuff in and out of the database

eXist is an XML database. It stores XML documents. It can also hold your executing XQuery and other code and your assets. So how do you get files in and out of it?

Collections app
: Surf to eXist's Collections application (available through the Dashboard). This allows you to browse through the contents of your database and maintain this.

eXide
: eXist has a built-in native IDE called eXide which has facilities for down- and uploading files. Click *File, Manage* (or *ctrl-shift-M*).

WebDav
: eXist's WebDav (the Web-based Distributed Authoring and Versioning protocol) interface allows you to access the contents of the database like it was just another file store available to your OS. The address to use is `http://localhost:8080/exist/webdav/db/` or, when your OS requires safe URL's (like Windows 7), `https://localhost:8443/exist/webdav/db/` .

  Now how exactly to work with a WebDav server and what client tool to use is very platform specific and outside the scope of this book. Some operating systems, like Windows, will allow you to integrate it more or less into the normal file browsing capabilities, others need special client tools.

Java Admin Client
: eXist's Java Admin Client tool (see "The Java Admin Client" on page 22) also has some basic facilities for getting files in and out of the database.

External IDE
: Some external IDE's, oXygen is a good example, provide you with the option to work with eXist natively. This includes importing/exporting files. More information in "oXygen" on page 221.

Programmatically
: Of course you can do it programmatically. Write some XQuery code that performs what you want on the database. That's ok within applications, but a bit cumbersome for now. More information in "Controlling the database from code" on page 51.

Ant
: eXist provides a library for the Ant build tool to automate common tasks like backup/restore or importing a bunch of files. This method is recommended if you have the need for repeating batch tasks on your database. More information in "Ant and eXist" on page 226.

Hello eXist!

This section performs a first exploration of the fundamental mechanisms in eXist. How can you get it to actually do something: store/retrieve/filter information, show a web page, transform XML, etc. In other words: It's an extended "Hello world" section in which, in a (very) shallow way, the important processing features of the platform are touched upon.

This section assumes you've installed the example code and know how to access it, as described in "Installing and using the example code" on page 23.

Hello data

In the example code belonging to this book, there is an XML file in */db/apps/eXist-book/getting-started/xml-example.xml* that looks like this:

Example 2-1. Example XML file

```
<Items>
    <Item name="Bogus item">This is a complete bogus item</Item>
    <Item name="Funny item">Ha, ha, very funny indeed!</Item>
</Items>
```

Accessing data (and also scripts) is done through what is called the eXist *REST Interface*. To see it in action, fire up your browser and surf to:

```
http://localhost:8080/exist/rest/apps/eXist-book/getting-started/xml-example.xml
```

The result is that you see exactly the file from Example 2-1. Not very impressive maybe but, hey, this is only the beginning.

The REST interface allows you to directly query this file. For instance, assume you're interested in the first `Item` only. You can do this by adding a `_query` parameter:

```
http://localhost:8080/exist/rest/eXist-book/getting-started/xml-example.xml?_query=//
Item[1]
```

The result will be:

```
<exist:result xmlns:exist="http://exist.sourceforge.net/NS/exist" exist:hits="1"
exist:start="1" exist:count="1">
  <Item name="Bogus item">This is a complete bogus item</Item>
</exist:result>
```

Because it's a query, eXist wraps the result in an `exist:result` element with additional information in its attributes. There are other query parameters will let you limit the size of the result set and even retrieve the results block by block. More information about the REST interface in "Querying the database using REST" on page 40.

Hello XQuery

Of course, the main language when dealing with eXist is XQuery, *the* language to access XML databases. Put your XQuery script in a file (or database document) with the extension `.xql`.Example 2-2 shows you a very basic example that outputs some XML.

Example 2-2. A very basic XQuery example returning XML

```
xquery version "1.0";

let $msg := 'Hello XQuery'
return
  <results timestamp="{current-dateTime()}">
     <message>{$msg}</message>
  </results>
```

Now what if you want show the result as a HTML page? That's called *serialization* and here is one of the ways to change this:

Example 2-3. A very basic XQuery example returning HTML

```
xquery version "1.0";

declare option exist:serialize "method=html media-type=text/html";

let $msg := 'Hello XQuery'
return
  <h3>It is now {current-dateTime()} and so {$msg}!</h3>
```

The XQuery initiated reader might have noticed that we did not declare the `exist` namespace prefix. eXist has most eXist specific namespace prefixes pre-declared for you, so you don't have to explicitly mention them in your code.

Hello XSLT

XSLT is built into eXist using the Saxon XSLT processor. The examples contain a simple stylesheet to show you how this works. The stylesheet in Example 2-4 takes the XML from Example 2-1 and turns this into an HTML page.

