Pixaxe - A Declarative Web Application Framework

Rob King

DVLabs
TippingPoint Technologies

June 24, 2010

Introducing Pixaxe

Pixaxe is a web application framework with several design goals:

Introducing Pixaxe

Pixaxe is a web application framework with several design goals:

Declarative Pixaxe encourages a very declarative, functional style of interface and logic specification that centers around the evaluation of functions.

Introducing Pixaxe

Pixaxe is a web application framework with several design goals:

Declarative Pixaxe encourages a very declarative, functional style of interface and logic specification that centers around the evaluation of functions.

Client-Focused Pixaxe runs entirely on the client, including functionality that often resides on the server.

Introducing Pixaxe

Pixaxe is a web application framework with several design goals:

Declarative Pixaxe encourages a very declarative, functional style of interface and logic specification that centers around the evaluation of functions.

Client-Focused Pixaxe runs entirely on the client, including functionality that often resides on the server.

Lightweight Pixaxe transmits a page only once; afterwards, only changes to the data model are sent. The server need only support static file serving and, optionally, JSON (de)serialization.

Introducing Pixaxe

Pixaxe is a web application framework with several design goals:

Declarative Pixaxe encourages a very declarative, functional style of interface and logic specification that centers around the evaluation of functions.

Client-Focused Pixaxe runs entirely on the client, including functionality that often resides on the server.

Lightweight Pixaxe transmits a page only once; afterwards, only changes to the data model are sent. The server need only support static file serving and, optionally, JSON (de)serialization.

Simple Pixaxe is designed as a lightweight layer that augments HTML and related technologies, rather than replace them.

Introducing Pixaxe

Pixaxe is a web application framework with several design goals:

Declarative Pixaxe encourages a very declarative, functional style of interface and logic specification that centers around the evaluation of functions.

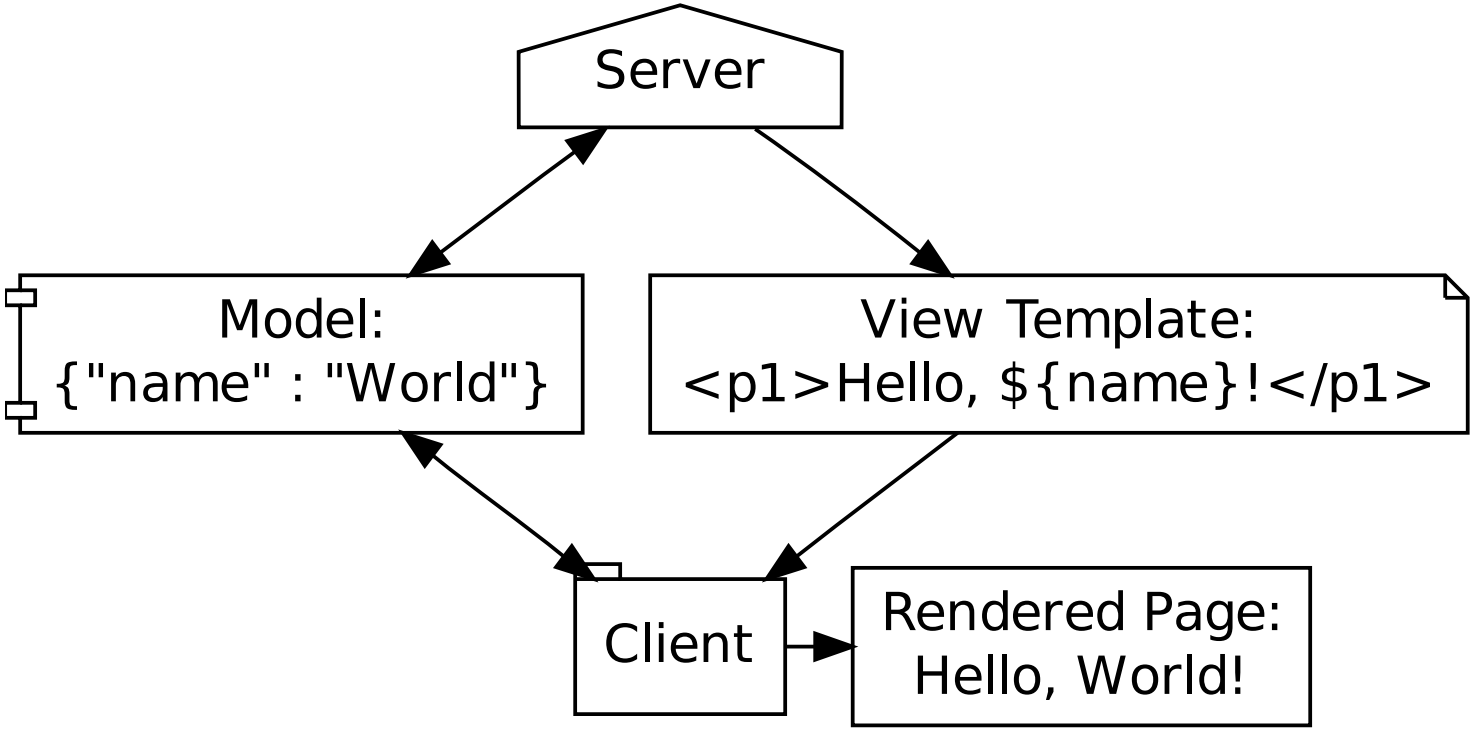
Client-Focused Pixaxe runs entirely on the client, including functionality that often resides on the server.

Lightweight Pixaxe transmits a page only once; afterwards, only changes to the data model are sent. The server need only support static file serving and, optionally, JSON (de)serialization.

