summer school

Publishing with XML

17th September 2013

XML Web Applications

Adam Retter

adam.retter@googlemail.com

@adamretter

My/Your Approach

To share knowledge

and explore interesting questions

1. **Interrupt** me...

    1. Ask questions

    2. Share experience, related and relevant information (with all)

2. Direction of the class, is influenced by **you**!

...This *should NOT* be a lecture!

Learning Objectives

1. Explore what is meant by "XML Web Application"
2. Learn about options for XML Web Applications
3. Understand how XML Web Applications assist Publishing
4. Explore the nuts-and-bolts of a Publishing XML Web Application

Contents

A plan, from which we *may* digress:

Talk:

1. Defining 'XML Web Application'
2. The Web, Publishing and Applications
3. Example Publishing Pipeline
4. XML Web Application Architectures
5. Working with the Web from XQuery

Examples and Demos:

6. Document Submission
7. Editorial
8. Assembly

summer school

Defining “XML Web Application”

mostly questions

What is an 'XML Web Application'?

- But first – What is a 'Web Application'?

  "*any application that uses a web browser as a client*"

  Daniel Nations / about.com

- However:
  - Client/Server architecture
  - Dynamic Processing
  - Code may run on Server and/or Client

What is an 'XML Web Application'?

It could be -

- XML applied to a Web Application?
  - Ship your Web App inside an XML file?
  - Configure your Web App using XML files?
  - Code your Web App using XML?

- A Web Application applied to XML?
  - Serve XML Documents?

What is an 'XML Web Application'?

- *Google* 'XML Web Application' offers little -
  - "**XML and Web Applications**" - Brian Stafford
    - Promotes the use of XML for building Web Apps
    - Reducing Client-Server interaction by using XML
    - Laments browser support (for XForms in particular)
  - Many links about "**web.xml**" files
    - Java EE Servlet Deployment Descriptor

What is an 'XML Web Application'?

- "**XML in HTML**" - w3schools.com
  - Creating HTML Elements from XML with JavaScript
    - Parse XML with JavaScript
    - Modify HTML DOM using JavaScript
- "**Practical Web Applications**" - XMLSS 2012
  - A Web Application (Ruby) applied to XML
- "**Web Applications and XML Technologies**" - Adam Retter
  - Zero-Translation, i.e. both approaches together:
    - XML applied to Web Applications
    - Web Applications applied to XML

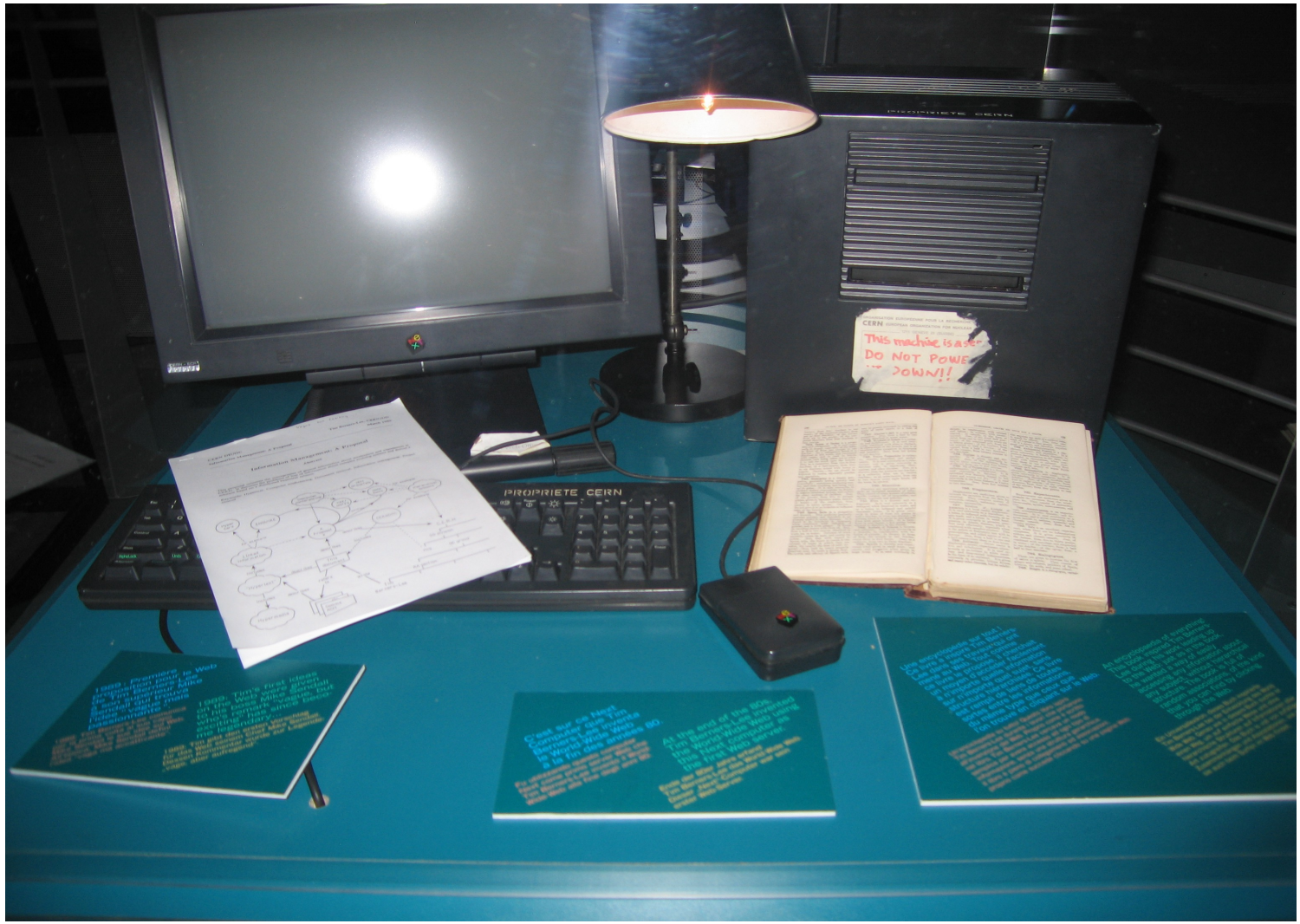
summer school

The Web, Publishing and Applications

a brief historical diversion

The first Web Server (1990)

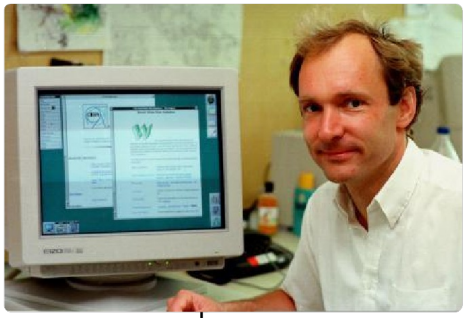
summer school



Picture from Simon Bisson - Flickr CC-by-nc-nd

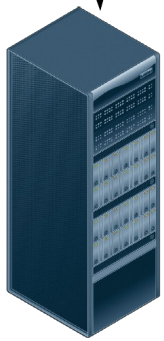
"This machine is a server.
DO NOT POWER IT DOWN!!"

Web Publishing 1990's



Author HTML
Prepare gif/jpg images

Upload using FTP



Home User

HTTP Download

Server for
www.something.com

Dial-up Modem

Web Publishing 1990's

- Getting online is hard and expensive!
  - Computer
  - CHAP/PAP, TCP/IP and WebBrowser Software
  - Telephone Line + Rental
  - Modem
  - ISP Dial-up account

- Publishing online is even Harder!
  - Need to know HTML.
  - FTP Software
  - Web Space (Rental/Free?)
  - Understand - Domain Name Reg. and DNS

Early Websites looked:

The World's Worst Website

Gratuitous use of frames is a common mistake of web designers.

Many older browsers do not support frames. They disrupt the flow of the website and can be difficult to anticipate where a page may appear when a link is clicked. Click here for an example of a frames page which is opening in the wrong window. Use your browser's 'Back' button to escape.

If you must use frames, use the tag <base target="_blank"> between <head> and </head> to assure links will open in a new window.

Check out these links to websites whose opinions about frames is self evident:

The "I Hate Frames" Frames Page

Another I Hate Frames Page

The International I Hate Frames Club

Why Frames Suck (Most of the Time)

Welcome To My Website!

Welcome to the World's Worst Website!

This web was designed to graphically demonstrate the most common mistakes made by new Web Page designers.

Where am I and where are the links to other pages?

An easy to use navigation structure is essential to any well designed website! Important information should never be more than 2 clicks away.

As you can see, this text is difficult to read. There needs to more contrast between the background color and the text color. Here's another example of a poor choice of a background/ text color and size.

