Seminar OHJ-1860
Web Oriented Software Development

Pasi Liimatainen

Introduction

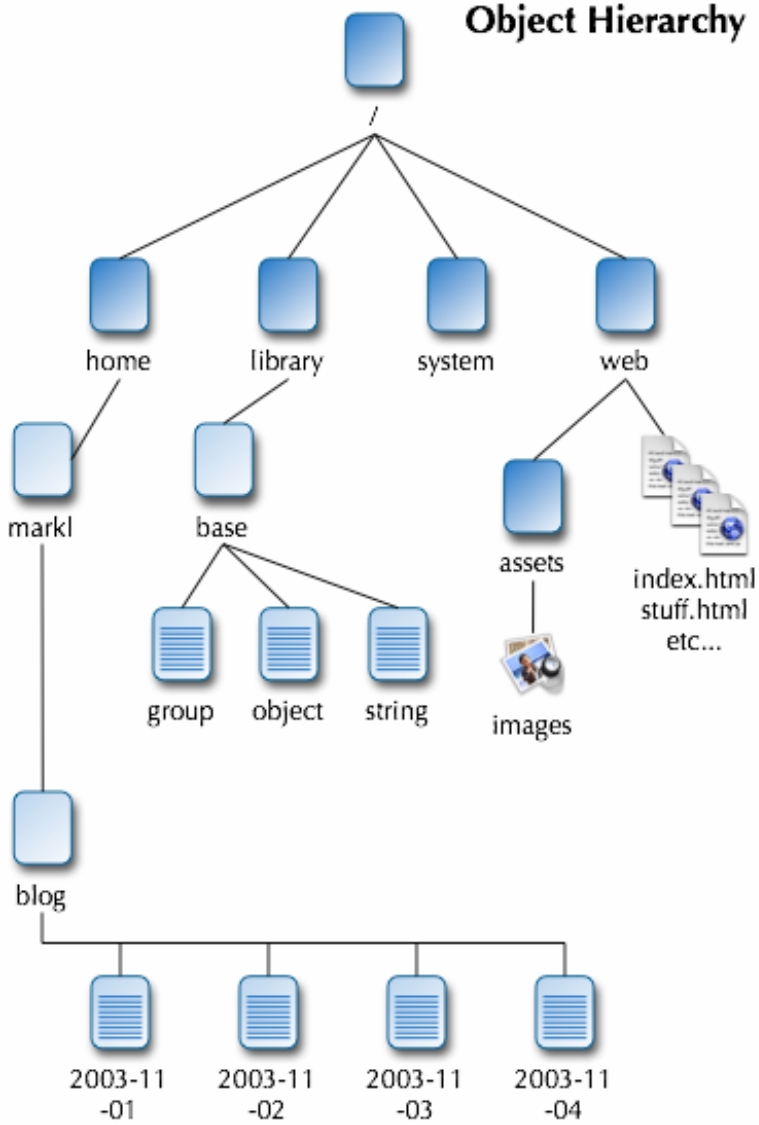
- Language, library and environment for small to medium scale dynamic web sites
- Objects as persistent, globally accessible entities
- Web based: every object has a URI, VM is a web server, development environment is a wiki
- First prototype in Nov 2003, but now apparently stalled
- Licenced under Academic Free License v. 2.0

Language

- Language in flux – lots of syntax & semantics change proposals in the wiki
- OO language – everything is an object
- Distinct features:
  - all objects have a distinct location (home object)
  - each object has a globally unique name (across the whole Internet)
  - no direct pointers or references to objects, only links
  - highly decoupled from storage (objects in different locations can be stored on different media)

Language - Object Model

- Explicitly hierarchical object model (tree)
- Objects have names & containers, and are tied to the life of their containers
- Everything is in a single tree – from integers to databases



Language - Inheritance

- Objects are prototype based
  - Any object can act as a class for another
- When an object's property is accessed, its ancestors are searched until one with the requested property is found (and its access is allowed)

