Using TAL+XSLT to achieve language independent views in web applications

Zach Miller
zach@zarfmouse.us
http://twitter.com/zarfmouse

Separation of Data & Presentation Nirvana!

Zach Miller
zach@zarfmouse.us
http://twitter.com/zarfmouse

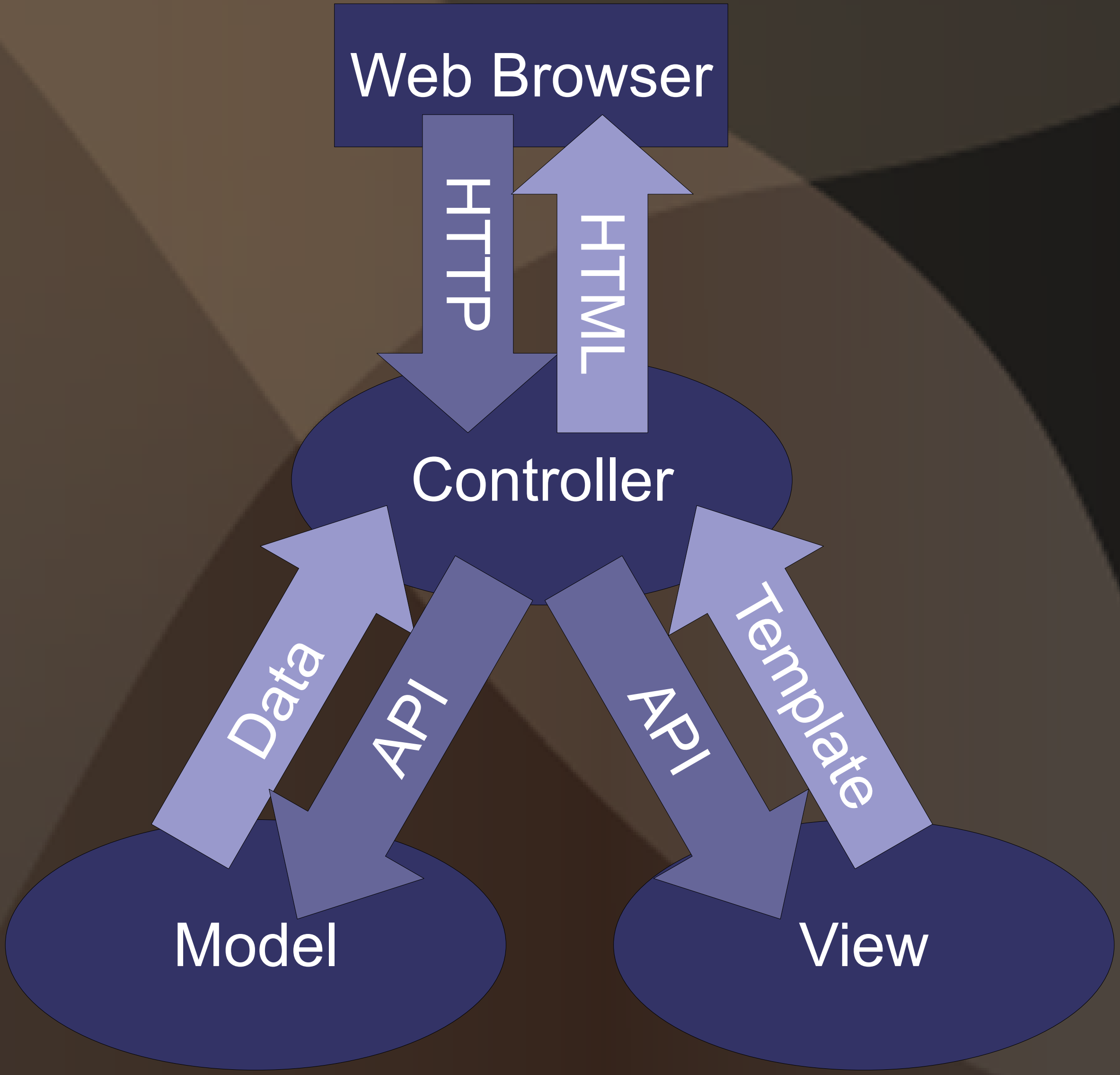
Who is Zach?

- Web Development Lead Programmer/Architect.
- Writing web applications since 1994.
- This funny accent is from Chicago.

Web Application Design Process

- A good web designer may design beautiful and usable HTML/CSS pages, perhaps with Dreamweaver, but might not be a very good programmer.
- A good programmer may write awesome database queries and abstract business logic but designs ugly web pages.
- The need for integration between these roles creates project friction.
- Effort can be saved by separation.

Model-View-Controller



Language Independence

- The View is HTML+CSS+JavaScript.
  - Why should it depend on anything else?
- The View is primarily declarative.
  - Why should it have embedded logic?
- What if you decide to re-implement in a different language?