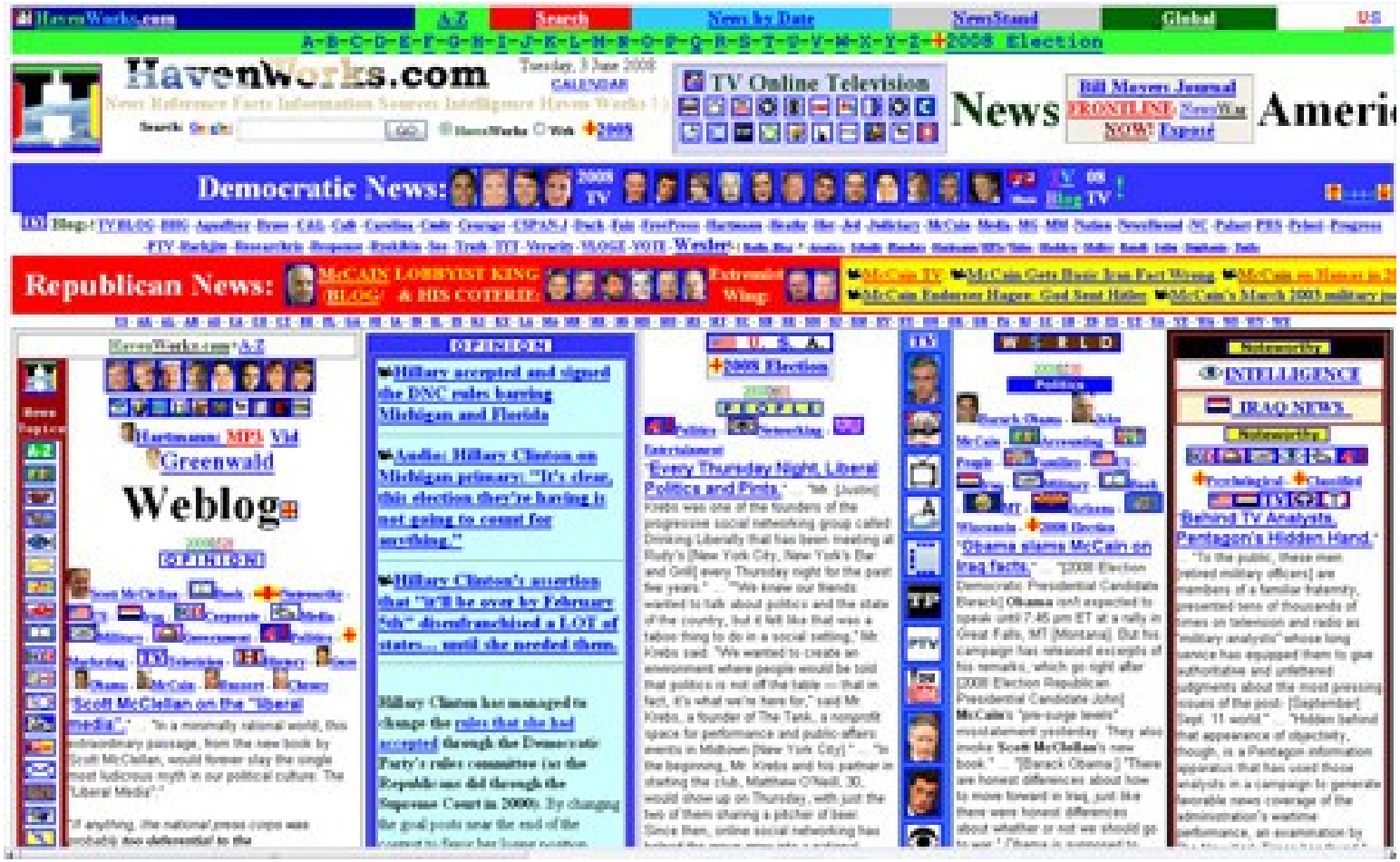
Keep your backgrounds simple. White or light colors usually work best. Your background should not compete with the content of the page for the users attention. If you would like to use a background picture, select a picture that uses muted colors or format your picture as a watermark. Select text colors which will contrast well with the background picture.

Constantly running animations can be distracting when used excessively. There should be no more than one animated object in your view at any time. Also in this category are excessive, large, flashing & obnoxious advertisements.

Excessive advertisements: Aggressive pop ups, banners, flash ads and other intrusive ads annoy your visitors, make your site difficult to use and bury your message in a sea of clutter.

summer school

Early Web Publishing



Web Publishing <1996

- Majority of Web Site Content is Static
- Lack of Quality/Beauty
  - Web Formatting and Publishing tech. Immature!
  - Block content only. Frames and/or Tables.
- Almost all content appears original
  - Content is Hyper-Linked, instead of Syndicated
  - Some copy/paste. Few High Quality outlets
- 1995: nytimes.com, msn.com
- 1997 – news.bbc.co.uk

Early Web Publishing 1994 - 1998

summer school

1994

1996

1995

1998

Let's get Nostalgic!

Probably the best thing *published* on the Web in 1998:

Web Publishing <1996

- However, it was not all bad -
  - Little *evil* commercial interest
    - Spam, Phishing etc. have yet to take off
    - Domain squatting is non-existent
  - Few worried about security
  - Little in the way of advertising and marketing
  - SEO has not yet been invented
  - Social Media has not yet been invented
  - You can still remember most web-addresses you need ;-)

Beyond Static Content

- Content could be generated by CGI-BIN (>1993)
  - Difficult for programmers
  - Standard?
  - Where to store state?
  - Generated Content is itself Static
- Simple generation of HTML and possible Images
  - Web Counter
  - Form Mail
  - Guestbook

The first Facebook?

Yes! GPA!

(Credit: Angelfire Screenshot by Chris Matyszczyk/CNET)

Non-static Web Publishing

- Rise in Server-Side Scripting + Database Support
  - 1995 PHP 1.0 (mSQL)
  - 1996 Microsoft ASP 1.0 (ODBC)
  - 1999 Sun JSP 1.0 (JDBC)

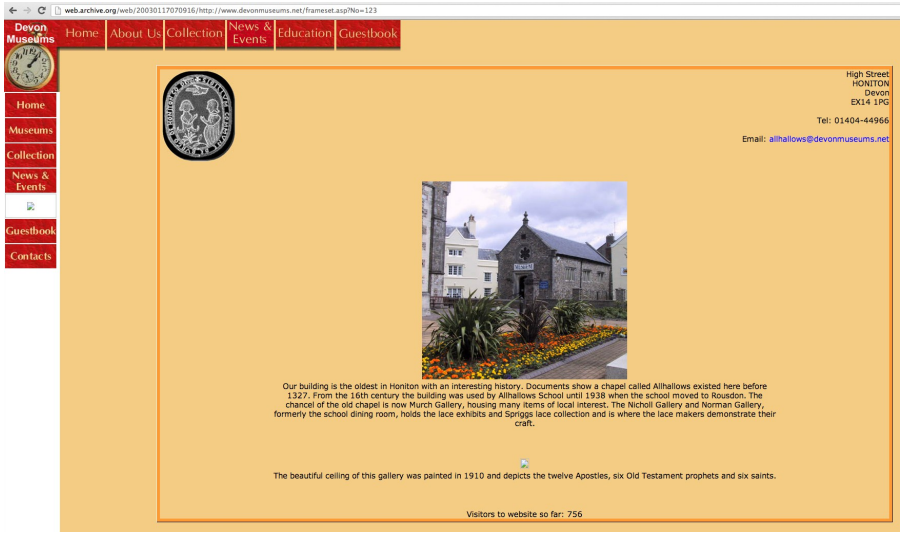
- Simple generation of Web Pages **with Database content**

- User Activity and Form Responses
  - Could also be persisted to the database
    - ***Effect generation of subsequent pages***

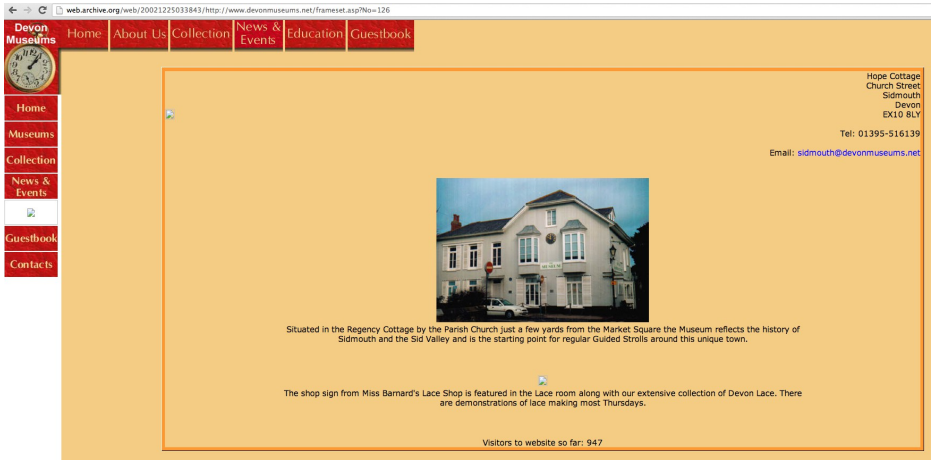
Template Driven Web Publishing

- Same page different *bits* of content
  - Template Pages + Relational Database
    - Glued together with Server Side Scripting

- e.g. 1999 - devonmuseums.net:



Allhallows Museum



Sidmouth Museum

summer school

Problems of Template Driven Web Publishing

- *Impedance Mismatch*
  - Web is Document oriented
  - Relational Database is Key/Value Oriented
    - Deciding how to break page content apart?
    - Modelling a Document in as Key/Values in RDBMS?
    - Simple Templates... complex content = complex template!
- Templates of Templates (Hierarchies)
  - Feature of many CMS
  - Ultimately still RDBMS backed
  - Hard to understand how page is constructed

Transformation Driven Web Publishing

- Input → Transform → HTML

- Initially they still tried to do it from RDMS!
  - Create HTML with Proprietary Template bits
  - Write some complex SQL queries
  - Apply Transformation to each query result-set
  - Ultimately the result is a folder of HTML files

- Then XSLT 1.0 came along 1998/1999
  - Content *is* a Document. Transformed to HTML.
  - Problem of single document input/output.

Today... We have come a long way!

