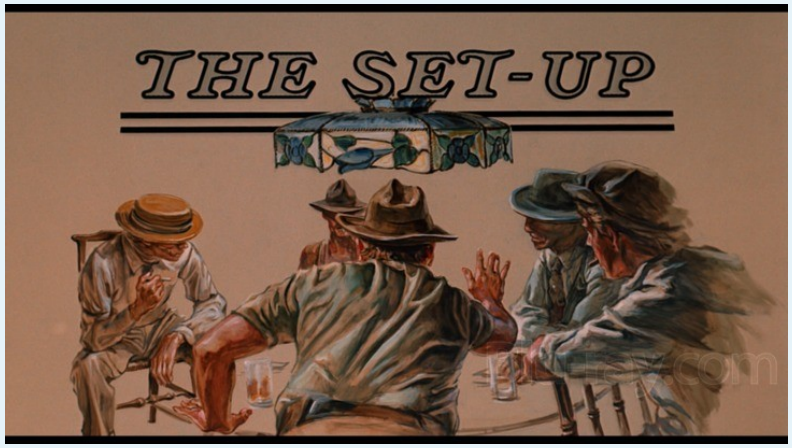
Lightweight Web Services with Pyramid

PyCon Canada 2012

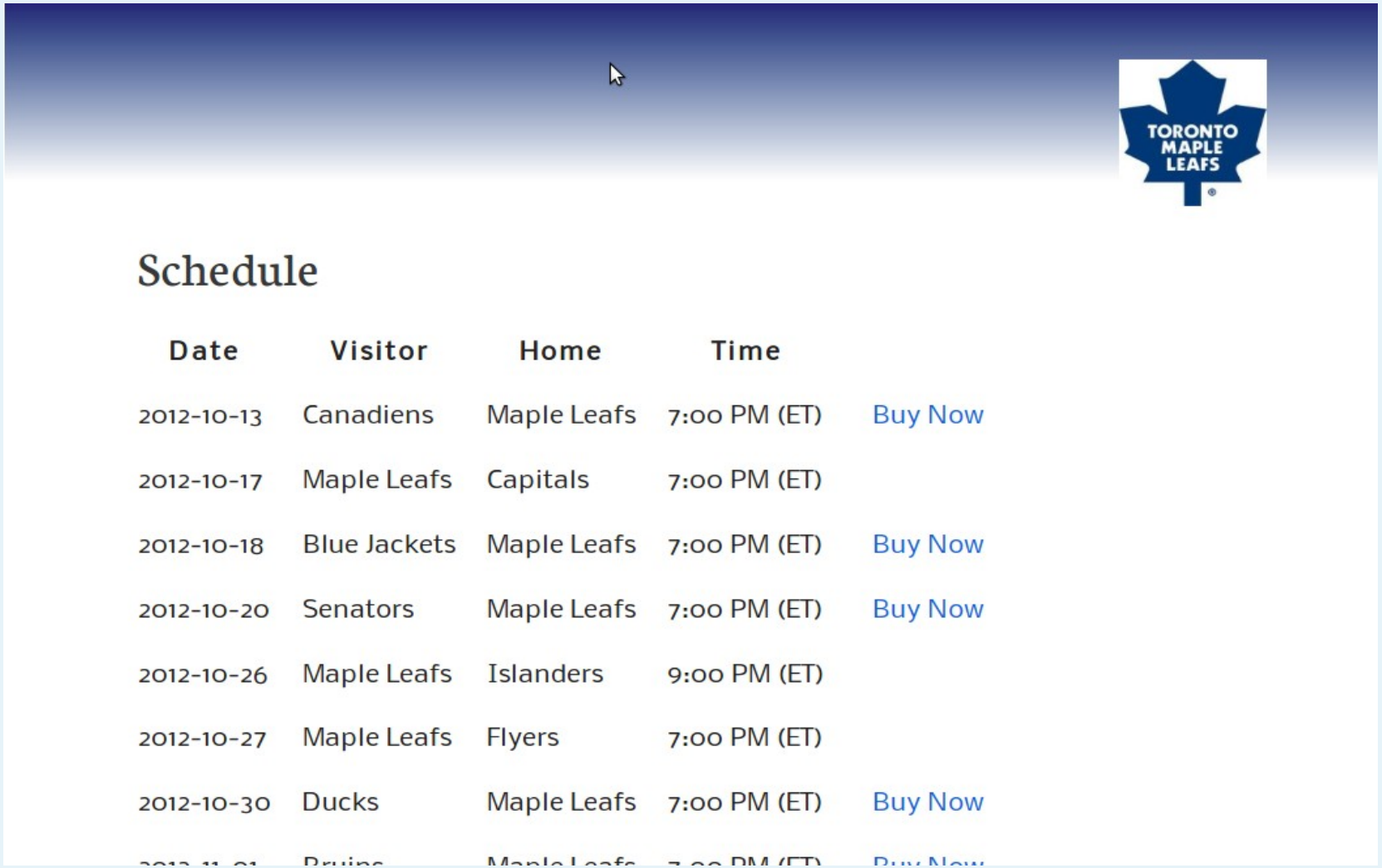
Tres Seaver
Agendaless Consulting
tseaver@agendalesscom
@tresseaver

