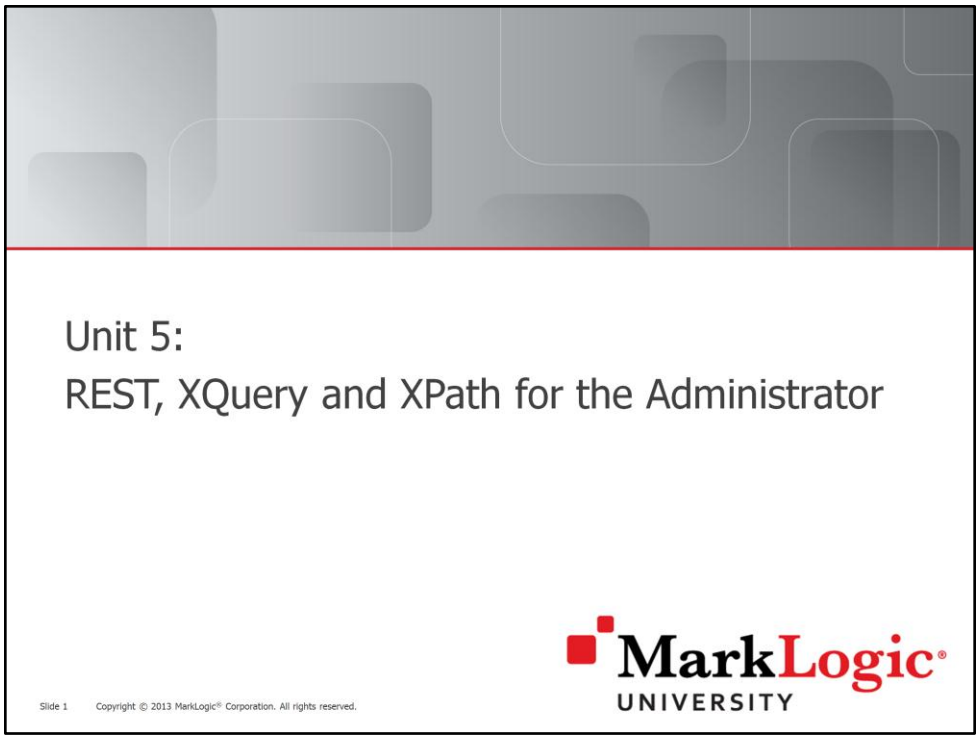
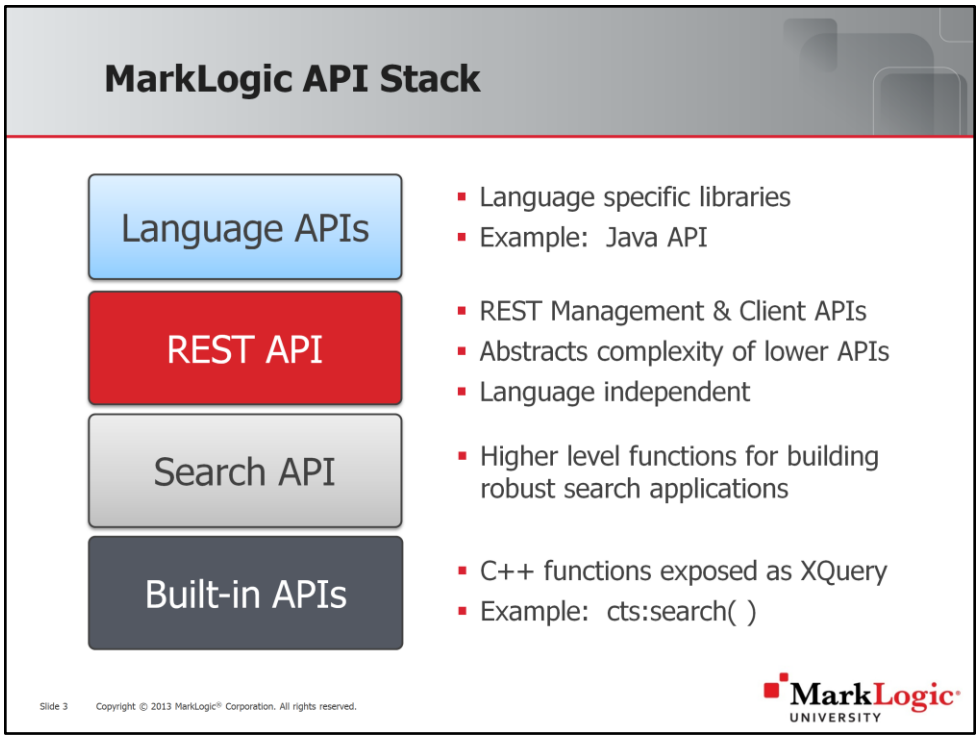


Learning Objectives

- Use the REST API for CRUD
- Describe XQuery, XPath, XSLT and their relationships.
- Describe basic XQuery and XPath syntax.
- Describe namespaces and their use within XPath.
- Write XPath code to navigate documents, work with element and attribute values, implement predicates and utilize axes.
- Write FLWOR expressions.

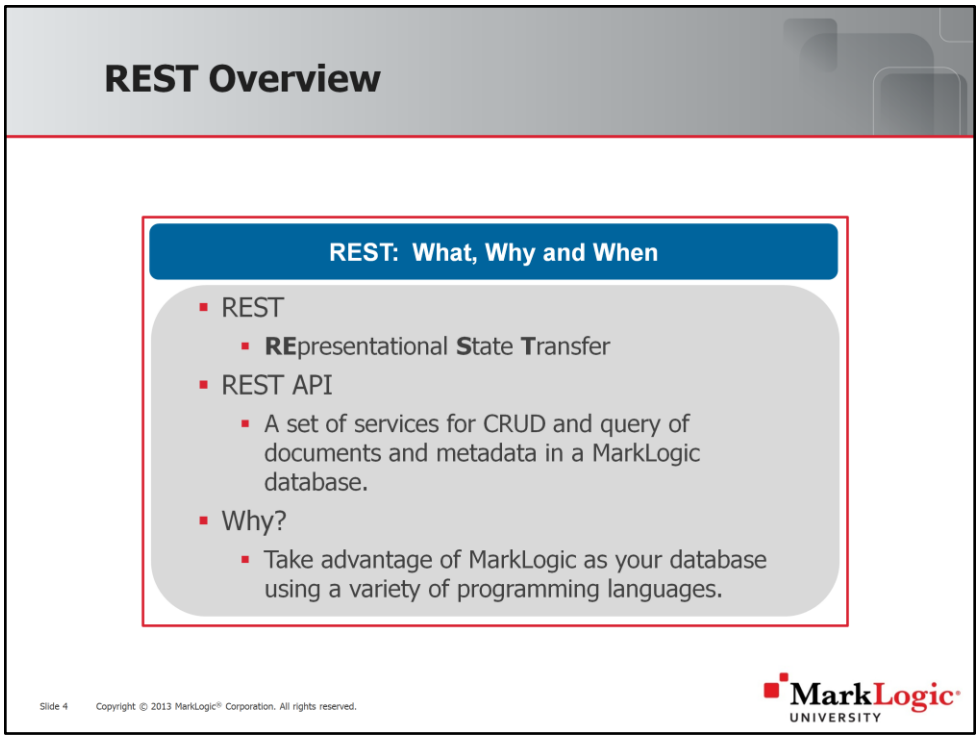
MarkLogic UNIVERSITY



At the base of the API Stack are the Built-in APIs, The Built-in APIs are C++ functions exposed as XQuery. Above the Built-in APIs is the Search API. The Search API pulls all the Built-in APIs together with a powerful interface.

Above the Search API is the REST API. The REST API provides a high-level task-specific interface that abstracts away much of the complexity of the low-level APIs for common use cases: Google-style search, pagination, facets, buckets, search suggestion, snippets, etc. The REST API provides a language-independent interface above this functionality and adds some extra goodies, like document management. This allows developers to do Search API-style things over HTTP, which every language and environment can speak

Above the REST API are the Language APIs. The Language APIs then wrap the HTTP interface to expose an idiomatic interface for that language, for example the Java API that we will focus on in this training class.