- Standards for Documents, Transformations, Query and Presentation.
  - XML 1.1, XSLT 2.0, XQuery 3.0, HTML5, CSS 3
- Content Generation is still mostly Server-Side
- Dynamic Interaction is mostly JavaScript (Authoring?)
- ***Publishing now rarely involves FTP'ing static HTML code***

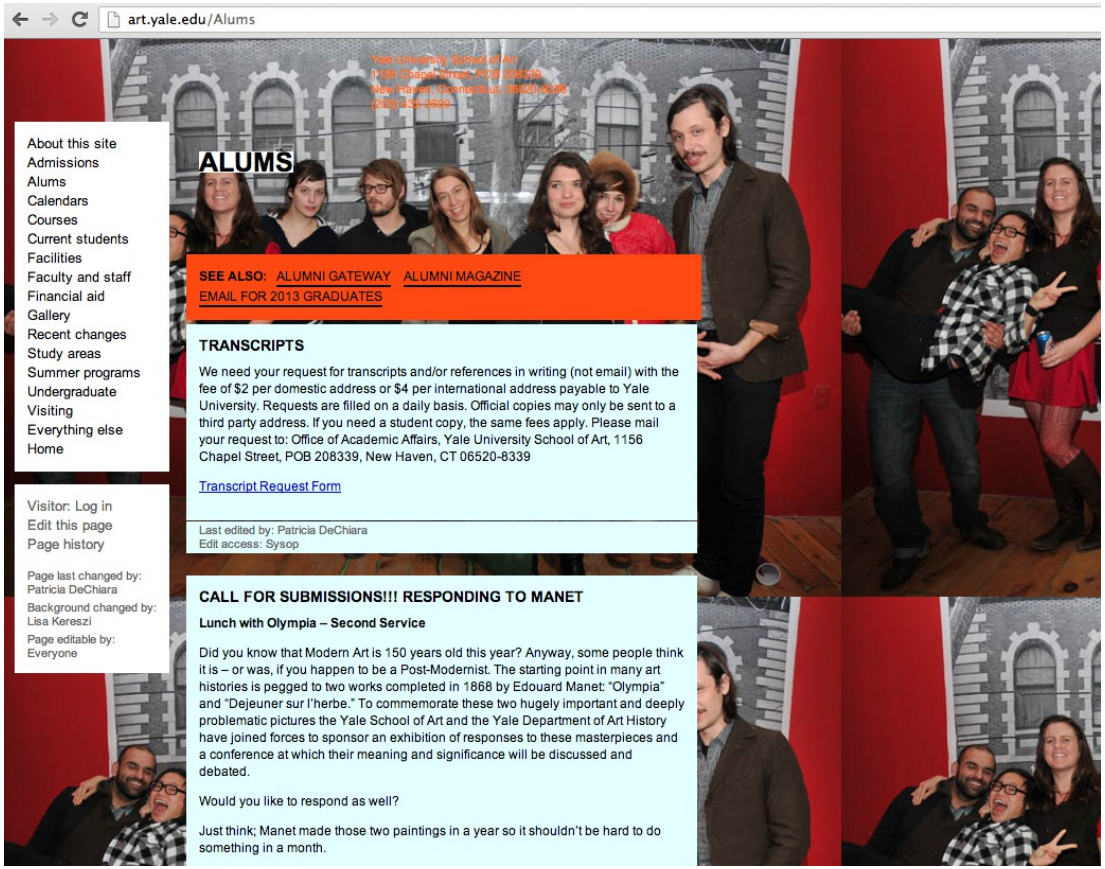
Getting here...

summer school

Year	Web Technology	XML Technology
1991	HTML Tags, CGI-BIN 1.0	
1993	HTML Draft	
1995	HTML 2.0 (<form/>), PHP 1.0	
1996	CSS 1.0, ASP 1.0, JavaScript 1.0	**XML 1.0 (D)**
1997	HTML 3.2 (<script/>), HTML 4.0, JavaScript 1.2	
1998	CSS 2	XML 1.0 (R), XSLT 1.0 (D), XHTML 1.0 (D)
1999	HTML 4.01, JSP 1.0	XPath 1.0, XSLT 1.0 (R), XHTML 1.1 (D)
2000		**XForms 1.0 (D)**
2001		**XQuery 1.0 (D), XPath 2.0 (D)**
2002	ASP.net 1.0	XML 1.1 (D)
2003		XForms 1.0 (R)
2004	CGI-BIN 1.1 (RFC 3875)	XML 1.1 (R), XHTML 1.0 (R)
2007	HTML 5 (D)	XQuery 1.0 (R), XPath 2.0 (R), XQuery 3.0 (D)
2009		XForms 1.1 (R)
2011	CSS 2.1	

Or have we?

- Deserves a special mention!
  - Yale University School of Art



Or have we?

- Deserves a special mention!
  - Yale University School of Art

summer school

Defining “XML Web Application”

answering the question

Classic Web Application Applied to XML

- Static
  - Web Server + File System
  - Just serves up the Raw XML
- Dynamic
  - Server Side Code (e.g. Python/Perl/PHP/Java)
  - Transformation of XML documents
  - Authoring of XML documents
  - Where to keep your XML?
    - File System
    - RDBMS

XML Applied to a Web Application

- Static
  - Document Management System containing XML
    - May offer some dynamic facilities!

- Dynamic
  - Server Side Code (e.g. XSLT, XQuery, XProc, XForms?)
  - Transformation of XML documents
  - Authoring of XML documents
  - Where to keep your XML?
    - File System
    - Native XML Database

An XML Web Application *is*:

Both approaches combined -

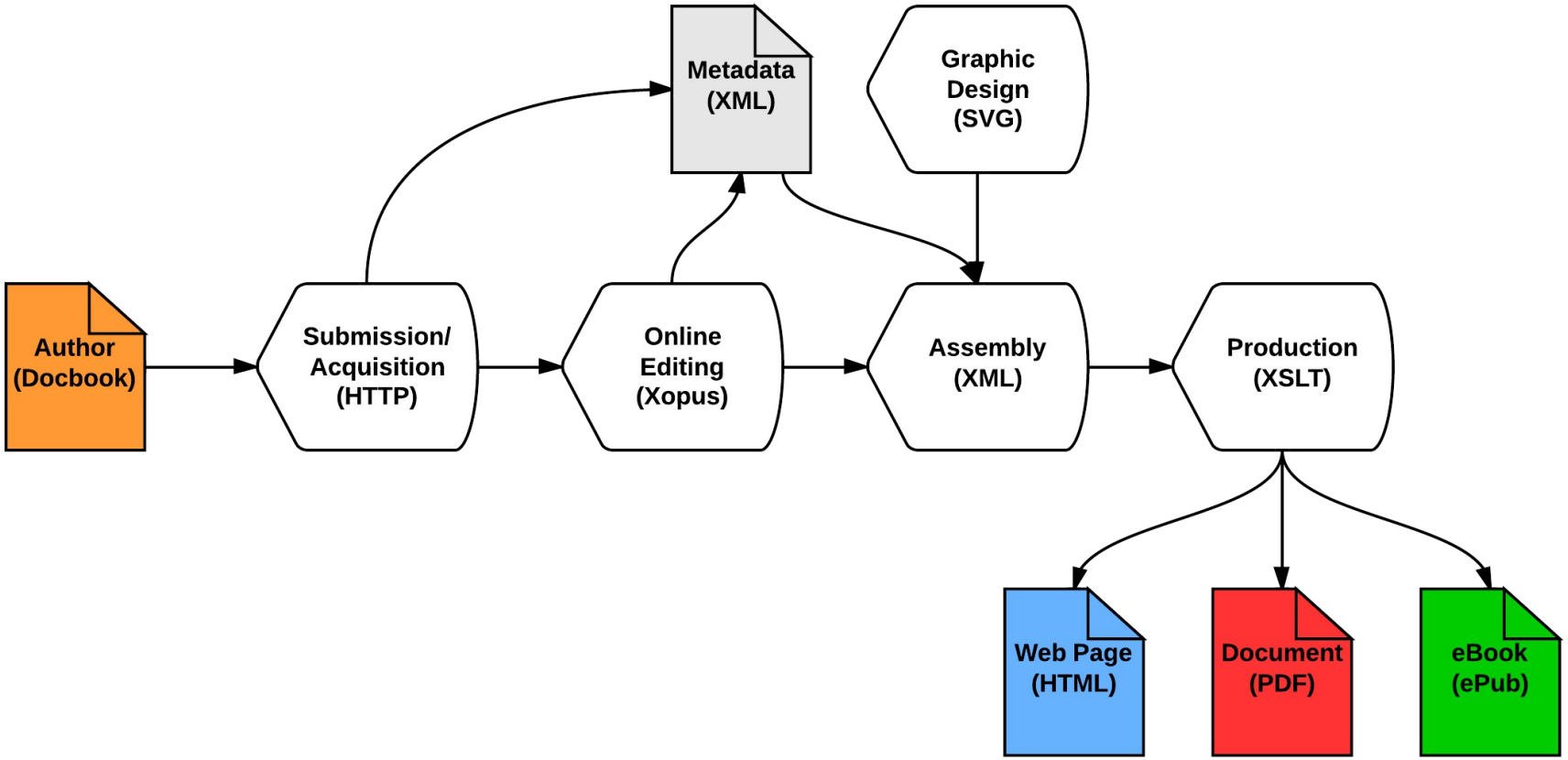
- A Web Application applied to XML
  - i.e. delivering your content
- XML applied to a Web Application
  - Built using XML technologies

Used when -

- Your content model is XML
- You want to deliver an Application (Intranet and/or Internet)

Relevance

- Why should traditional Publishers care?
  - XML Recognised as a good Content Model
  - Many arguments for XML early/throughout



Relevance

- Publishing is *NOT* Web Publishing
  - Publishing on the Web, previously minor part!
  - However, more important every day:
    - 566.4 % growth in Internet usage (2000-2012)
      (internetworldstats.com)

"everyone of working age online by the end of this Parliament"

Networked Nation Manifesto, Martha Lane Fox, 2010

Relevance

- IMHO:

  Publishers *must* recognise **Web as delivery** not destination!