Language - Constructs

- A script describes a tree of objects
- When compiled and loaded, real objects are created in Wheat’s object name space

```
``( simple script example ``)
a-book: {
    title: "Visual Explanations"
    author: { last-name: "Tufte";
    first-name: "Edward"; }
    id: 8274 in-circulation: true
    on-loan: false
}
a-room: {
    name: "Community Meeting Room"
    capacity: 85
    projection: true
    size: [ 25, 32 ]
}
```

Language - Statements

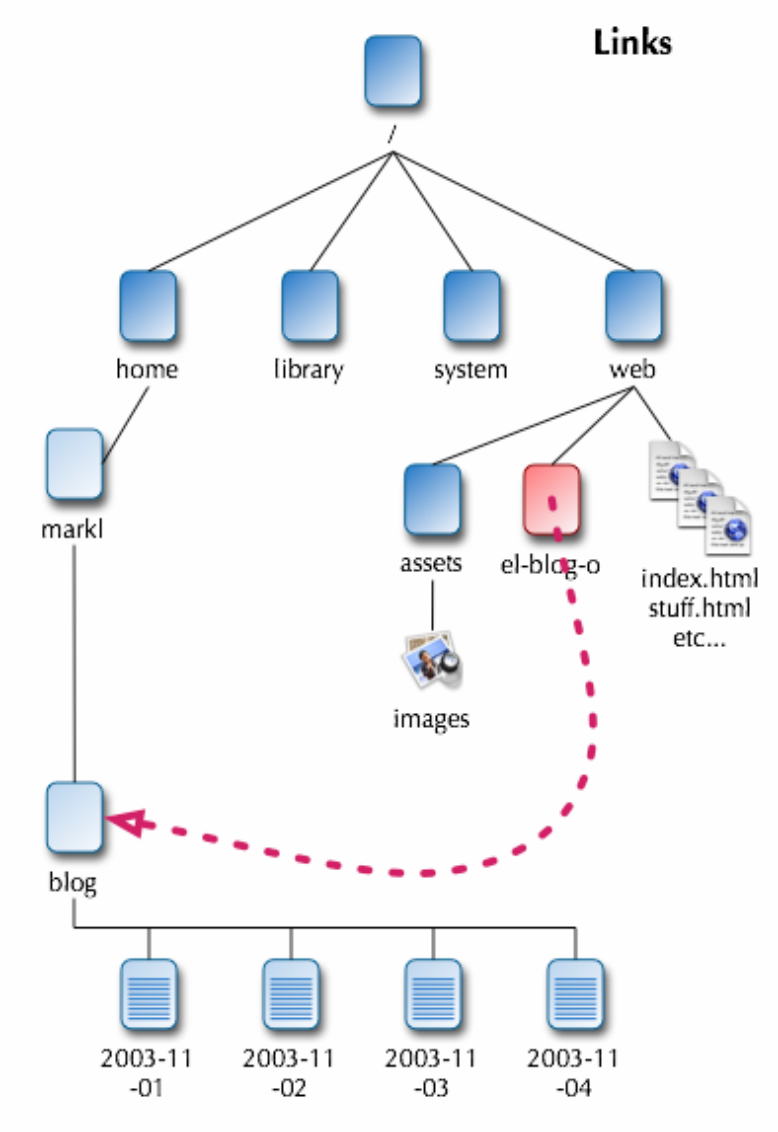
```
``( statements in a method ``)
record-vote(user, choice): {
   if (choice < 0 || choice >= #num-choices) {
        return !!!("bad-choice");
   }
   old-choice := #votes[user];
   if (old-choice?) {
        #counts[old-choice] -= 1;
        #total-count -= 1;
   }
   #votes[user] := choice;
   #counts[choice] += 1;
   #total-count += 1;
   i := 0;
   while (i < #num-choices) {
        #percentages[i] = #counts[i] / #total-count;
   }
}
```