Introducing TAL

- Template Attribute Language
- Originally specified by ZOPE (Python CMS)
- Start with a plain old static HTML document.
- Add tal attributes to map data from some structured data source into the HTML.
- Now your static HTML has become a view template for a dynamic web application.
- Multi-platform: data source could be Python, Perl, PHP, JavaScript, XML.
- Let's look at using XML.

Template Attribute Language

```
<page>
    <title>Dynamic Value</title>
</page>
```

Model

```
<span tal:content=”/page/title”>
    Dummy Value
</span>
```

View

```
<span>
    Dynamic Value
</span>
```

Template Attribute Language

```
<page>
    <class>dynamic</class>
</page>
```

Model

```
<span class="testing"
  tal:attributes="class /page/class">
    Some text.
</span>
```

View

```
<span class="dynamic">
    Some text.
</span>
```

Template Attribute Language

```
<page>
    <list>
        <item>one</item>
        <item>two</item>
    </list>
</page>
```

Model

```
<ul>
    <li tal:repeat=”item /page/list”
      tal:content=”$item”>Test</li>
</ul>
```

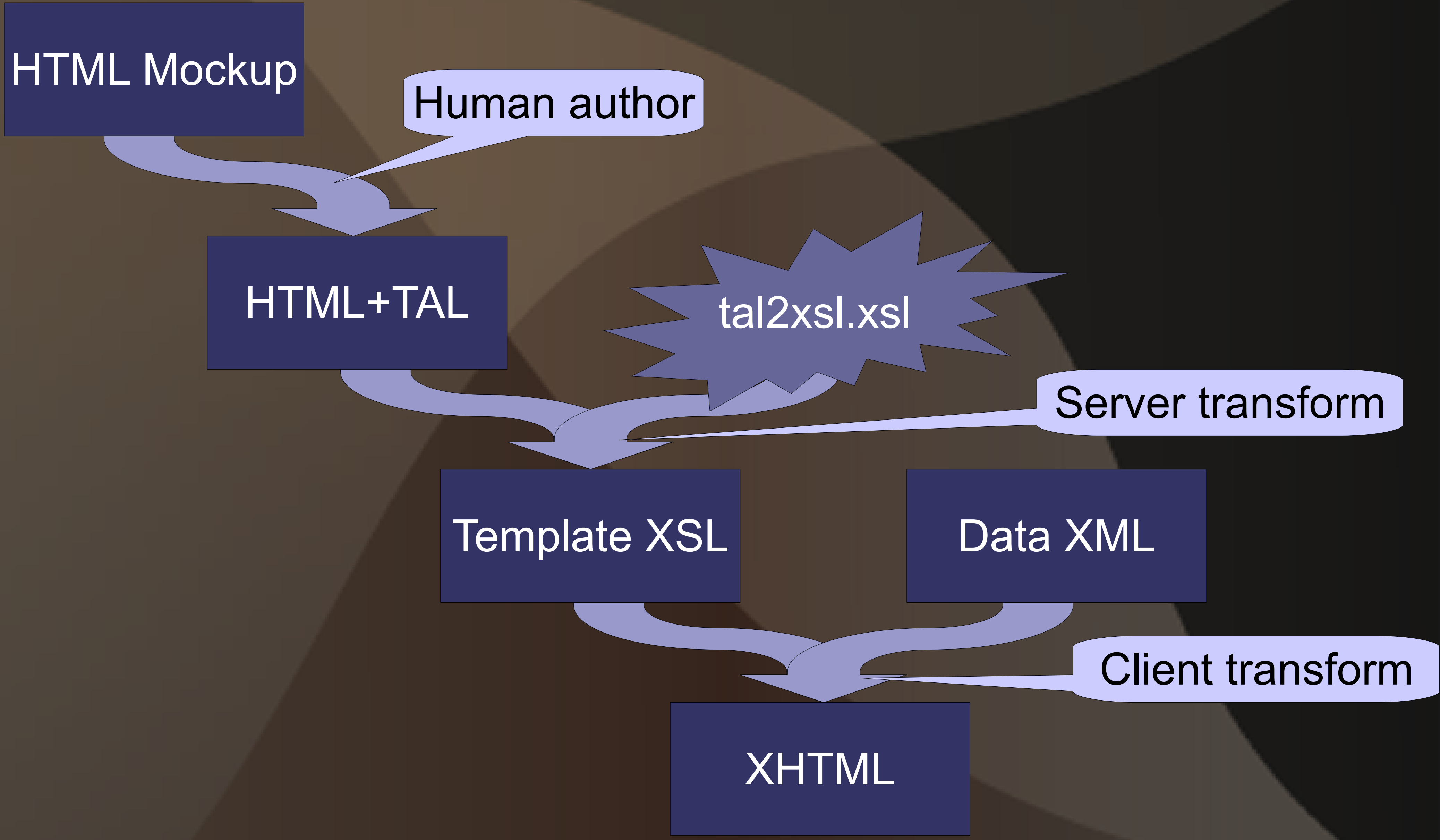
View

```
<ul>
    <li>one</li>
    <li>two</li>
</ul>
```