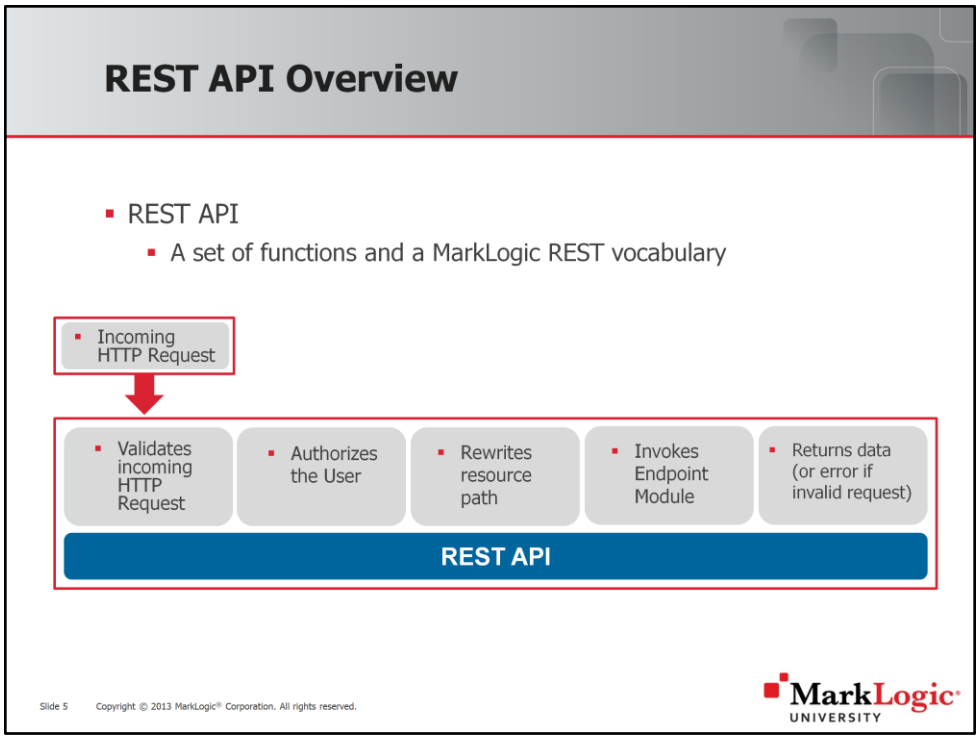


In MarkLogic 6 we've exposed the core MarkLogic functionality as a set of REST API services. What is REST? REST is a lot like the web in the sense that everything you access is a URL. You speak HTTP natively and make good use of HTTP verbs (like PUT or GET for example). Browsers like to say PUT something at this URL or GET me this URL.

By exposing online services and features as URLs, you can view, modify, maybe even delete (if you're on an authenticated client). With REST, every resource (like a document) is a first class entity with it's own URL target space that you can talk to.

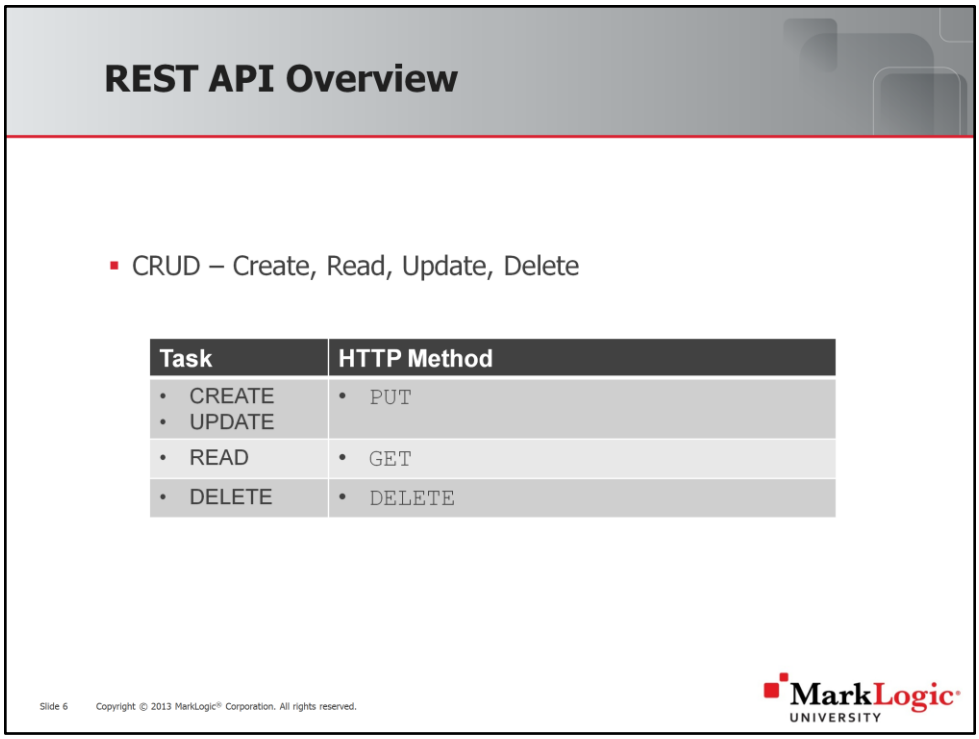
REST stands for Representational State Transfer. REST is an architectural style that, in the context of monitoring MarkLogic Server, describes the use of HTTP to make calls between a monitoring application and monitor host.

The REST API enables developers to easily integrate MarkLogic into existing enterprise IT environments through a common HTTP interface. This ensures that the REST API works well with load balancers, caching proxies and SSL.



The REST API contains a set of XQuery functions and a MarkLogic REST vocabulary of common operations that simplifies the task of describing web service endpoints. These descriptions include the mapping of URL parts to parameters and conditions that must be met in order for the incoming request to be mapped to an endpoint.

The REST API does the following:
- Validates the incoming HTTP request
- Authorizes the user
- Rewrites the resource path to one understood internally by the server before invoking the endpoint module
- If the request is valid, the endpoint module executes the requested operation and returns any data to the application.
- Otherwise, if the request is invalid, the endpoint module returns an error message



The REST API contains a set of XQuery functions and a MarkLogic REST vocabulary of common operations that simplifies the task of describing web service endpoints. These descriptions include the mapping of URL parts to parameters and conditions that must be met in order for the incoming request to be mapped to an endpoint.

The REST API does the following:
- Validates the incoming HTTP request
- Authorizes the user
- Rewrites the resource path to one understood internally by the server before invoking the endpoint module
- If the request is valid, the endpoint module executes the requested operation and returns any data to the application.
- Otherwise, if the request is invalid, the endpoint module returns an error message