  - Web is Bi-directional! *Deliver* online authoring and user generated content.
  - Web Applications now run in/on:
    - Desktop/Laptop
    - Mobile Phone
    - Tablet
    - e-Reader
    - ...How long until e-Paper?

Relevance

- IMHO:
  - Application Architecture
    - Most interesting applications have a client/server aspect
    - You might not need it online today, but...
    - Building fat-client Apps is harder than Web-Apps!
      - **All Applications should be Web Applications***
        - Publicly accessible?
- A Publishing Pipeline is an Application (or a few)
  - **Idea:**

    Let's build a Publishing Pipeline Web Application...

summer school

Publishing Pipeline: XML Web Application

what should it do?

Our Publishing Pipeline Requirements

- What should our Publishing Pipeline do?
  - **Input** - Articles from Authors
  - **Transform + Assemble** – Journal from Articles
  - **Output** - Journal

Our Publishing Pipeline Requirements

- Required steps in the Pipeline

  1) Take submissions from Authors
  2) Allow Authors to track progress
  3) Validate submissions from Authors
  4) Allow Editors to make changes online
  5) Validate post-editorial Content
  6) Assemble and Transform Articles
  7) Produce Web Page and PDF

- Web App bonus: Chargeable API for access to Content (RESTXQ)

summer school

XML Web Application Architectures

options for the architect and programmer

XML Web Application Architectures

- Three main options available:
  - Build with Traditional Web App Framework
    - ½ approach:
      - Applying Web App to XML (as data is in XML)
      - Not really Applying XML to Web App:
        - Unless also using XSLT/XQuery/XProc inside App*
  - Build with Native XML Web App Framework
    - Both:
      - Applying Web App to XML (as data is in XML)
      - Applying XML to Web App (as built using XML techs.)

Traditional XML Web App

- Traditional Programming + MVC? Framework e.g.
  - Java + Spring + Facelets
  - Ruby on Rails / Ruby + Sinatra
  - .net + Nancy

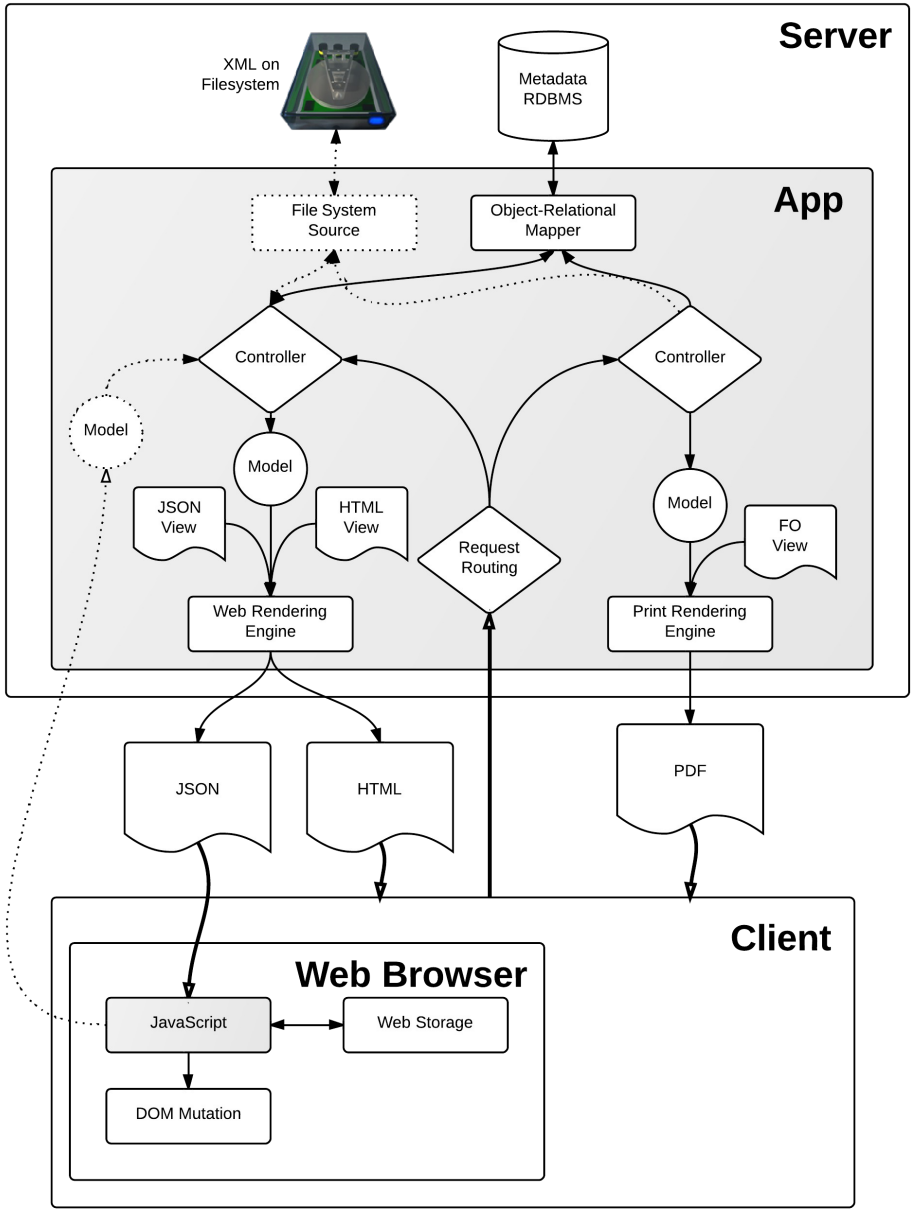
- How?
  - XML to/from File System/RDMS/NXDB etc
  - Transform XML files in your App language
    - DOM? SAX? XML-Object binding?
    - and/or call-out to XSLT processor
  - Generate PDF
  - Update parts of XML documents

Traditional XML Web App

- Advantages
  - Well worn path
  - Large established communities
  - Plenty of available, cheap programmers

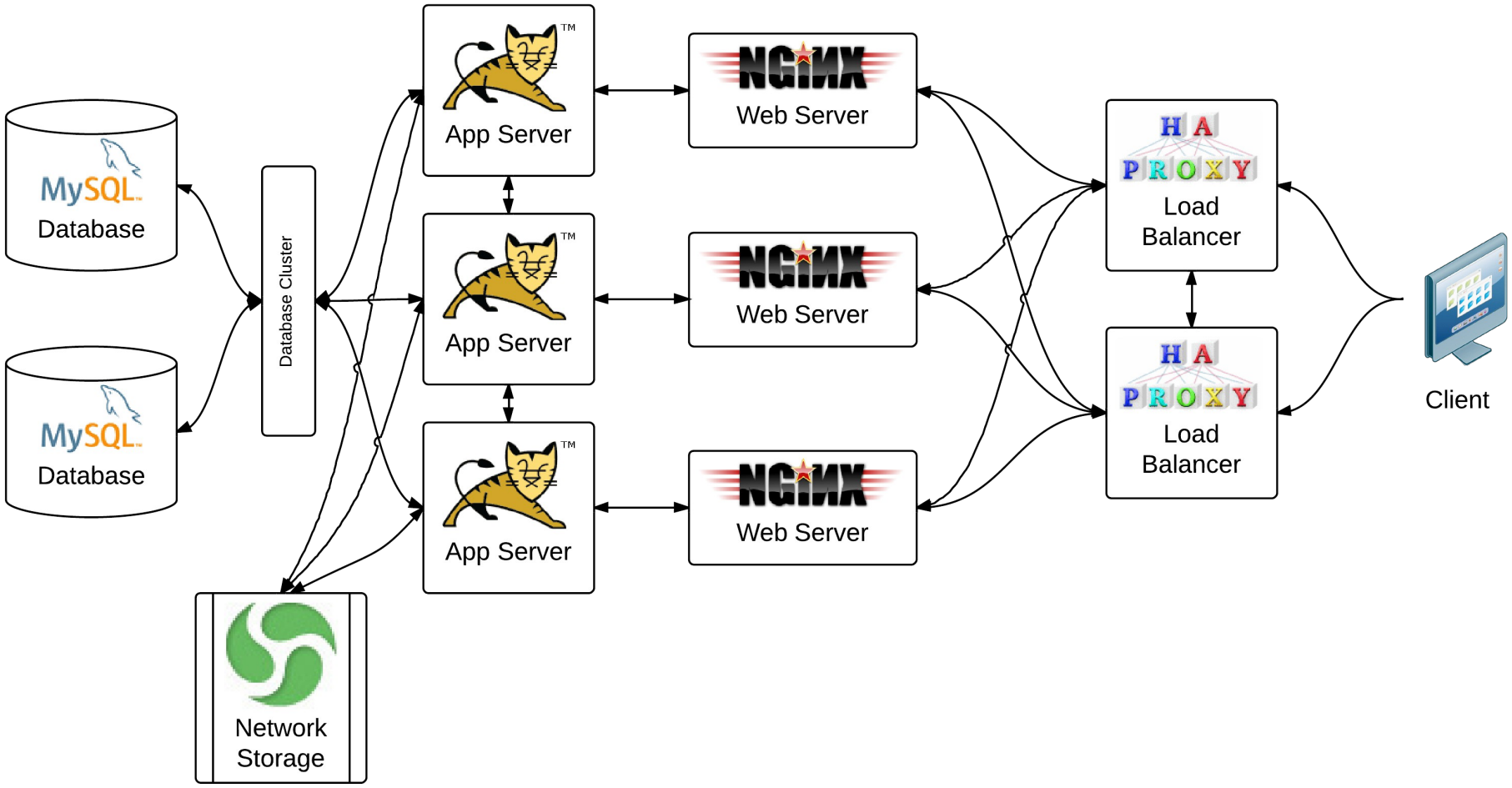
- Disadvantages
  - Hard to think Document Oriented (bytes, ints, strings, arrays, etc.)
  - Mapping in and out of XML (impedance mismatch)
    - Lots of lower-level code to write and maintain
    - Performance of model translation?
  - Many moving parts to the system

