


**MELD3:
ALTERNATIVE
TEMPLATING**

**PRESENTED BY
CHRIS McDONOUGH
FOR THE 2006 PLONE SYMPOSIUM
NEW ORLEANS, LA**

WHY?

- Why in the name of all that is good and holy do we need *another templating system*?
- We don't. I was just avoiding real work.
- I'm sorry.

REASONABLE EXCUSES

- I hate hand-writing XML and HTML.
- But I  to write Python.
- GUI layout tools like *Dreamweaver* and *NVU* definitely don't work with non-XHTML or HTML-compliant templates.
- They even seem to ravage ZPT from time to time.
- `meld3` provides the possibility of cross-language template sharing.

HISTORY

- Casey Duncan: “Inside-Out ZPT” and “TAL Inheritance” (proposals)
- Richie Hindle: PyMeld
- Paul Winkler: Meld2 (prototype)
- Procrastination bore meld3.

ISSUES

- meld3 isn't for everybody. It's radically different than ZPT / DTML and others.
- meld3 can only template XML and HTML (not arbitrary text).

SIMILAR SYSTEMS

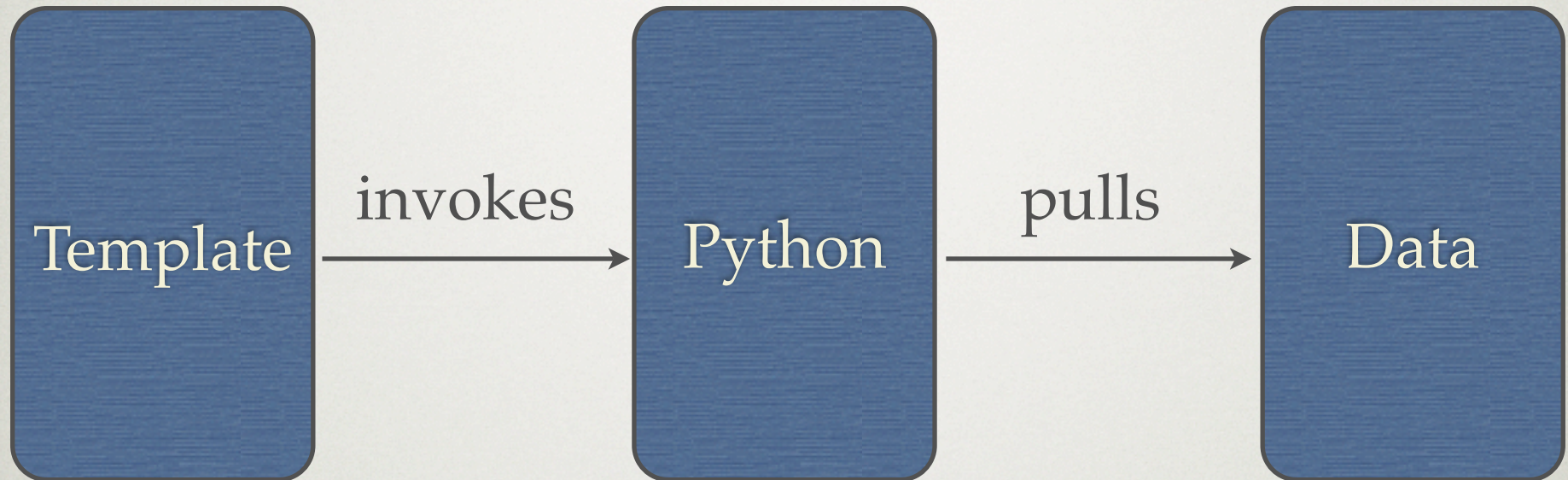
- HTMLTemplate / texttemplate (Python)
- XMLC (Java)
- Amrita2 (Ruby)

Interesting to note: some of these templating systems can share templates across languages.

COMPARING TEMPLATING PARADIGMS

- “Pull-based” (aka macro-based) systems like PHP, DTML, ZPT, Kid.
- “Push-based” (aka DOM-based) systems like meld3.