REST API Services Examples

- Many more services exist…see the documentation for the inclusive list

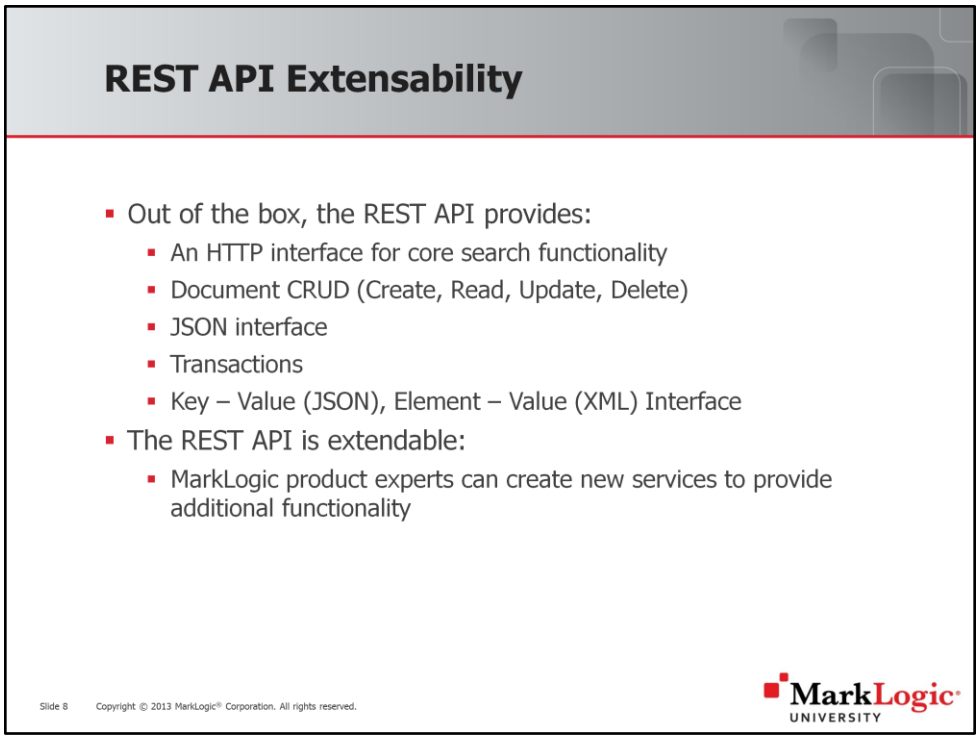
Service Name	Description

/v1/rest-apis	• POST
	  • Create an instance of the MarkLogic REST API, including an HTTP app server, required modules DB, and optionally a content DB.
	• GET
	  • Retrieve a list of REST API instances, including config details.

/v1/documents	• PUT
	  • **Insert or update** document contents and/or metadata, at a caller-supplied **document URI**.
	• GET
	  • **Retrieve** document content and/or metadata from the database.
	• DELETE
	  • **Remove** a document, or reset document metadata.

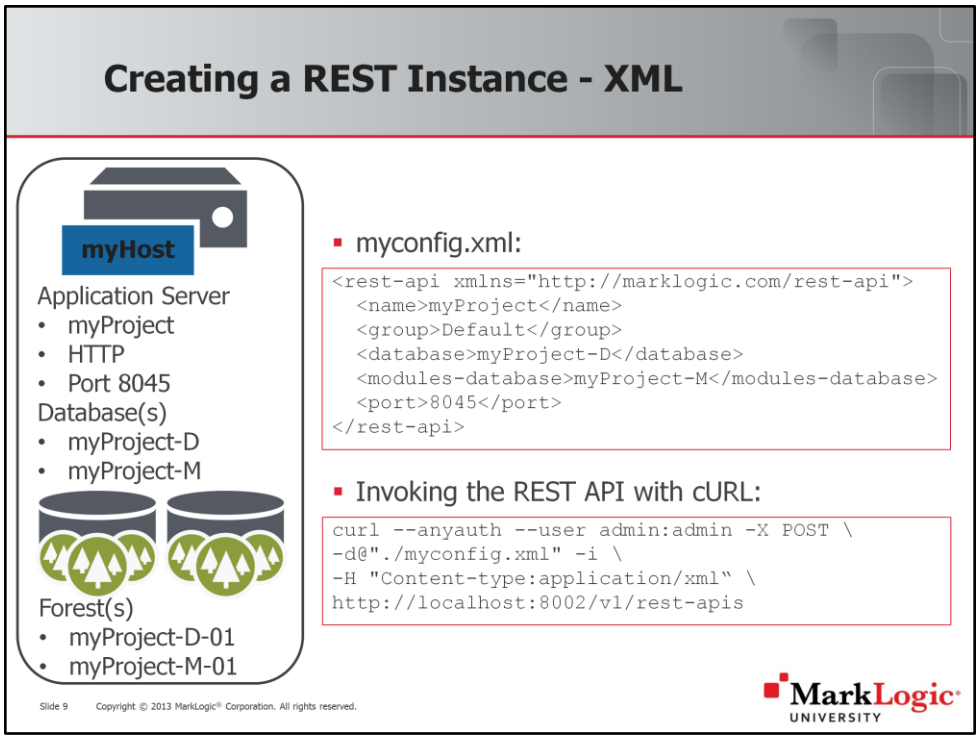
/v1/search	• GET
	  • **Search** the database using a **string and/or structured query**.
	• DELETE
	  • **Remove** documents in a collection or directory, or clear the DB.

MarkLogic UNIVERSITY

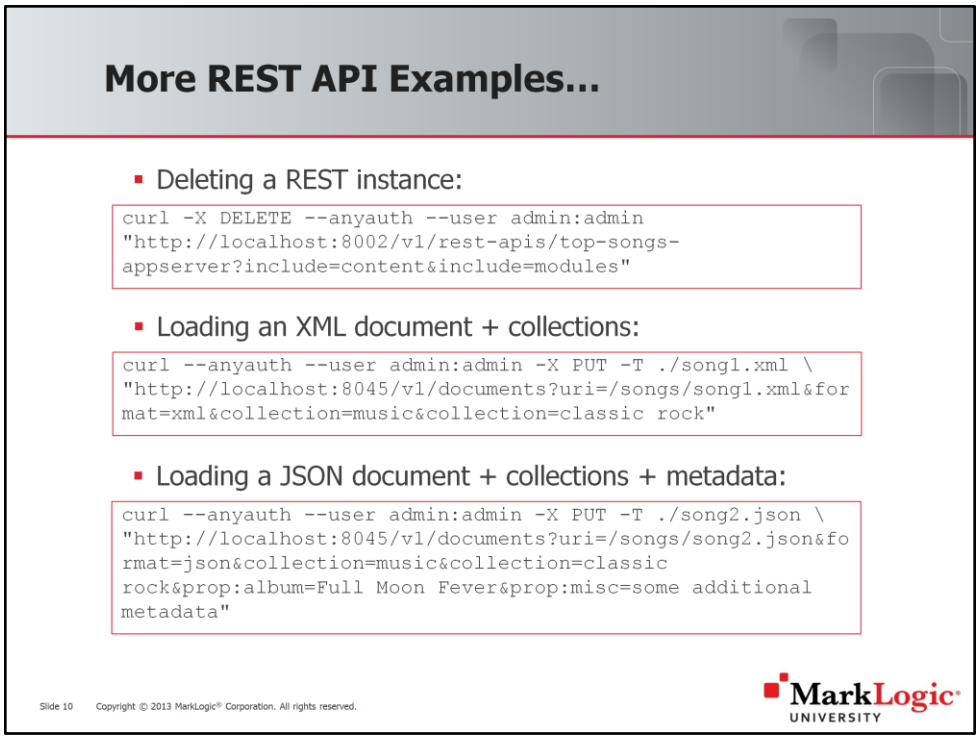


The REST API contains a set of XQuery functions and a MarkLogic REST vocabulary of common operations that simplifies the task of describing web service endpoints. These descriptions include the mapping of URL parts to parameters and conditions that must be met in order for the incoming request to be mapped to an endpoint.

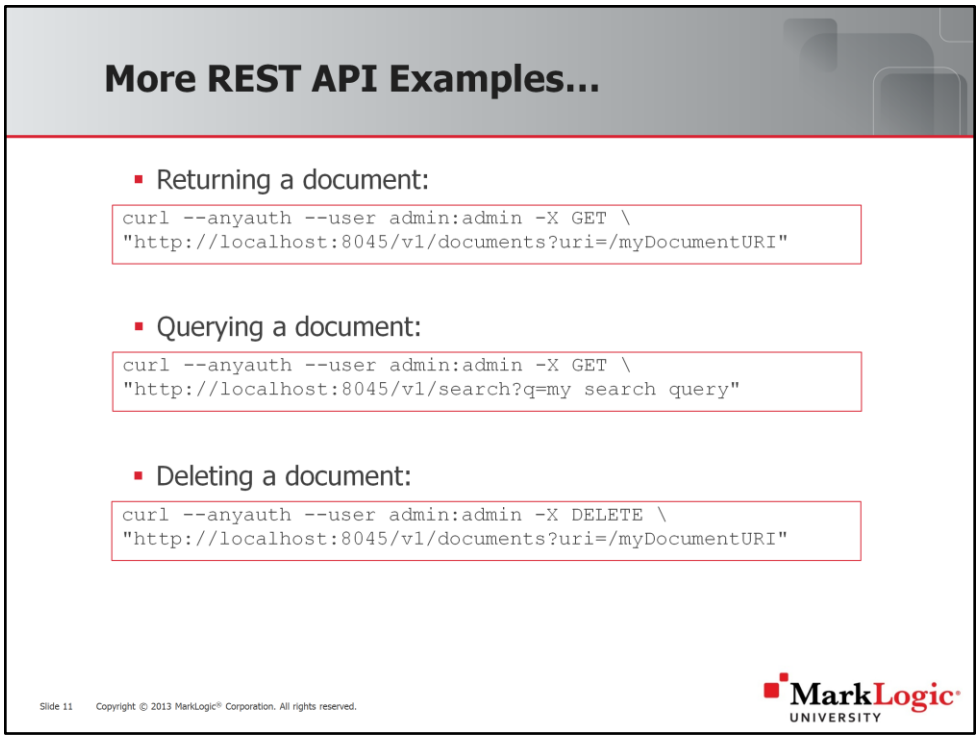
The REST API does the following:
- Validates the incoming HTTP request
- Authorizes the user
- Rewrites the resource path to one understood internally by the server before invoking the endpoint module
- If the request is valid, the endpoint module executes the requested operation and returns any data to the application.
- Otherwise, if the request is invalid, the endpoint module returns an error message