Example 2-4. Transformation of the example XML into HTML

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html>
      <h1>Item overview</h1>
      <ul>
```

```
        <xsl:for-each select="//Item">
          <li>
            <xsl:value-of select="@name"/>: <xsl:value-of select="."/>
          </li>
        </xsl:for-each>
      </ul>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

To run an XSLT stylesheet over some XML you need to use an *extension module*. Extension modules are, well, extensions to the basic XQuery capabilities. eXist has lots of them and we devote an entire chapter to this subject: Chapter 6. An overview and all function documentation is accessible through the XQuery Function Documentation App, available through the Dashboard.

Transforming documents with XSLT is done with the `transform` extension module. The little XQuery script that performs this transformation is shown in Example 2-5 and its result in Figure 2-5.

Example 2-5. Using XSLT with the transform extension module

```
xquery version "1.0";
declare option exist:serialize "method=html media-type=text/html";

transform:transform( doc('xml-example.xml'), doc('convert-items.xsl'), ())
```

Figure 2-5. Result of the XSLT transformation

Notice that the `transform` extension module was not explicitly declared in the XQuery script. This is done implicitly for you by eXist. The third parameter of `transform:transform`, which is now passed an empty sequence, can contain parameters for the stylesheet.

More about using XSLT transformations within eXist in "XSLT" on page 175.

Hello XInclude

eXist can also do XInclude processing for you. This means that, on the way out, when the final results of an XQuery operation are serialized, this is inspected for `<xi:include>` elements. When found, these references are expanded.

An interesting feature of the XInclude processing is that you can also refer to XQuery scripts. The script is executed and the result included. This is demonstrated in the following XML file:

Example 2-6. An XInclude example

```
<XIncludeEnvelope xmlns:xi="http://www.w3.org/2001/XInclude">
    <xi:include href="xinclude-content.xml"/>
    <xi:include href="hello-world-1.xql"/>
</XIncludeEnvelope>
```

The `hello-world-1.xql` is the XQuery script presented in Example 2-3. The included XML file contains:

Example 2-7. XML fragment to include with XInclude

```
<XIncludeContent>This element was included by the XInclude processing in eXist. Yes!</
XIncludeContent>
```

Now if you retrieve `xinclude-envelope.xml` from the database, the XInclude references are resolved and the result is:

Example 2-8. The result of the XInclude processing

```
<XIncludeEnvelope xmlns:xi="http://www.w3.org/2001/XInclude">
  <XIncludeContent>
    This element was included by the XInclude processing in eXist. Yes!
  </XIncludeContent>
  <results timestamp="2013-02-21T13:12:21.399+01:00">
    <message>Hello XQuery</message>
  </results>
</XIncludeEnvelope>
```

There are more features to the XInclude processing, like `<fallback>` instructions and the ability to pass parameters to XInclude-d XQuery scripts. More about this in "XInclude" on page 180.

Hello XForms

XForms is a W3C standard (For more information: *http://www.w3.org/TR/xforms*) that defines declaratively the contents of a form on a web page, its behavior and its result. It's neither a very thick nor a very complicated standard. However, fully understanding what's going on and several details (like forms submission) can give you headaches!

Exist has two third party XForms processor built-in: betterGorms and XSLTForms. This allows you to create pages that contain XForms logic and have them rendered and executed as the XForms specification describes. To see this in action have a look at the following "Hello World" XForms example:

Example 2-9. A simple XForms example

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:xf="http://www.w3.org/2002/xforms"
  xmlns:ev="http://www.w3.org/2001/xml-events">
  <head>
    <title>Hello XForms</title>
    <!-- The XForms data model: -->
    <xf:model id="xforms-data-model">
      <xf:submission action="HelloXFormsSubmit.xql" id="submit-id" method="post"/>
      <xf:instance xmlns="">
        <Data>
          <Name/>
          <Date/>
        </Data>
      </xf:instance>
      <xf:bind id="NameBind" nodeset="/Data/Name" required="true()" type="xs:string"/>
      <xf:bind id="DateBind" nodeset="/Data/Date" required="true()" type="xs:date"/>
    </xf:model>
  </head>
  <!-- -->
  <body>
    <h1>Hello Xforms</h1>
    <xf:group>
      <xf:input bind="NameBind">
        <xf:label>Name</xf:label>
      </xf:input>
      <xf:input bind="DateBind">
        <xf:label>Date</xf:label>
      </xf:input>
      <xf:submit submission="submit-id">
        <xf:label>Submit</xf:label>
      </xf:submit>
    </xf:group>
  </body>
</html>
```

This example will let you fill in a simple form. Notice that, because we bound the `/Data/Date` field to the data type `xs:date`, the form will automatically show a date picker for the date input field! When you press the submit button (after filling in the values), the posted XML will show through the `hello-xforms-submit.xql` page:

Example 2-10. Getting the results of an XForm

```
xquery version "1.0" encoding "UTF-8";
<XFormsResult>
  {request:get-data()}
</XFormsResult>
```

More about XForms in “XForms” on page 188.