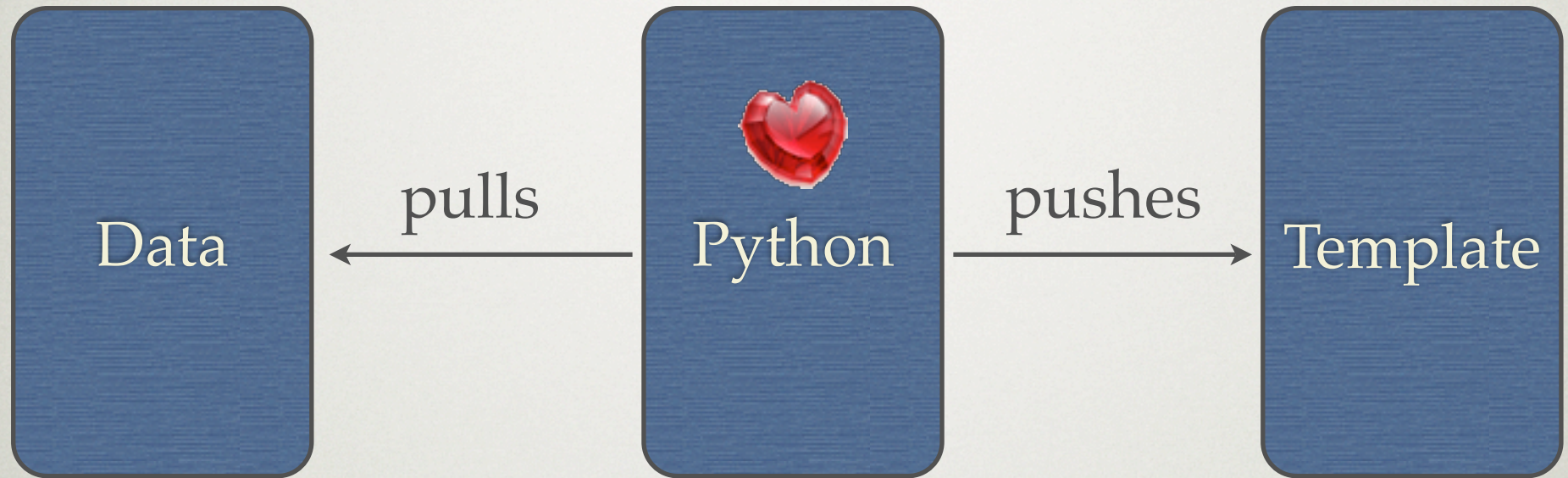
“PULL-BASED” TEMPLATING



Pull-based (aka “macro”) templating systems invoke Python, which operates against data.

`tal:define="here/getData"`

“PUSH-BASED” TEMPLATING



Push-based (aka “DOM-style”) templating systems pulls data and pushes it into the template.

`template.findmeld('replaceme').replace('Hello')`

WHO CARES ABOUT PUSH?

- If you're a designer that works in GUI tools or you work with a designer, you probably care.
- If you want your customers to be able to edit your templates without needing to understand the "programmery" bits, you probably care.
- If you need to share templates across languages, you probably care.
- If you'd rather write Python than HTML and XML, you probably care.

WHAT DOES MELD3 LOOK LIKE?

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:meld="http://www.plope.com/software/meld3">
  <head><title meld:id="title">This is the title</title>
  <body>
    <table border="0" meld:id="table1">
      <tbody meld:id="tbody">
        <tr><th>Name</th><th>Description</th></tr>
        <tr meld:id="tr" class="foo">
          <td meld:id="td1">Name</td>
          <td meld:id="td2">Description</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

declaration

meld node

THAT'S IT.

meld3 templates include exactly two things


An xmlns identifier for the meld namespace:

```
xmlns:meld="http://www.plope.com/software/meld3"
```

Identifying attributes for “meld nodes”:

```
meld:id="td2"
```

BUT THERE'S NO "THERE" THERE?