You can use the REST API to deploy an HTTP Application Server (configured to use the REST API for your applications) as well as a database to contain your data and a second database to contain your code and REST API configurations.



- Deleting a REST instance uses the /v1/rest-apis/ service should be provided the application server name.
  - You have the option to delete the content and modules database as well as shown in the example.
- Loading a document (CREATE) uses the /v1/documents/ service and requires that you specify a document URI.
  - If the document URI already exists in the database, it becomes an UPDATE transaction.
  - Optional parameters include security permissions on the document, collections, metadata, etc. For full description of available optional parameters, please reference the product documentation.



- Returning a document (READ) and deleting a document (DELETE) are both accomplished using the /v1/documents/ service and providing it a document URI.
- You can query a database using the /v1/search/ service and providing it a search query. Under the covers the /v1/search/ service utilizes the MarkLogic search API. This means it automatically understands search grammar, for example:
  - term1 OR term2
  - “look for this phrase” AND term1
  - Etc.

What is XQuery?

- Functional vs. Object Oriented vs. Procedural
- W3C Specification
  - http://www.w3.org/XML/Query/
- Provides logic for working with XML data
  - FLWOR
- *.xqy
- MarkLogic libraries provide added functionality.

Slide 12

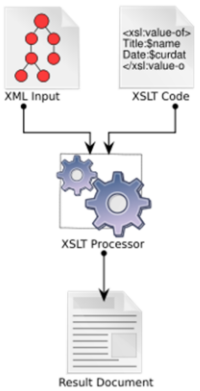
What is XPath?

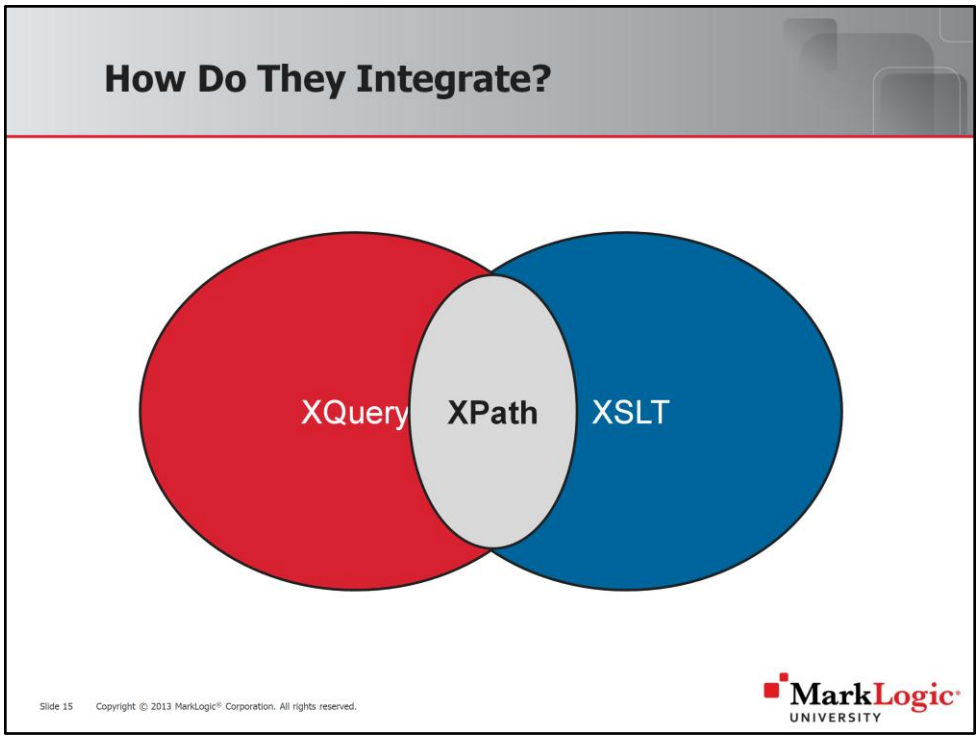
- Provides the ability to navigate and extract data
- Visualize XML as a tree structure

```
<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

</bookstore>
```

What is XSLT?

- Designed for transforming XML documents.
- Example:
  - Creating a new document based on content from an existing XML document.
  - The new document can be XML, or some other type (PDF, HTML, etc.)

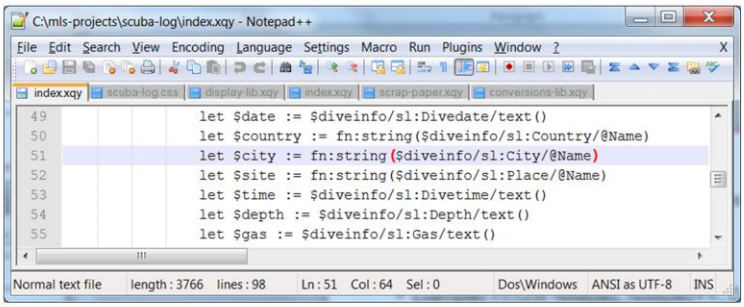




- XQuery and XSLT are both Turing Complete programming languages, meaning either language could be used to compute whatever is needed.
- XQuery and XSLT both share the same data model (XML) and share Xpath.
- With MarkLogic, using XQuery provides the version 1.0-ml dialect offering additional functionality for working with MarkLogic server.
    - Example: xdmp functions, try-catch error handling logic, etc.
- In practice, XSLT is generally used for transformations, although transformations could be done well in XQuery as well. As a matter of style, some programmers prefer to use XSLT for transformations.

XQuery Editors

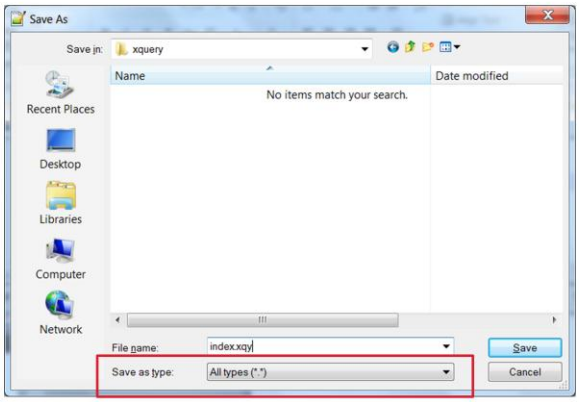
- Development Environment, may contain features like:
  - Automatic </end tag> placement
  - Color coding, parentheses and brace matching
- Examples include Notepad, Notepad++, Oxygen, XQDT



Once we have content loaded in a MarkLogic database, we can start to write XQuery code and build an application on top of that database. To write XQuery code, there are a host of tools available.