Traditional XML Web App



Traditional XML Web App

- Example Server Infrastructure



Native XML Web App

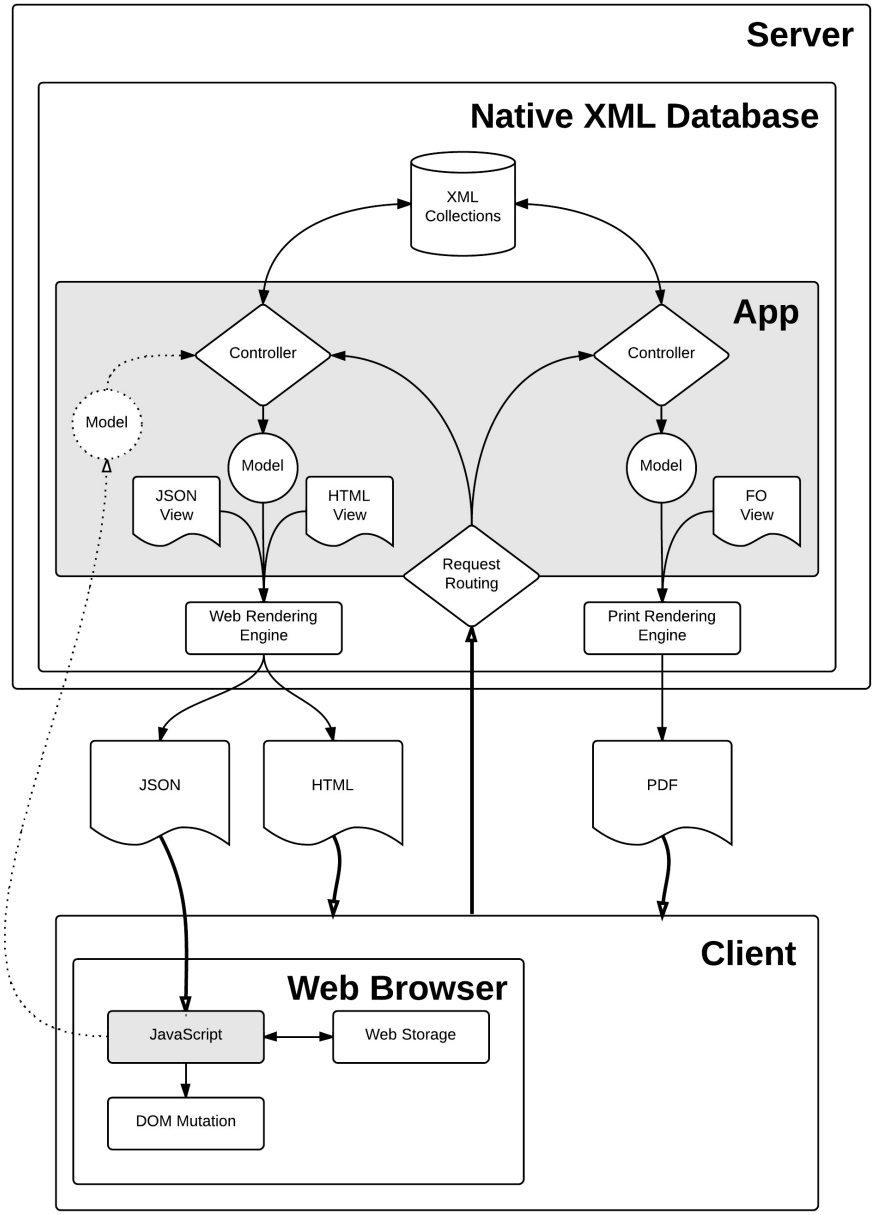
- Typically built around a Native XML Database
  - e.g. eXist, BaseX, Marklogic
- Programmed in XQuery/XSLT/XProc/XForms
  - **Many** Extension functions
    - Vendor provided (non-portable)
    - EXPath / EXQuery provided (portable)
  - Native XML DB offers Network capabilities
    - e.g. HTTP, FTP, SSH, WebDAV, etc.
    - Exposed through extension functions or injection
  - Provide XML Web App Frameworks

Native XML Web App

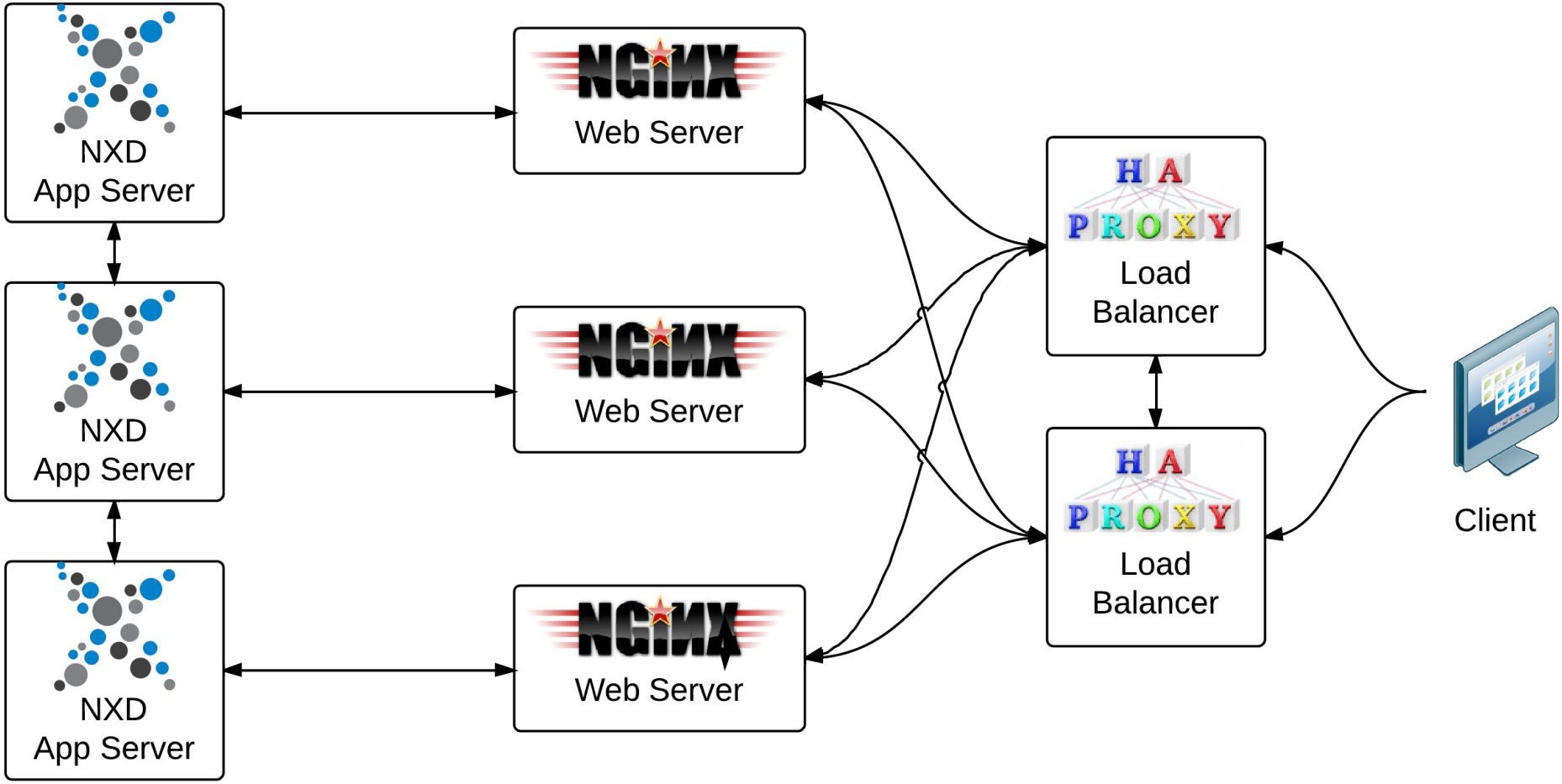
- Advantages
  - Everything is Document Oriented
    - Authors, Publishers, System Implementation and Web
    - Zero Translation
      - Less code to maintain (ETH*)
      - Less processing overhead?
  - Higher-Level Abstraction
    - Less to build, less to maintain
    - Faster to market!

- Disadvantages
  - Portability of code is difficult
  - Small established communities
  - Specialist skills (= more expensive programmers?)