- That's right. No loops. No conditionals. No variable definitions. No inline code. No macros.
- All the hard work is done in Python. 
- We mutate the template definition by working against the "meld nodes" in the tree. When we're done, we serialize the result.

WITNESS

```
from meld3 import parse_htmlstring
html = open('sample.html').read()
root = parse_htmlstring(html)
root.findmeld('title').content('My document')

data = ({'name': 'Boys',
         'description': 'Ugly'},
        {'name': 'Girls',
         'description': 'Pretty'},)

iterator = root.findmeld('tr').repeat(data)
for element, item in iterator:
    element.findmeld('td1').content(item['name'])
    element.findmeld('td2').content(item['description'])

result = root.write_htmlstring()
print result
```

parser

content

repeat

serialize

find

RESULT OF SERIALIZING

...

```
<head><title>My document</title></head>
```

result of content



...

```
<body>
```

...

```
<tr><th>Name</th><th>Description</th></tr>
```

```
<tr class="foo">
```

```
<td>Boys</td>
```

```
<td>Ugly</td>
```

```
</tr>
```

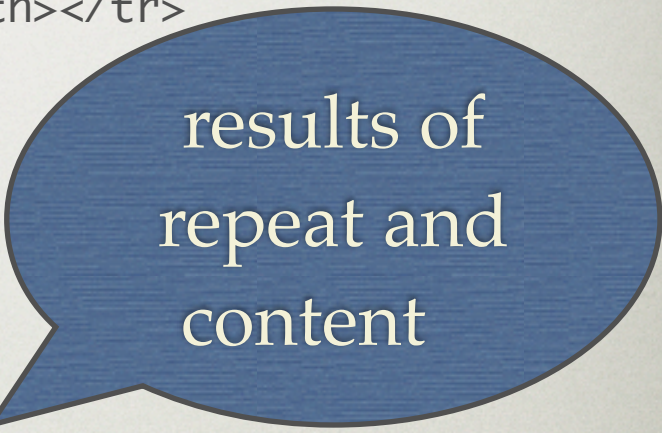
```
<tr class="foo">
```

```
<td>Girls</td>
```

```
<td>Pretty</td>
```

```
</tr>
```

results of
repeat and
content



...

```
</body>
```

...

BUT, BUT...

I told you it wasn't for everybody.

COMMON API METHODS

- *findmeld* - find a meld node
- *repeat* - repeat a meld node
- *replace* - replace a meld node
- *content* - replace the content of a meld node
- *attributes* - add attributes to a meld node
- *fillmelds* - fill meld node contents with the values of a dictionary

ELEMENTTREE API

- “meld” nodes are really just ElementTree nodes “under the hood”.
- Any API mechanism supported by ElementTree “_ElementInterface” nodes may be used against meld nodes as well. (e.g. `node.text = “foo”`)
- This means you can access nodes in the tree without doing “findmeld” but it’s not recommended.
- Mutation via ET API is OK, though.

FILLMELDS EXAMPLE

- *fillmelds* is a handy method. It accepts a keyword list as an argument.
- For each keyword argument, it finds a meld node in the template with that key, and fills the content of the meld node with the value associated with the key.

FILLMELDS EXAMPLE

```
SIMPLE_XML = r"""<?xml version="1.0"?>
<root xmlns:meld="http://www.plope.com/software/meld3">
  <list meld:id="list">
    <item meld:id="item">
      <name meld:id="name">Name</name>
      <description meld:id="description">
        Description</description>
      </item>
    </list>
  </root>"""
```

```
from meld3 import parse_xmlstring
root = parse_xmlstring(SIMPLE_XML)
d = {'description': 'Duh', 'name': 'Chris'}
root.fillmelds(**d); print root.write_xmlstring()
```



replacement
dict

FILLMELDS RESULT

```
<?xml version="1.0"?>
<root>
  <list>
    <item>
      <name>Chris</name>
      <description>Duh</description>
    </item>
  </list>
</root>
```



Node content after
filmelds

FILLMELDS SHORTCUT

Use the `__mod__` operator (%)

```
element % { 'description' : 'foo' }
```

STRUCTURE DIFFING

- two meld3 templates can be compared via a “structure diff”.
- a “structure diff” displays meld nodes which have been added, removed, or moved from one place to another.
- handy for seeing if someone broke your transform code by changing a template.