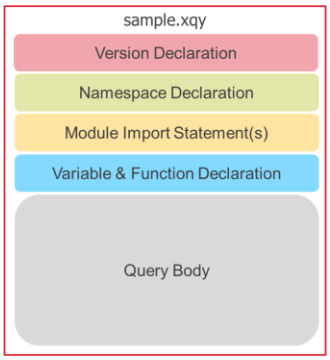
XQuery Editors

- Watch out for:
  - Filename.xqy.txt
- Avoid by selecting "All Types"



Anatomy of an XQuery File

```
xquery version "1.0-ml";

declare namespace wl =
"http://marklogic.com/mlu/world-leaders";

import module namespace co =
"http://marklogic.com/mlu/worldleaders/
common" at "modules/common-lib.xqy";

declare variable $leaders := /wl:leader;

xdmp:set-response-content-type("text/html;
charset=utf-8"),
'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
1.0 Strict//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">',
<html
xmlns="http://www.w3.org/1999/xhtml">
...
```



MarkLogic UNIVERSITY

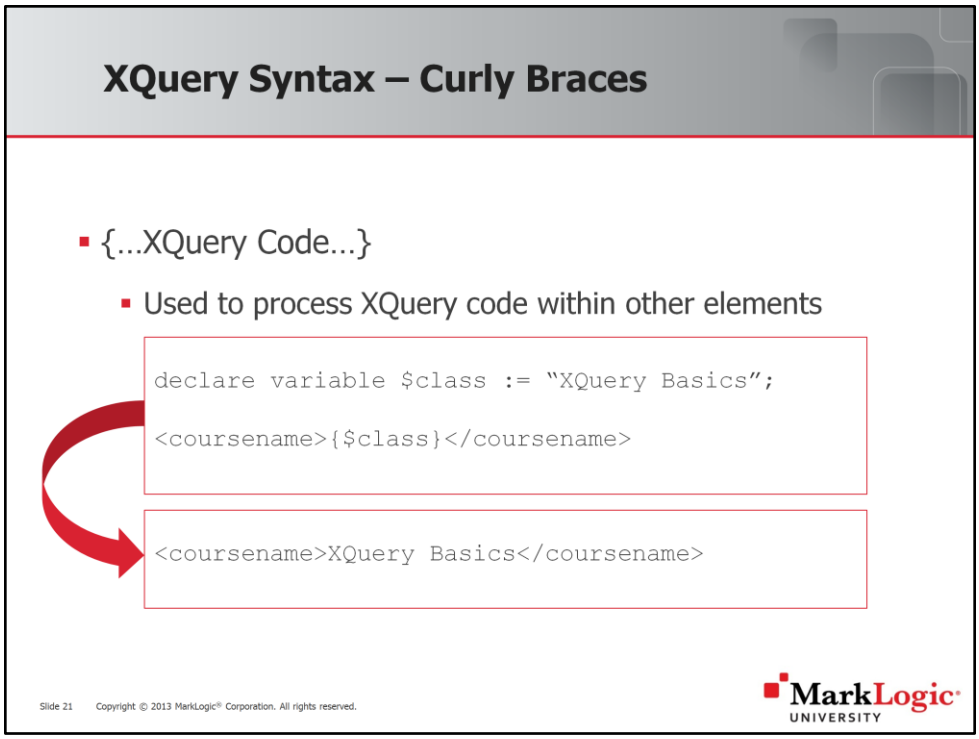
XQuery Syntax - Rules

- Semicolons after…
  - Version declaration
  - Namespace declaration
  - Import statements
  - Variable declarations
  - Function declarations
- No semicolons within FLWOR expressions
- Example:
  - `xquery version "1.0-ml";`

XQuery Syntax - Rules

- Variable Declaration & Assignment
  - `declare variable $class := "XQuery Basics";`
  - `declare variable $class as xs:string := "XQuery Basics";`
- If type is not specified it will be assigned automatically.

MarkLogic UNIVERSITY



XQuery Syntax - Sequences

- Everything in XQuery is returned as a sequence
- Can be empty, one or many items.
- Defined with parentheses and commas

```
declare variable $avg := (10,20);

<average>{fn:avg($avg)}</average>
```

```
<average>15</average>
```


MarkLogic UNIVERSITY

XQuery Syntax - Comments

```
(: show the average of a sequence of numbers :)

declare variable $avg := (10,20);

<average>{fn:avg($avg)}</average>

(:  comments can also span multiple lines with
only a singe start and end tag  :)
```

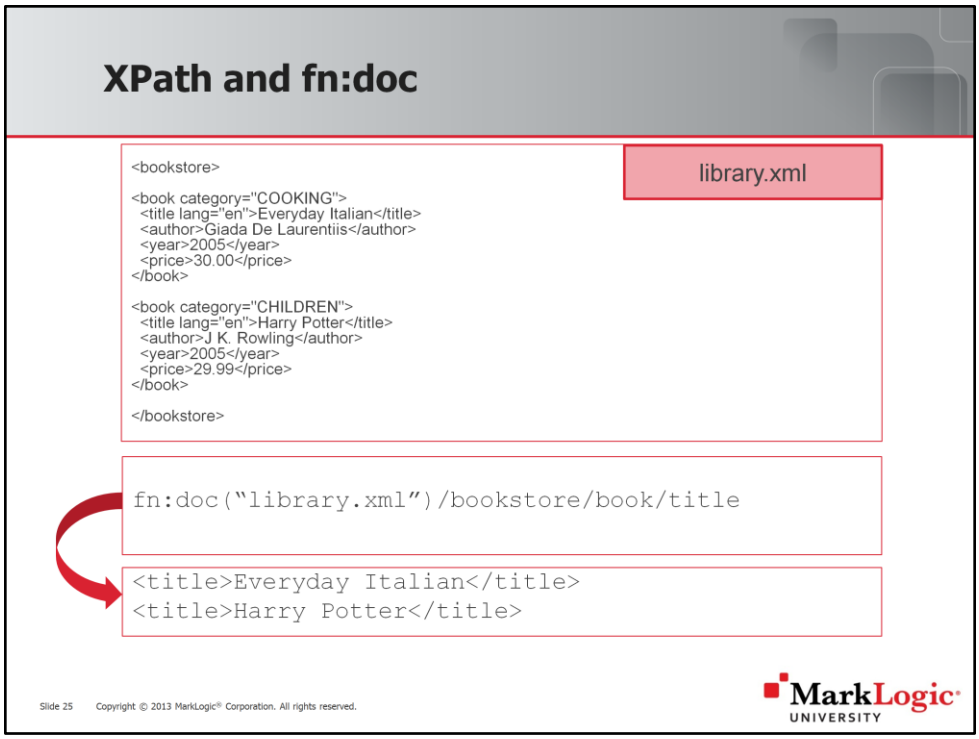
XPath & XML Tree Structure

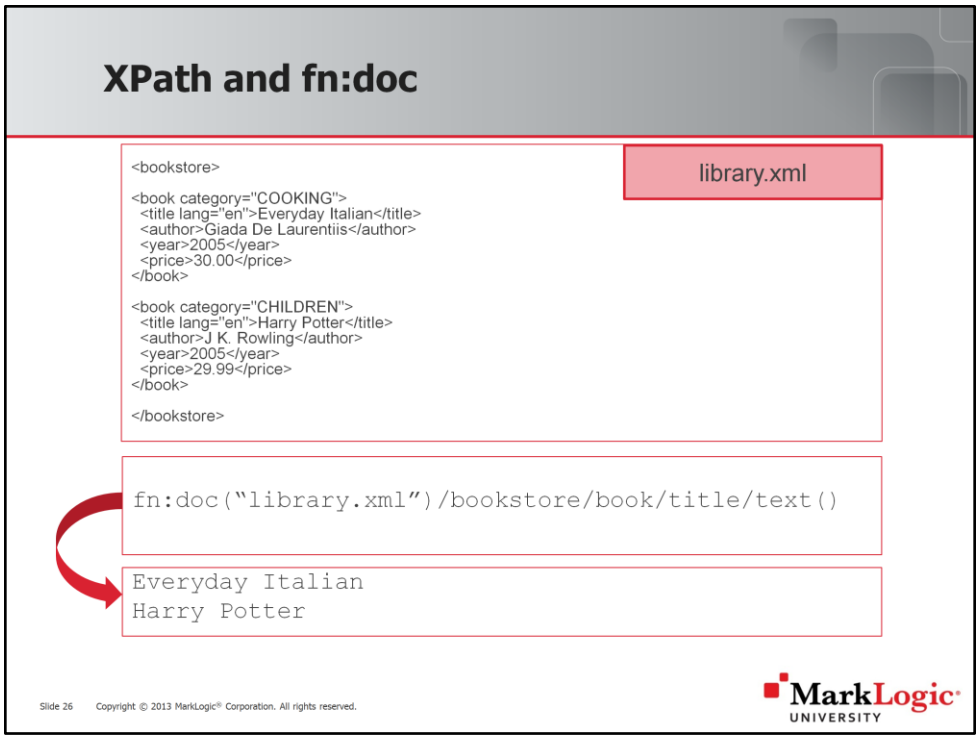
library.xml

```
<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

</bookstore>
```

- Root element node <bookstore>
  - Element node <book>
    - Element node <title>
      - Text node "Everyday Italian"
    - Element node <author>
    - Etc...