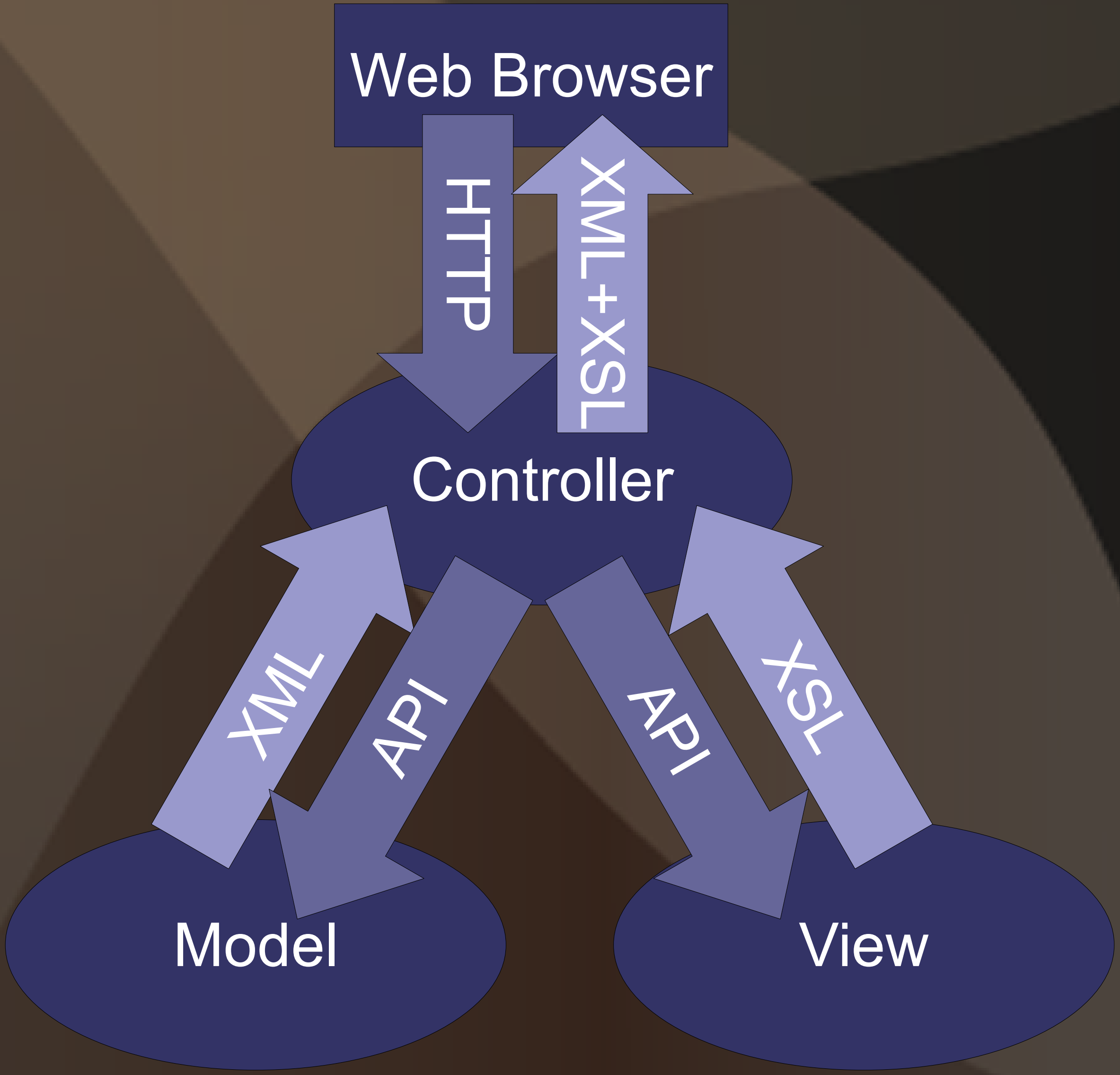
The Power of XSLT

- XSLT is XML that transforms XML into XML.
- TAL is XML.
- XSLT is XML.
- XHTML is XML.
- Let's use XSLT as the glue between our XML data and our TAL template!