Native XML Web App



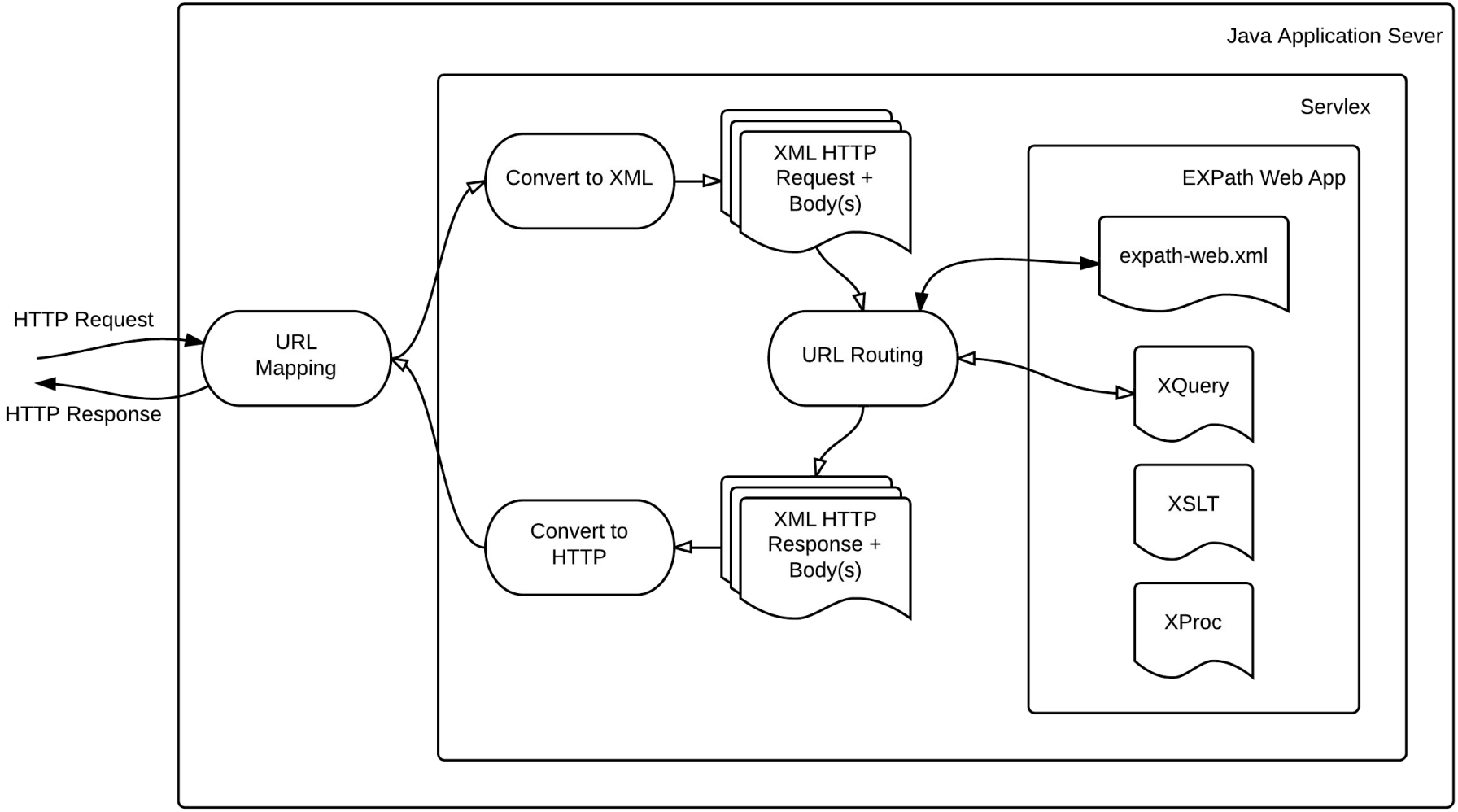
Native XML Web App



The Middle Road...

- Servlex
  - Implementation of EXPath Web Application spec.
    - Server + Framework
  - Between Traditional and Native approaches
    - Process Web Requests and Responses as XML docs
    - URL routing to XML processing Step
    - Impl. with XQuery, XSLT or XProc steps (can chain steps)
- Advantages
  - Java Servlet underneath; quicker than building your own!
- Disadvantages
  - XML is stored on File System
  - URI.Serialization/deserialization between steps

Servlex Architecture



Native XML Web App

- Why build around Native XML Database?
  - XML Documents are your key concern
  - Typically a Platform not *just* a database:
    - HTTP Server built-in
    - RESTful access to database
    - Binary storage and content extraction
    - Full-Text Search
    - JSON support
    - Transform/Query: XSLT, XQuery, XSLT, XProc
    - Author: XForms
    - and more...

Native XML Database

- XML Document is the unit of currency!
  - May also offer Binary storage (and extraction)
  - May also read and produce: RDF, HTML and JSON
- Documents organised into Collections*
- XQuery is the query language, *and more...*
- XML is stored:
  - In a manner that makes querying fast
  - Unlikely to be XML files on a file system

Why Native XML Database?

- Why not use a File System?
  - How to retrieve?
    - By file-path or some sort of lookup table?
    - i.e. Is a 'Directory' the same as a 'Collection'?
  - Where to keep metadata?
  - How to Query?
    - grep?
    - Integrate a search-engine (full-text), e.g. Apache Solr?
    - No direct XPath access!
  - How to Update?

Why Native XML Database?

- Why not use an RDBMS?
  - XML is just Text?!? (varchar / BLOB / CLOB)
  - Shredding
    - Every set of children is a table. Many tables!
    - Manual vs. Auto.
    - How to Query/Transform/Retrieve doc?
  - Many RDBMS offer XML storage (e.g. XMLType)
    - Oracle shred's behind the scenes, requires XML Schema.
    - Querying is often still driven from SQL
    - Joining XML and non-XML data is hard
  - How to Update? Full-text Search? Aggregate?

Native XML Database Options

- Marklogic
  - Commercial
  - XQuery 1.0, XSLT 2.0, XForms 1.1, RESTXQ, Bespoke Full-Text
  - Shared-Nothing Clustering
- eXist
  - Open Source. LGPL v2.1
  - XQuery 3.0, XSLT 2.0, XForms 1.1, XQuery Update, XProc, RESTXQ, EXPath, Bespoke Full-Text
  - Master-Slave Replication with Slave promotion.
- BaseX
  - Open Source. BSD License
  - XQuery 3.0, XQuery Update 1.0, RESTXQ, EXPath, XQuery Full-Text 1.0
- Others: Sedna / EMC Documentum xDB / Software AG Tamino / etc...

summer school

Working with the Web from XQuery

back to basics

XQuery 101

- XQuery is a Turing Complete Functional Programming Language
- We are not going to learn XQuery here!
- We need to know just enough
- Here is an XQuery:

```
xquery version "1.0";

<p>Hello to all at the XML Summer School</p>
```

XQuery 101

- A more interesting XQuery example:

```
xquery version "1.0";

<html>
<head><title>Example XQuery</title>
    <body>
        <p>Hello to {/person/name/text()} at the XML Summer
           School</p>
    </body>
</html>
```

- When given the XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <name>John Smith</name>
</person>
```

XQuery 101

- XQuery functions:

```
xquery version "1.0";

declare function local:say-hello() as element(p) {
    <p>Hello to {/person/name/text()} at the XML Summer School on
       {current-date()}</p>
};

<html>
<head><title>Example XQuery</title>
    <body>{local:say-hello()}</body>
</html>
```

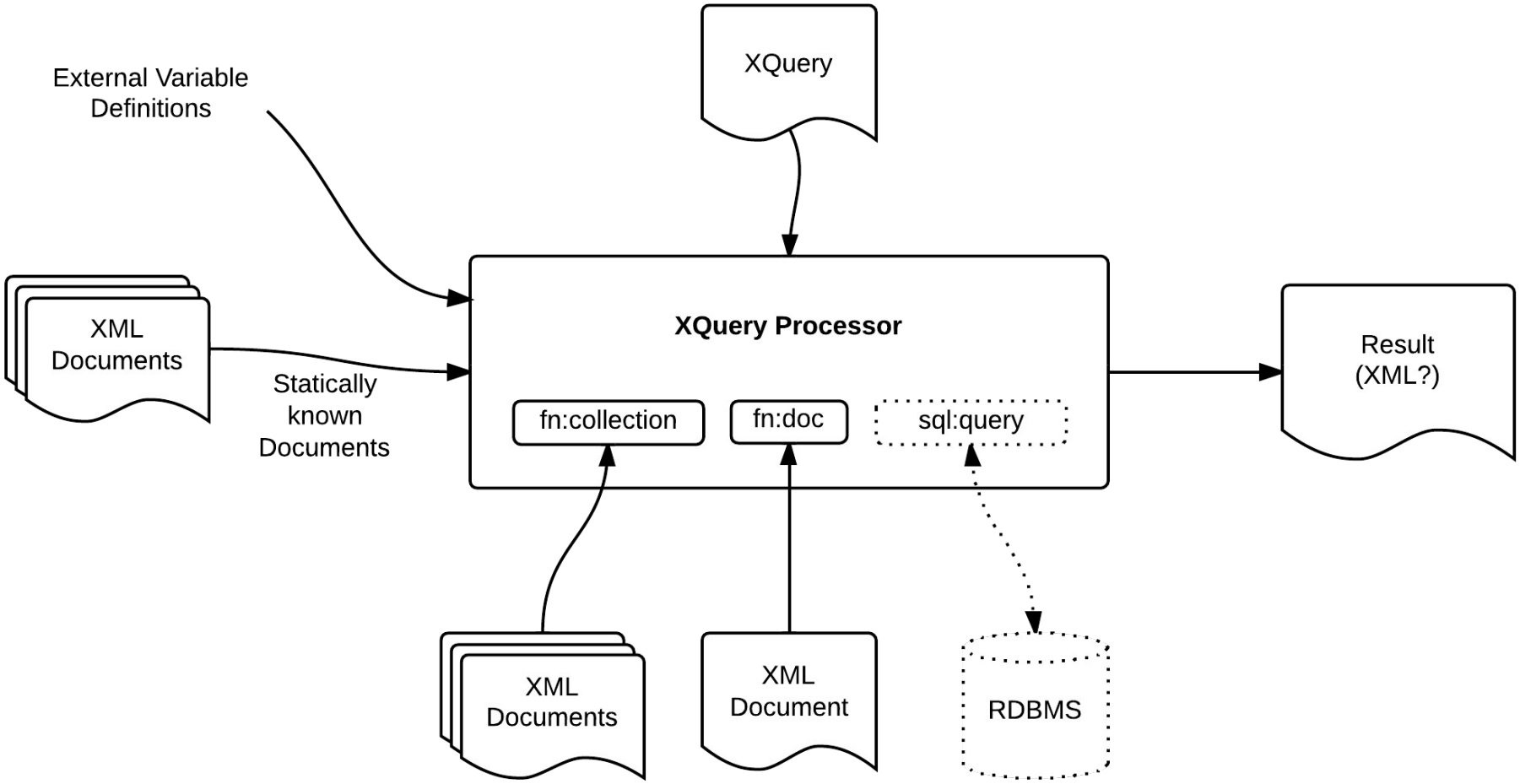
XQuery and the Web

- XQuery as specified by the W3C:
  - Has no concept of the Web or HTTP!
  - Not designed with Server Side Scripting in-mind!
- ∴ How can we use XQuery with the Web?

XQuery Processing

- XQuery can process input from:
  - Static Context
    - Statically known Documents
    - Value of External Variable Declaration
  - Function Call
    - XQuery 1.0 - fn:doc, fn:collection
    - XQuery 3.0 – fn:unparsed-text, fn:unparsed-text-lines, etc.
    - Extension function provided by XQuery Processor
      - Many available
      - Varies platform to platform

XQuery Processing



XQuery and the Web

- Two main integration options:
  - Loose: Call XQuery Processor from Web Server
    - Advantage: Easy to do
    - Disadvantage: XQuery Processor has no Web knowledge
  - Tight: XQuery Processor embedded in Web Server
    - Disadvantage: Harder to achieve
    - Advantage: XQuery has some Web knowledge
    - Advantage: Use one of many existing options
      - e.g. Native XML Database or Servlex

HTTP Request/Response with XQuery

- Tight integration options:
  1) Model HTTP Request/Response as XML(s)
     - Request is statically known document
     - Response is result of XQuery
  2) Just invoke the main XQuery
     - Provide *extension* functions to the XQuery
       - Functions to access HTTP Request properties
       - Functions to set HTTP Response properties
     - *Body* of Response is result of XQuery
  3) Direct XQuery function call
     - Inject HTTP Request (or properties) as function params
     - Response is result of XQuery

HTTP Request/Response with XQuery

- We will now look at example of each of the three approaches from the previous slide:
  - To understand the input and output for the XQuery
  - To understand how the XQuery
    - Processes the input
    - Creates the output

(1) HTTP Request/Response as XML Model