Language - Errors

- An error value represents a failure to proceed
- Errors are *poison,* any attempt to manipulate one will continue to produce the error
- Error values have an object *behind* them to accumulate information about the error
- Ignoring an error makes it propagate from a method in place of what the method would have otherwise returned

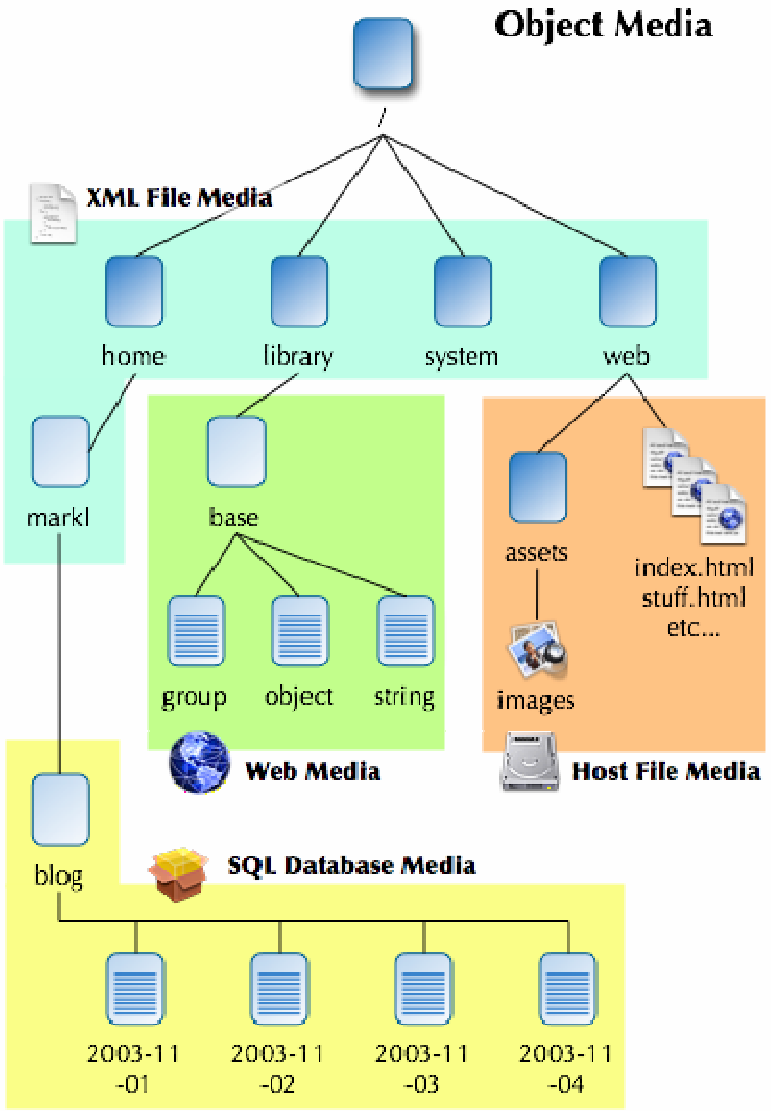
Language - Links

- Objects belong to a single container – two objects can’t share a subobject
- Links allow objects to refer other objects (”symbolic links”)
- Links are URLs
  - Absolute or relative
  - Local or remote