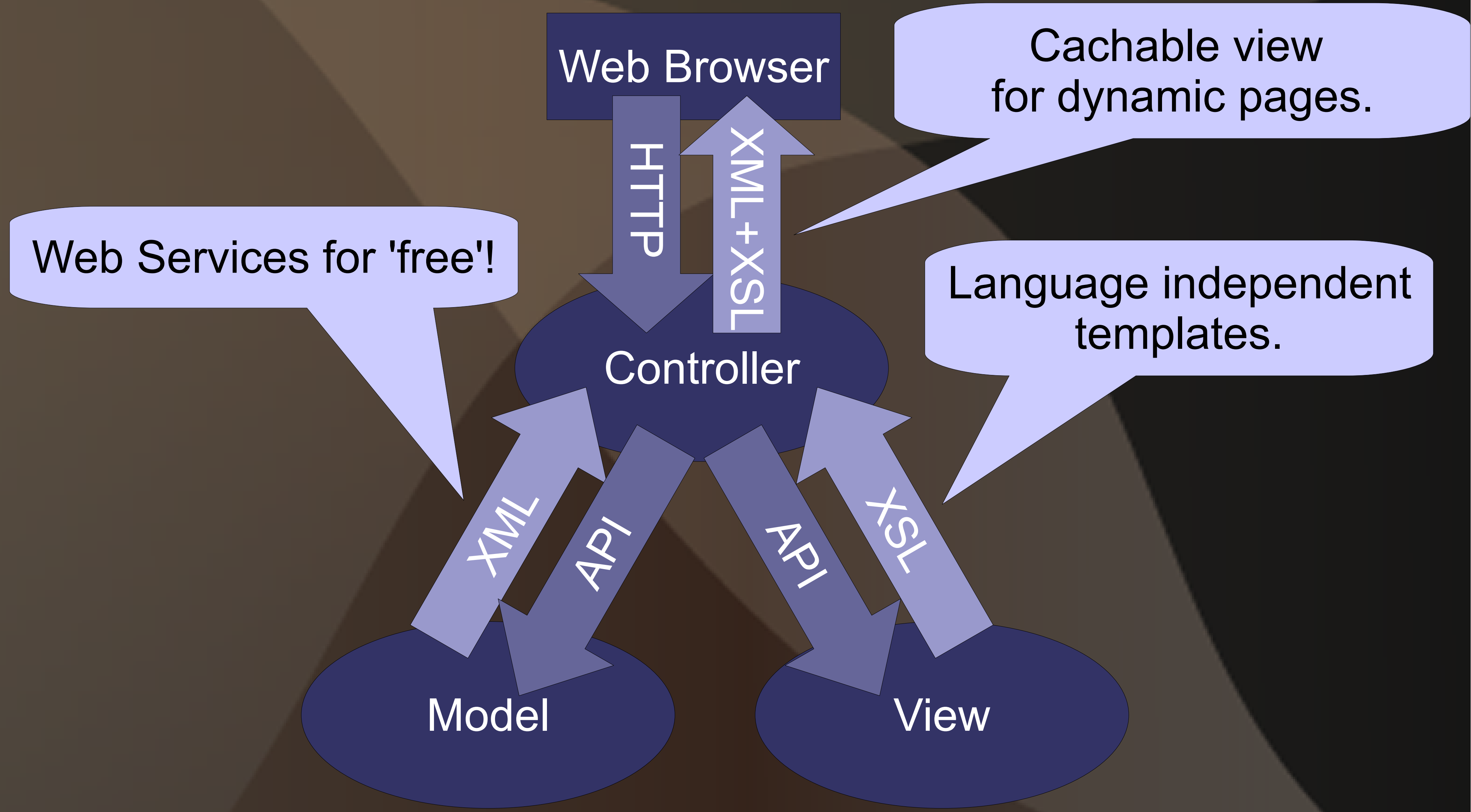
Transforming TAL to XSLT



Model-View-Controller



Model-View-Controller



It works!

- Tested in every modern browser. Even IE6!
- The XSL is cached on the server side.
- The XSL is cached on the client side.
- Fast page loads.
- Easy re-designs/re-brands.
- Easy LOTE/i18n.
- Manage the entire View in Dreamweaver.

What doesn't work?

- Mysterious whitespace.
- document.write();
    - But innerHTML does work.

What's to be done?

- Test the effects of this technique on SEO.
- Write a DTD for xhtml+tal.
- Use TAL files as unit tests for the controller. XSLT coverage?
  - Did the XSL use every value from the XML?
  - Did the XML provide every value the XSL needed?
- Support for fragment transforms for AJAX.

More information

- TAL Specification from Zope (Python CMS)
  - http://wiki.zope.org/ZPT/TALSpecification14
- Wikipedia
  - http://en.wikipedia.org/wiki/Template_Attribute_Language
- Original tal2xsl.xsl from Flux (PHP CMS)
  - http://wiki.flux-cms.org/display/FLX/Templates+XSLTAL
  - http://php5.bitflux.org/xsltal/
- Zach's tal2xsl.xsl
  - http://www.zarfmouse.us/tal2xsl