```
<http:request xmlns:http="http://expath.org/ns/http"
url="http://something.com/thing" method="post">
    <http:header name="Accept" value="application/xhtml+xml/>
    <http:header name="User-Agent" value="Your Browser"/>
</http:request>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <name>John Smith</name>
</person>
```

```
xquery version "1.0";
declare namespace http = "http://expath.org/ns/http";
(
    <http:response status="200">
        <http:header name="Context-Type" value="application/xhtml+xml"/>
    </http:response>
,
    <p>Hello {//name/text()}, I see you are using
          {//http:header[@name eq "User-Agent"]/string(@value)}</p>
)
```

(2) HTTP through Extension Functions

summer school

```
POST /thing HTTP/1.1
Host: http://something.com
Accept: application/xhtml+xml
User-Agent: Your favourite browser
Content-Length: 86
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<person>
    <name>John Smith</name>
</person>
```

```
xquery version "1.0";

import module namespace request = "http://exist-db.org/xquery/request";
import module namespace response = "http://exist-db.org/xquery/response";

let $body := request:get-data(),
$null := response:set-header("Context-Type" "application/xhtml+xml")
return
   <p>Hello {$body//name/text()}, I see you are using
      {request:get-header("User-Agent")}</p>
```

(3) Direct XQuery Function Call

summer school

```
<http:request xmlns:http="http://expath.org/ns/http"
url="http://something.com/thing" method="post">
    <http:header name="Accept" value="application/xhtml+xml/>
    <http:header name="User-Agent" value="Your Browser"/>
</http:request>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <name>John Smith</name>
</person>
```

```
xquery version "1.0";

module namespace app = "http://my-app";
declare namespace http = "http://expath.org/ns/http";

declare function app:some-thing($input as item()+) as item()+ {
  (<http:response status="200">
     <http:header name="Context-Type" value="application/xhtml+xml"/>
   </http:response>
  ,
  <p>Hello {$input//name/text()}, I see you are using
     {$input//http:header[@name eq "User-Agent"]/string(@value)}</p>)
};
```

XQuery for the Web

- So far we have seen:
  - XQuery can be used to generate XHTML
    - Basic Templating with {*code*}
  - XQuery can service HTTP Request/Response
    - Extract and use headers and body from HTTP Request
    - Set headers and body in HTTP Response
    - Requires some sort of **3rd Party** integration!
    - Several possible approaches.

What about XRX?

- XRX (XForms, REST and XQuery)
  - A Client/Server Web Application Architecture
  - 2MVC. MVC in XForms + MVC in XQuery.
  - Zero-Translation architecture. i.e. XML end-to-end
  - **Shallow XRX**: replace XForms with XML consumer
  - XQuery, may be replaced with XSLT or XProc
- We are keeping it simple!
  - Shallow XRX: Web-Browser + REST, + XQuery

XQuery on the Web extras

- Exciting things we have yet to consider:
  - HTTP QueryString Parameters
    - Extension Functions or Function Parameter Injection
  - Processing HTML Forms (HTTP QueryString/Body)
    - Extension Functions or Function + Parameter Injection
  - HTTP URI Templates and Routing
    - Web Frameworks!
  - Authentication
  - Serialization – Text, HTML5, JSON, JSON-P, etc.

XML Web App Frameworks*

- Written in XQuery (excl. Servlex† and RESTXQ‡)
- Provide a Higher Level of Abstraction
  - Less glue/boiler-plate for you to write
  - Operate by convention
    - Main Execution / Function + Parameter Injection
- Usually MVC or similar
  - Separation of Concerns
  - Controllers written in XQuery†
- URL Routing

XML Web App Frameworks*

	Created	Server Arch.	App Framework	Portability
REST Server	2003	REST + XQY execute	n/a	eXist
HTTP App Server	???	XQ exec.	n/a	MarkLogic
XQuery URL Rewrite	2008	n/a	URL Routing + View Pipelines	eXist
URL Rewrite	2009	n/a	URL Routing	MarkLogic
XQMVC	2009	n/a	MVC	eXist, MarkLogic
Corona	2010	REST	n/a	MarkLogic
REST	2011	REST + XQY execute	n/a	BaseX
REST Endpoint	2011	n/a	URL Routing	MarkLogic
mustache.xq	2011	n/a	View Templating	BaseX, eXist, MarkLogic
Roxy	2012	n/a	mVC	MarkLogic
REST API	2012	REST + XQY execute	n/a	MarkLogic
RESTXQ	2012	REST API	URL Routing View Serialization	BaseX, eXist, MarkLogic
templates.xql	2012	n/a	View Templating	eXist*

(3.1) Direct XQuery Function Call (RESTXQ)

```
POST /thing HTTP/1.1
Host: http://something.com
Accept: application/xhtml+xml
User-Agent: Your favourite browser
Content-Length: 86
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<person>
    <name>John Smith</name>
</person>
```

```
xquery version "3.0";

import module namespace rest = "http://exquery.org/ns/restxq";
declare namespace output = "http://www.w3.org/2010/xslt-xquery-
serialization";