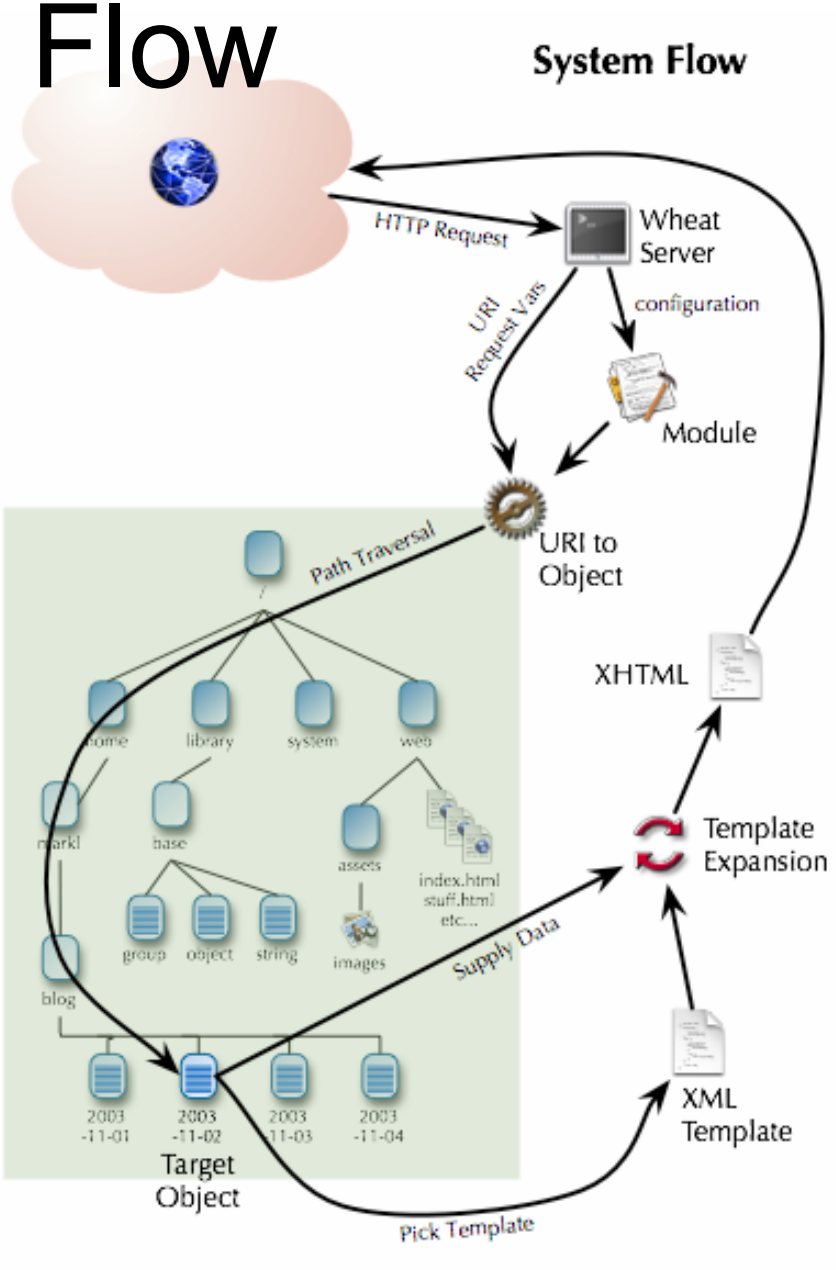
Language – Object Media

- Essentially mount points
- Parts of object tree stored with different techniques on different media
- Wheat programs see the tree as seamless object space
- Persistance and storage not application writer's concern