Slide 24





XML Namespace

- Element and Attribute nodes are always in a namespace.
  - If none defined, they are in the "empty" namespace
  - Namespace has a URI associated with it.
  - URI does not need to actually exist
- Namespaces are defined inside an element:

```
<bookstore xmlns="http://www.marklogic.com/bookstore">
```

XML Namespace

- To reference any elements/attributes of a document with a namespace in XQuery code you must:
  - Declare the namespace
    - Note the required semicolon
  - Reference the namespace in XPath code

```
declare namespace bks="http://www.marklogic.com/bookstore";

/bks:bookstore/bks:book/bks:title/text()
```


MarkLogic UNIVERSITY

Why Use a Namespace?

- A namespace is useful as a logical differentiator
  - Identical element names in the same database
  - Versioning
- Example:
  - Our online bookstore sells both books and DVDs

```
<bookstore xmlns="abc">
  <book>
    <title>
```

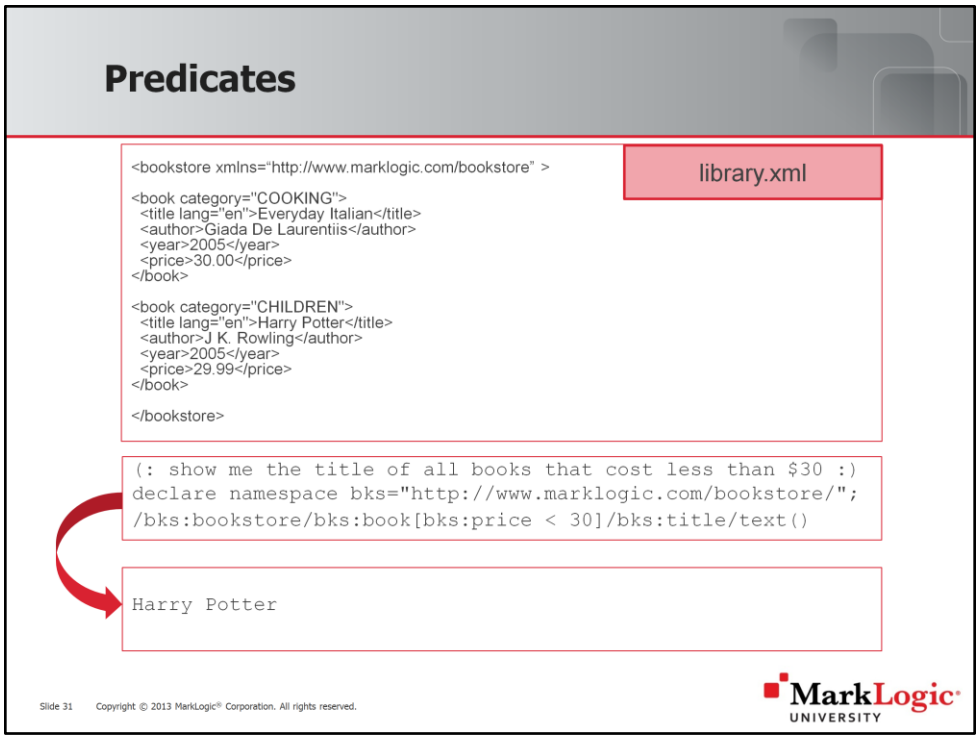
```
<bookstore xmlns="xyz">
  <dvd>
    <title>
```

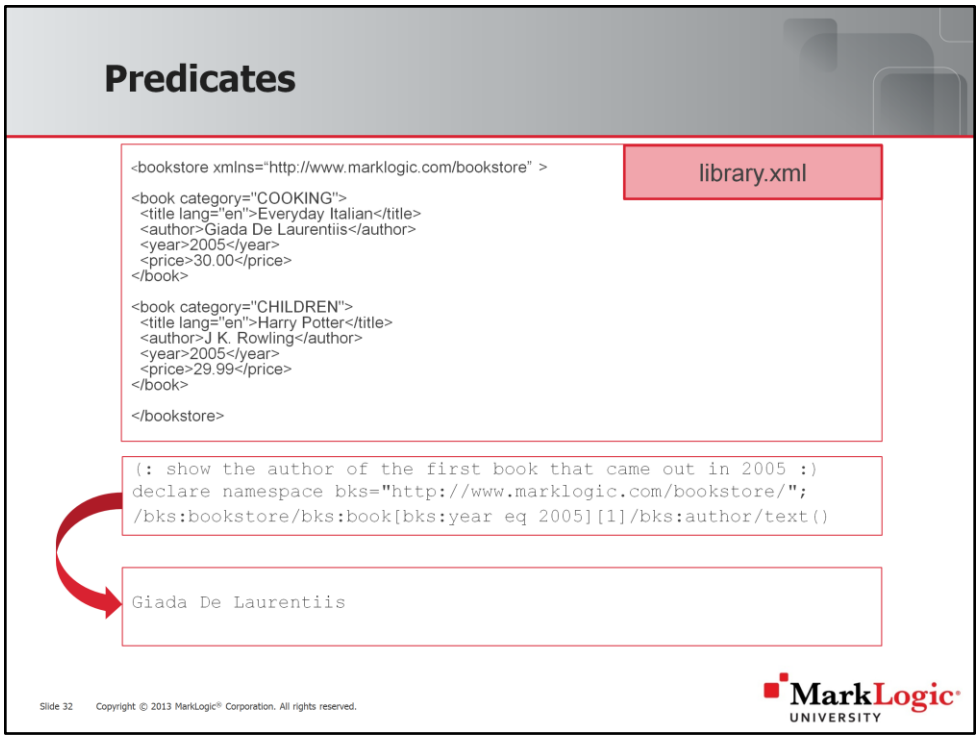
Slide 29

Predicates

- Similar to WHERE clause in SQL
- Syntax:
  - /element[predicate]/
- Operators include:
  - eq (equals), <, >, <>, ge (greater than or equal to), le (less than or equal to), and, or, position