declare
    %rest:POST("${body}") %rest:path("/thing")
    %rest:header-param("User-Agent", "{$user-agent}")
    %output:method("xhtml") %output:media-type("application/xhtml+xml")
function bc:author($body, $user-agent) {
    <p>Hello {$body//name/text()}, I see you are using {$user-agent}</p>
};
```

XML Web App Frameworks*

- Lots of Server and Framework options available
  - Each takes a different approach.
  - Very few are portable!

- Building an API? Use RESTXQ
  - Can build apps, but extra framework would help
    - mustache.xq is cross-platform (too simple?)

- Platform Specific
  - eXist: RESTXQ with betterForm or templates.xq
  - MarkLogic: Roxy / RESTXQ with XSLTForms

summer school

Architecture

Example:
Publishing Pipeline
XML Web Application

Example Publishing Pipeline

- Built on eXist Native XML Database
  - Controllers written entirely in XQuery
  - Models are just more XML
  - Views written in XSLT and XSL-FO
  - WebDAV and HTTP API for Submission
  - RESTXQ for API
  - templates.xql for HTML5 generation
- Publish to Web with HTML 5, CSS and JavaScript
  - Bootstrap and jQuery
- Publish to PDF with Apache FOP

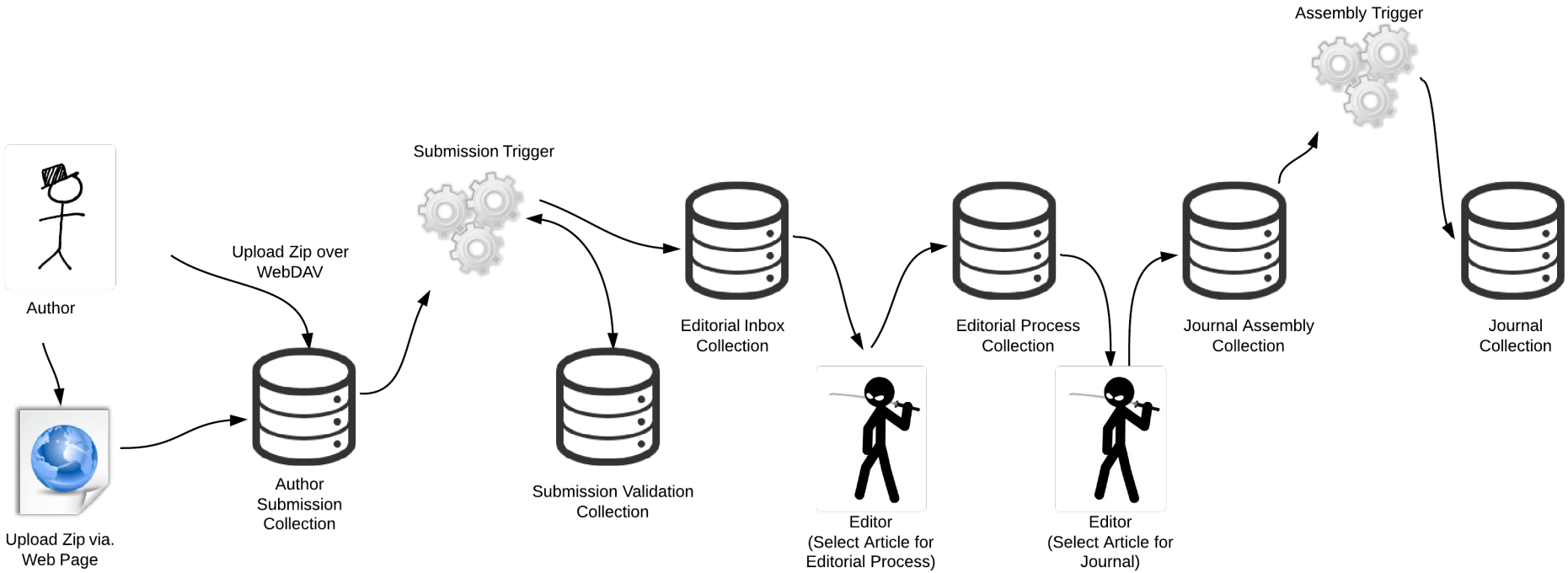
HTML Page Generation

- We are using Server Side HTML Generation
  - Quick using templates.xql to create HTML5 pages
  - Does not require learning lots of JavaScript
  - Alternative is *fat* HTML page + JavaScripts
    - Client is standalone – entire site is really one page
    - JavaScript to make AJAX calls to RESTXQ server on events
    - Mutate DOM with data retrieved from server
  - Obviously – Could combine both approaches!

Example Publishing Pipeline

- Collections model each stage of the Pipeline
  - Articles move from one Collection to the next
  - Collection Triggers allow us to automate Processes
    - Validation, Assembly etc.
- Authors and Editors have different Security Constraints!
  - Security applies to Processes as well as Documents
  - Authors should only see their own articles
  - Editors should see all
  - Public need to be able view published Journal!

Publishing Pipeline Collections



Security

summer school

- Two User Groups – Authors and Editors

Collection (under /db/publishing)	Security
articles/submission	Authors can Open Collection, cannot Read (so cannot see others articles) or Write.
articles/submission/<author>	Author owner can Open Collection and Read and Write.
articles/submission/validation	Authors can Open Collection, cannot Read (so cannot see others articles) or Write. Editors can Read and Write.
articles/submission/validation/<author>	Author owner can Open Collection and Read and Write.
articles/editorial	Authors can Open Collection, cannot Read (so cannot see others articles) or Write. Editors can Open and Read.
articles/editorial/inbox	Authors and Editors can Open, Read and Write.
articles/editorial/process	Only Editors can Open, Read and Write

summer school

Document Submission

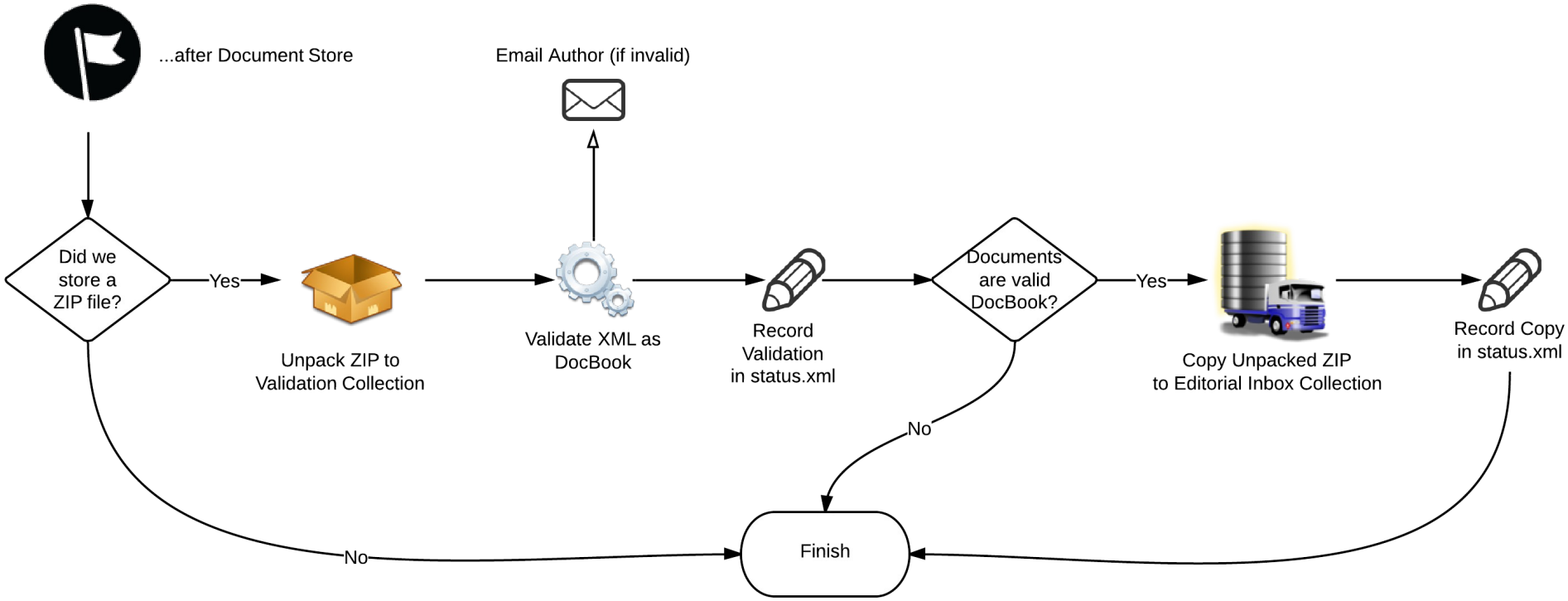
Example:
Publishing Pipeline
XML Web Application

Document Submission

- Author Submits Document
  - ZIP File containing DocBook and any externals
- Submission by HTTP:
  - WebDAV
    - Users know their OS File Browser!
  - Web Page Upload
    - Users only need a Web Browser

Submission Processing

- Processing a Submission
  - Collection Trigger, fires after a document is stored



Database Triggers

- Triggers
  - Document or Collection
  - Configured on a per-Collection hierarchy basis
  - Collection configuration in /db/system/config/db/...
  - Written in XQuery or Java
- Collection Trigger
  - Pre and/or Post Event
  - Create, Copy, Move, Delete: Collection
- Document Trigger
  - Pre and/or Post. Also SAX Stream (valid/store)
  - Create, Update, Copy, Move, Delete: Document

Submission Trigger

- Written in XQuery
  - We implement: `trigger:after-create-document($uri)`
  - In namespace: `http://exist-db.org/xquery/trigger`
  - Fired *after* the document was stored in the database
  - Document may be XML or Binary!
  - Path to document provided in `$uri`
  - We could have validated pre-store!
    - Can reject during pre-store, but then document is not stored into db!

XQuery Extensions used in Submission Trigger

- XQuery Update Extensions
- XMLDB Operations
  - `xmldb:create-collection($parent-coll, $name)`
  - `xmldb:get-child-resources($coll)`
  - `xmldb:copy($src-coll, $dest-coll)`
  - `xmldb:remove($coll)`
- Processing ZIP
  - `compression:unzip($bin, $filter-fn, $filter-params, $output-fn, $output-params)`
- Security
  - `sm:get-account-metadata($user, $property)`
- XML Validation
  - `validation:validate-report($doc, $schema)`

Authors Web Page

- Web Endpoint is provided by RESTXQ

```
declare
    %rest:GET
    %rest:path("/boredcat/author")
    %output:method("html5")
    %output:media-type("text/html")
function bc:author() {
```

- HTML is generated using templates.xql

```
templates:apply(
        $content,
        bc:templates-fn-resolver#2,
       $model,
       $bc:templates-config
)
```

Authors Web Page

- Three pieces of HTML templated together
  - One *surrounds* the other...
  - authors.html

```
<div class="templates:surround?with=page.html&amp;at=page-
content">
    <div>
        <h2>Your Articles</h2>
```

  - page.html

```
<body>
        <div class="templates:include?path=navbar.html"/>
```

Future Work

- Submission by:
  - email
    - Use Mail XQuery extension module for POP/IMAP
  - Version Control System
    - e.g. Git/GitHub hook. Use EXPath HTTP Client XQuery extension module.

- Submission of standalone DocBook (as well as Zip)

- DocBook Business Rules Validation
  - All external resources present in ZIP?
  - Meets various layout and formatting concerns?

summer school

Online Editing

Example:
Publishing Pipeline
XML Web Application

To be continued...

- http://www.adamretter.org.uk/presentations/xml-web-applications_xml-summer-school_oxford_20130917.pdf
- http://xmlsummerschool.com/materials/2013/Retter_xml_web_apps.pdf