The Setup



- Successful startup
- Web 1.0 app built w/ Pyramid

Web 1.0 Startup: Local Sports Tix



The Crisis

LOCKOUT!

Schedule

Date	Visitor	Home	Time
2012-10-13	Canadiens	Maple Leafs	Cancelled
2012-10-17	Maple Leafs	Capitals	Cancelled
2012-10-18	Blue Jackets	Maple Leafs	Cancelled
2012-10-20	Senators	Maple Leafs	Cancelled
2012-10-26	Maple Leafs	Islanders	Cancelled
2012-10-27	Maple Leafs	Flyers	Cancelled
2012-10-30	Ducks	Maple Leafs	Cancelled
2012-11-01	Bruins	Maple Leafs	Cancelled

The Angle

- Crisis or opportunity?
- Build a mobile version, FAST
- More revenue opportunities
- Pad your resumé ;)

The Sting

- Your site is built on Pyramid
- The back-end is already nearly done!
- Now you get to write the mobile bit, too

Pyramid

- Map HTTP requests to your code
- “View” is a function/callable which takes a request and returns a response

- ```
  def view(request):
      return Response(body='<h1>Hello, World!</h1>')
  ```
  ```

Renderer

- View can return data, used by “renderer” to generate response

- ```
  def view(request):
      return {'prompt': 'Hello, World!'}
  ```
  ```

Templates are Renderers

- Chameleon ex. →
- Pyramid supports Chamelon, Jinja2, and mako

```
<html>
<body>
<h1>${prompt}</h1>
</body>
</html>
```

Example view: Season Schedule

```
#views.py
SEASON = [('2012-10-13', True, 'Canadiens',
 'Maple Leafs', '7:00 PM (ET)'),#...
]

def _mapping(row):
    return {'date': row[0], 'avail': row[1],
            'visitor': row[2], 'home': row[3],
            'time': row[4]}

def schedule(request):
    return {'schedule':
             _mapping(game)for game in SEASON}
```

Example template

```
<html>
<body>
<table>
 <tr><th>Date></th><th>Visitor</th>...
 <tr tal:repeat=”game schedule”>
  <td>${game['date']</td>
  ...
 </tr>
</table>
</body>
</html>
```

Configuring the View

```
# views.py
from pyramid.view import view_config
@view_config(route_name='home',
             renderer='templates/schedule.pt')
def schedule(request):
    # as before
```

Configuring the Application

```
# main app config
from pyramid.config import Configurator

def main(global_config, **settings):
    config = Configurator(settings=settings)
    config.add_route('home', '/')
    config.scan()
    return config.make_wsgi_app()
```

Alternate renderer: JSON

- Instead of a template, we can pass a callable, or the name of a registered renderer
- Multiple views can be declared for a single view function

Configuring Another View

```
# views.py
from pyramid.view import view_config
@view_config(route_name='json',
             renderer='json')#registered name
@view_config(route_name='home',
             renderer='templates/schedule.pt')
def schedule(request):
    # as before
```

Add Another Route

```
# main app config
from pyramid.config import Configurator

def main(global_config, **settings):
    config = Configurator(settings=settings)
    config.add_route('home', '/')
    config.add_route('json', '/index.json')
    config.scan()
    return config.make_wsgi_app()
```

Profit!

```
$wget -o - http://example.com/index.json
{"schedule": [{"date": "2012-10-13",
"available": true, "time": "7:00 PM (ET)",
"home": "Maple Leafs", "visitor":
"Canadiens"}, ...
```

Cornice

- Mozilla framework atop Pyramid
- Validated schemas for RESTish web services
- Support for unified collection / item handling
- Auto-generated API docs

Cornice Example

```
#views.py
from cornice.resource import resource

@resource(collection_path='/members/',
          path='/members/{m_id}')
class MemberProfiles(object):
    def __init__(self, request):
        self.request = request
    def collection_get(self):
        return _members
    def get(self):
        m_id = self.request.matchdict['m_id']
        return members[m_id]1
```

Links / Q&A

- Pyramid
  http://www.pylonsproject.org/
- Cornice
  https://github.com/mozilla-services/cornice