MarkLogic UNIVERSITY

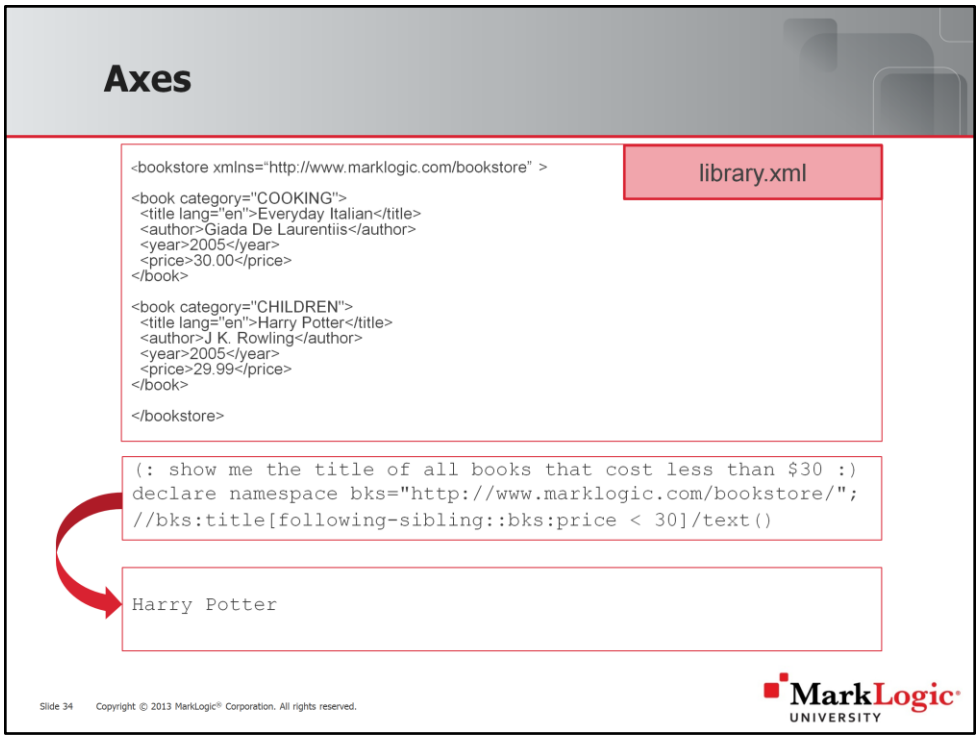




Axes

- Axes provide options for navigating through the tree
  - Up / down (Y axis)
  - Left / right (X axis)
- Chart in upcoming lab exercise outlines many navigation options and shortcuts

MarkLogic UNIVERSITY

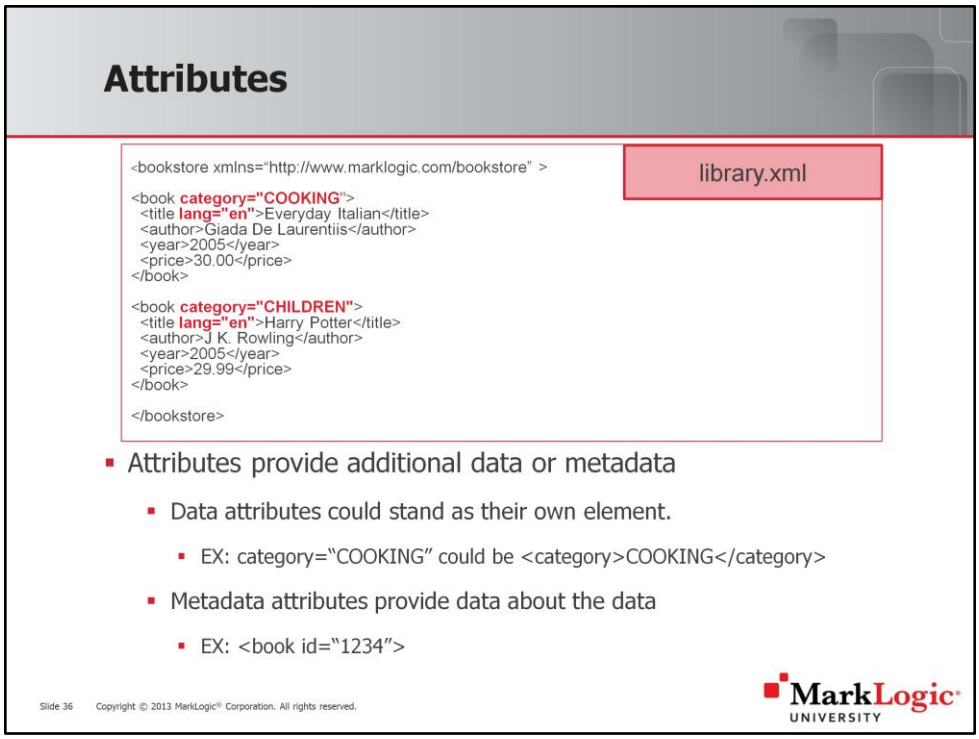


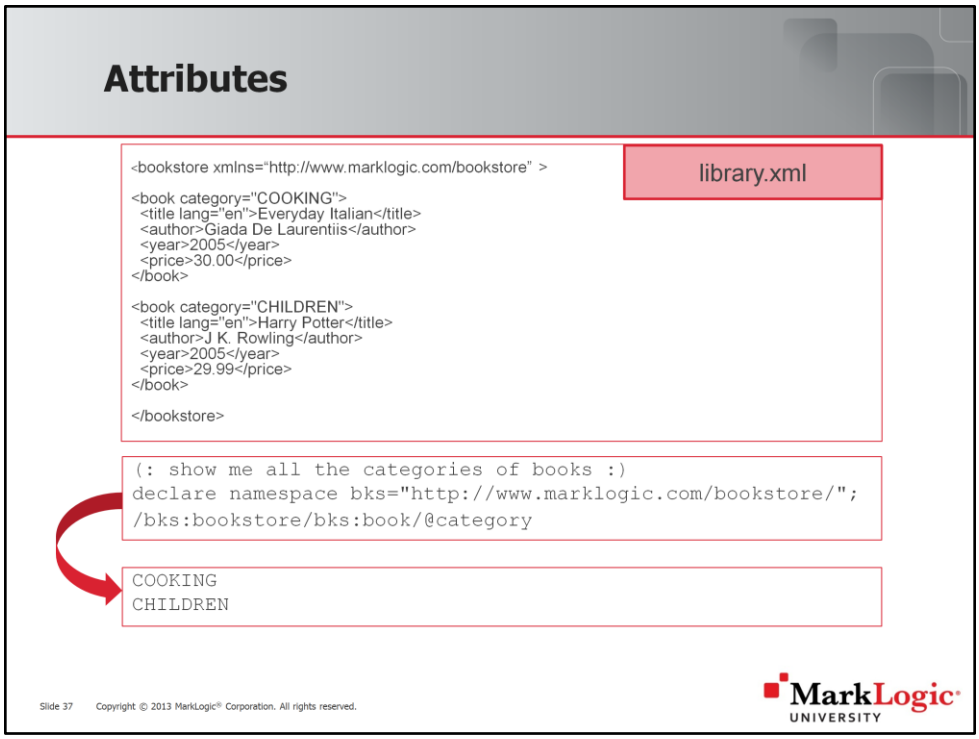
Why Use Axes?

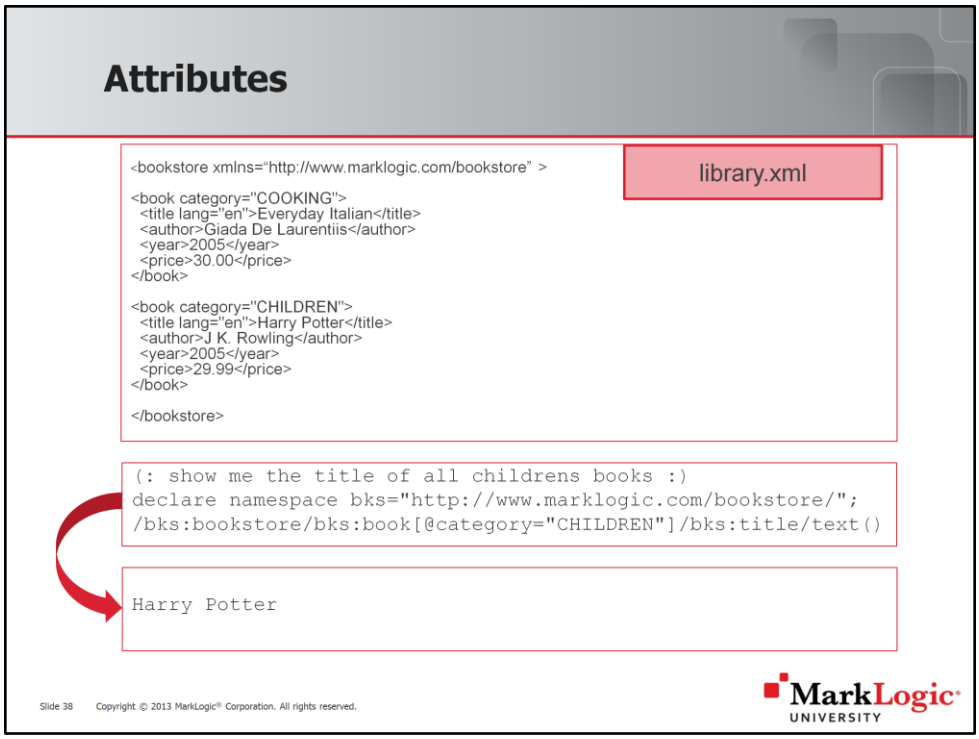
- Axes allow more generic references in XPath code
  - Example of bookstore selling both books and DVDs
    - Show all titles using XPath with no axes
      - `/bookstore/book/title/text()`
      - `/bookstore/dvd/title/text()`
    - Show all titles using XPath and axes
      - `//*:title/text()`

```
<bookstore xmlns="abc">
  <book>
    <title>
```

```
<bookstore xmlns="xyz">
  <dvd>
    <title>
```