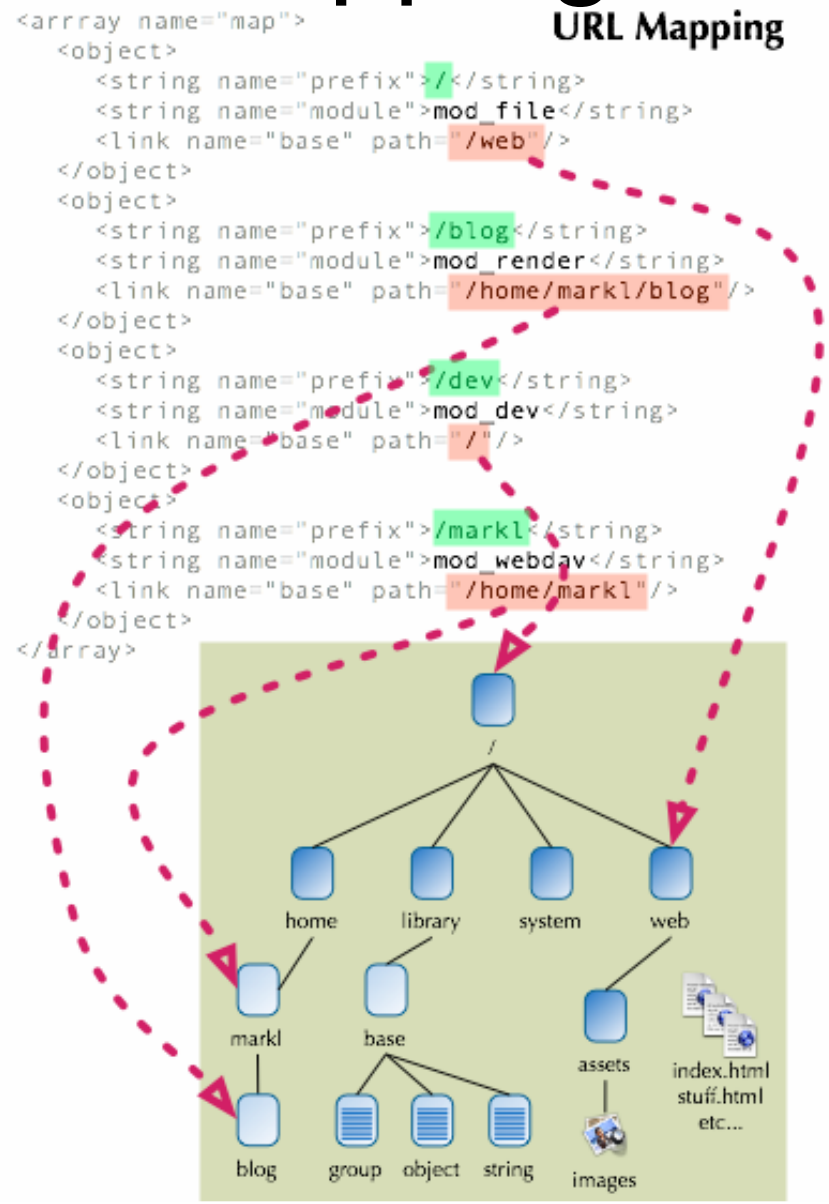
System Flow

- Wheat's VM is a multi-threaded web server
- HTTP request is mapped to a module based on URL and server configuration
- Module maps URL to an object
- Object renders itself using a template
- XHTML is returned



Modules and URL Mapping

- Modules allow mapping various parts of the object tree to different parts of the URL space



Template Engine

- A template is an XML document that provides the user interface for an object
- The template contains no executable code - all computation is in objects
- Using Tiny Template Engine
  - XML transformation system that has been implemented in several languages
  - clean separation of mark-up for presentation from both data and the logic behind presentation

Program Domain = App Domain

- Bridging the gap between user and developer experiences
  - Objects are persistent
  - Objects have URLs as part of a universal address space
  - Objects are rendered into XHTML and offer services via HTTP POST
  - The application model is stateless: there is no main(), and no long-term processes

Development Environment

- Wheat's development environment is a collaborative environment
  - ”Wiki for programming”
  - ”Pair programming, only multiplied”
- Code is ”always” hyperlinked to itself and its documentation

Instant Open Source

- Couple “every object has a URL” and “every object can be published on a web site" with collaborative environment
  - ➔ Nearly trivial open development

Example

- See blog application from Wheat's CVS

Evaluation

- The web-based development environment seems like an odd choice – would this work for anything but small projects?
- Objects directly on the web – hard to see why every integer would need to be on the web
- However, the object tree model with object media is interesting in itself
- Templates separating code from presentation is nice and used in several other toolkits
- Syntax and semantics not very nice/clear and in a flux
- Current examples are very small → scaling?
- Project stalled → no easy, working installation available (a prototype at best)

Summary

- Wheat contains some interesting concepts
- Unfortunately some of them might have been a bit high-flying
- If one doesn’t want to develop Wheat itself, one should not start working with it