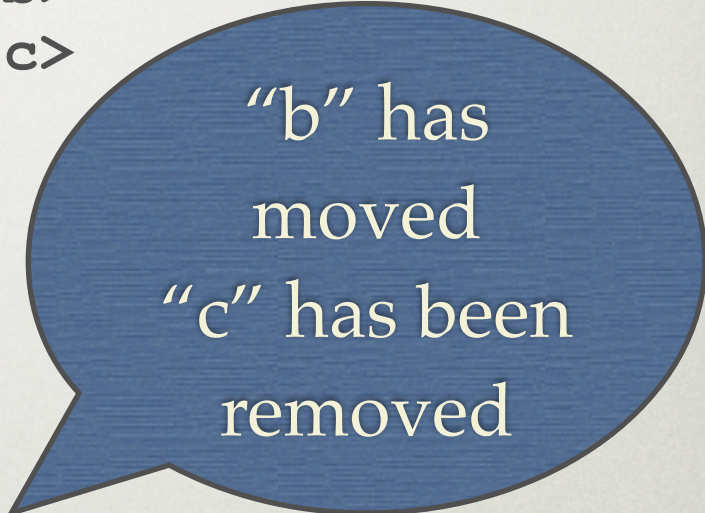
STRUCTURE DIFF INPUT

before.xml

```
<root>
  <a meld:id="a">
    <b meld:id="b"></b>
    <c meld:id="c"></c>
  </a>
</root>
```


after.xml

```
<root>
  <a meld:id="a"></a>
  <b meld:id="b"></b>
</root>
```



"b" has
moved
"c" has been
removed

STRUCTURE DIFF OUTPUT



melddiff.py
ships with meld3

```
[chrism@oops meld3]$ python melddiff.py \  
before.xml after.xml
```

Removed: c

Moved: b

```
[chrism@oops meld3]$
```

MELD3 AND ZOPE

- The “meld3” package isn’t tied to Zope in any way. It is Python and C without any dependencies on Zope.
- There is a “z3meld” bindings package for Zope3 / Five that makes meld3 easy to use in Zope.
- “z3meld” provides ZPT-macro-like functionality and easy view integration.

Z3MELD SCREENCAST

As if the other ones weren't, this slide is pointless.

WHERE DOM TRAVERSAL SUCKS

- For lots of tasks, traversing the DOM and mutating nodes just plain sucks.
- `meld3` has a helper for the task of filling in HTML form values, which is one place where using the DOM really, really sucks.
- *`fillmeldhtmlform`* accepts a dictionary and mutates `meld` nodes which purport to be form values in useful ways.

FILLMELDHTMLFORM SCREENCAST

I'll just show it.

NEED FOR SPEED

meld3	1.26ms / rendering (meld3 CVS, simple test script)
ZPT	2.48ms / rendering (ZPT from Zope 3.1.0, simple test script)

See http://www.plope.com/Members/chrism/meld3_zpt_profiling_madness

BUT ALL ISN'T ROSY

meld3	9.5 ms/request (z3meld CVS view, Zope 3.1.0 benchmarked via 'ab')
ZPT	9.5 ms/request (ZPT view from Zope 3.1.0)

Something is slowing the meld bindings down. See plope.com (same URL as last slide) for details.

LESSONS LEARNED

- The Zope3 view composition machinery is fairly magical.
- But it's still fairly easy to integrate new templating systems into Zope 3 and Five via the component architecture.
- ZPT is very fast for being written entirely in Python.
- Benchmarking is hard but useful.

WHERE TO GET IT

<http://www.plope.com/software/meld3>

THINGS I'D LIKE TO SEE

- meld3 integration with other Python frameworks.
- meld3 implementations in other languages.
- Configurable meld identifier (e.g. "id" instead of "meld:id").
- Fix speed of bindings under Zope 3 / Five.

THE END