FLWOR

- For
- Let
- Where
- Order by
- Return

FLWOR Definition

- FOR : creates a sequence of tuples
  - Tuple is an ordered list (sequence) of elements
  - Multiple sequences can be established for joins
- LET : binds a sequence to a variable
- WHERE : filters according to a constraint.
- ORDER BY : sorting criteria
- RETURN : gets evaluated once for every tuple
  - When returning more than a single expression, put the expressions in a sequence: return (a, b, c)

MarkLogic UNIVERSITY

FLWOR Requirements

- At least 1 **FOR** clause **or** at least 1 **LET** clause
  - Can be multiple of each
  - Can be in any order
- 1 **RETURN** clause
- **WHERE** and **ORDER BY** are optional

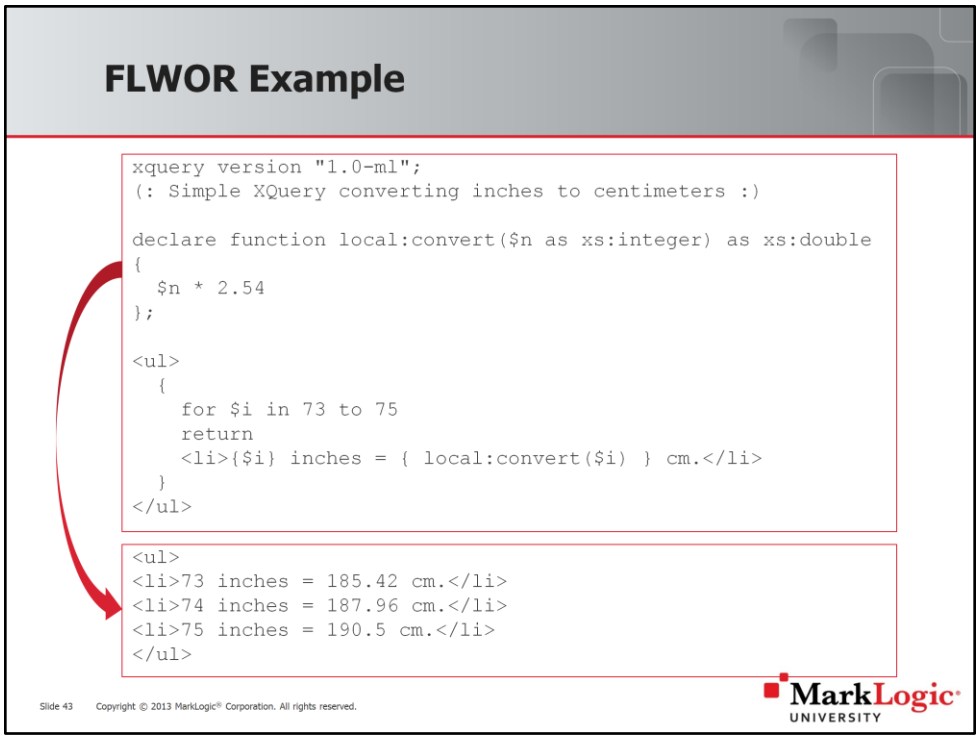
VALID	VALID	INVALID

`FOR $i in (1,2)`
`RETURN $i`
	`LET $i := "HI"`
	`RETURN $i`
		`Declare variable`
		`$x := "HI";`
		
		`RETURN $x`

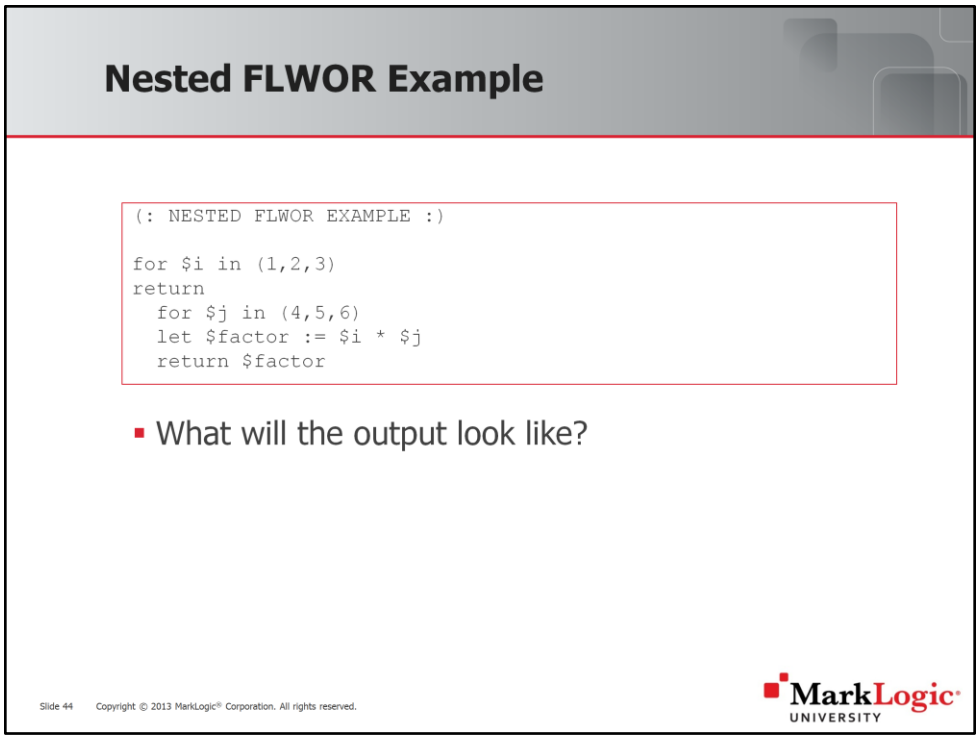
FLWOR Example

```
(: return numbers in a sequence that when squared are > 25 :)

for $i in (1,2,3,4,5,6,7,8)
let $squared := $i * $i
where $squared >= 25
order by $i descending
return $i
```

```
8
7
6
5
```



- Notice the local function declaration
- Notice the nesting in the FLWOR statement



FLWOR statement, like all expressions in XQuery are nestable. It is perfectly acceptable for the return statement of one FLWOR to contain another FLWOR statement. Take a minute to study the nested FLWOR example shown here.

How would this statement be evaluated? What would the resulting output look like?

Nested FLWOR Example

```
(: NESTED FLWOR EXAMPLE :)

for $i in (1,2,3)
return
  for $j in (4,5,6)
  let $factor := $i * $j
  return $factor
```

```
4
5
6
8
10
12
12
15
18
```

Unit 5: Applying the Learning Objectives

- Use the REST API for CRUD
  - Exercise 1 | Exercise 2 | Exercise 3 | Exercise 4
- Describe XQuery, XPath, XSLT and their relationships.
- Describe basic XQuery and XPath syntax.
- Describe namespaces and their use within XPath.
- Write XPath code to navigate documents, work with element and attribute values, implement predicates and utilize axes.
  - Exercise 5 | Exercise 6
- Write FLWOR expressions.
  - Exercise 7

Slide 46

MarkLogic UNIVERSITY