Simple Pixaxe is designed as a lightweight layer that augments HTML and related technologies, rather than replace them.

Decoupled Pixaxe is built up of several distinct technologies, and these technologies can be used individually as required.

The Anatomy of a Pixaxe Application



Model-View-Controller

Pixaxe follows a traditional Model-View-Controller design:

- The Model is viewed as a single JSON document that is synchronized among the server, browser, interface, and user. Pixaxe automatically synchronizes the model when necessary.
- The View is a single XHTML document with embedded expressions in Pixaxe's template language. This view is automatically re-rendered in response to changes in the model.
- The Controller exists entirely within the client, and relationships between the controller and the model are also specified declaratively.

Specifying Views

Web Pages as Expressions

- The technology used for the specification of views is known as Jenner, and is usable independently of the rest of Pixaxe.
- Jenner is an expression language; all XHTML pages are valid Jenner expressions, but Jenner is a superset of XHTML. Rendering a page is identical to evaluating the Jenner expression.
- No explicit calls need to be made to render a Jenner expression; the web page itself is considered the Jenner source code and it is reevaluated whenever necessary.

A Simple Jenner Expression

- The Model, a JSON document:

```
{"name" : "world"}
```

- The View, a Jenner expression:

```
<head>
    <title>Example</title>
    <script lang="text/javascript"
            src="pixaxe.js" />
    <script lang="text/javascript"
            src="model.js" />
</head>
<body>
    <p1>Hello, ${name}!</p1>
</body>
```

- The Result, a rendered web page:

```
Hello, world!
```

A More Complicated Jenner Expression

The Model

```
{
    "log" : [
        {"level" : "CRIT"
         "text"  : "Core temp critical"},
        {"level' : "INFO",
         "text"  : "Fries are done"}
    ]
}
```

A More Complicated Jenner Expression

The View

```
<head>
    <title>Log Messages</title>
    <script lang="text/javascript" src="pixaxe.js" />
    <script lang="text/javascript" src="model.js" />
</head>
<body>
    <ul>
        ${for i from 0 to log.length - 1
            var m := log[i]
            return
            <li class="${m.level == 'CRIT' ?
                        'log-red' : 'log-black'}">
                ${m.text}
            </li>
        }
     </ul>
</body>
```

Jenner

Language Features

- Standard operators, including modular arithmetic, Boolean combination, etc.
- List comprehensions.
- Lexical scoping (the Model serves as the root environment).
- Document elements as expressions and results.
- Attribute values may be Jenner expressions.
- Functions and a foreign function interface with JavaScript.
- Can be used separately from Pixaxe as a powerful client-side template engine.
- Can use XSLT as a page-load-time macro language.

Jenner

Example Macro Usage

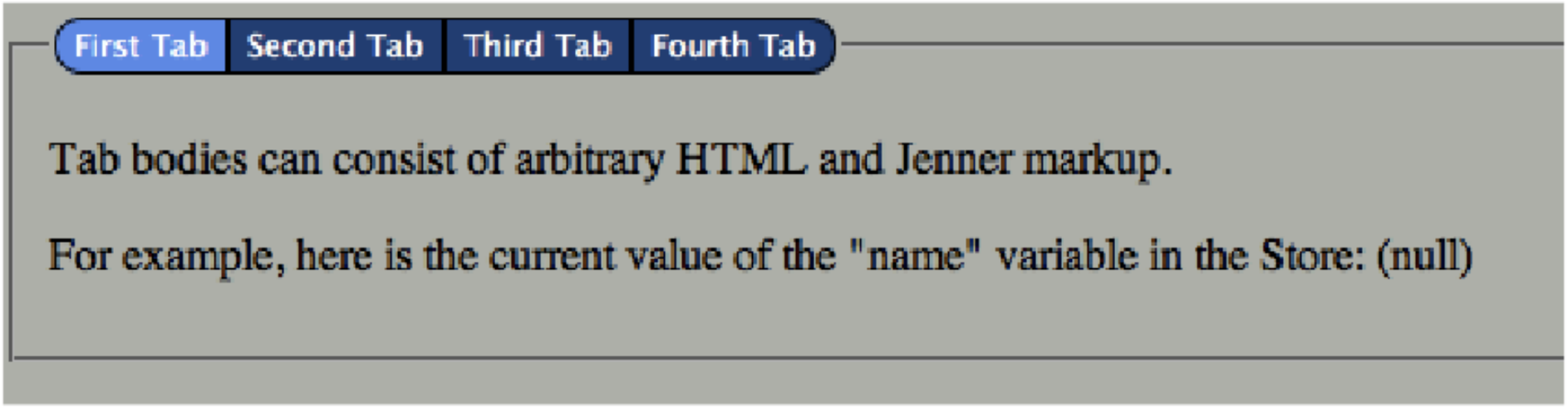
- Jenner provides a richer “target language” for browser XSL transformations than plain XHTML.
- Pixaxe provides a variety of XSLT macros, for things like tab boxes, lightboxes, and AJAX-style file uploads.
- Below is a page fragment using the standard “tab-box” macro.

```
<dppx:tab-box>
 <dppx:tab label="First Tab" selected="true">
  <p>Tab bodies can consist of arbitrary HTML and
     Jenner markup.</p>
  <p>For example, here is the current value
     of the "name" variable in the Store: ${name}</p>
 </dppx:tab>
 <dppx:tab label="Second Tab">
  <p>Another tab.</p>
 </dppx:tab>
    ...
```

Below is a partial expansion of the macro:

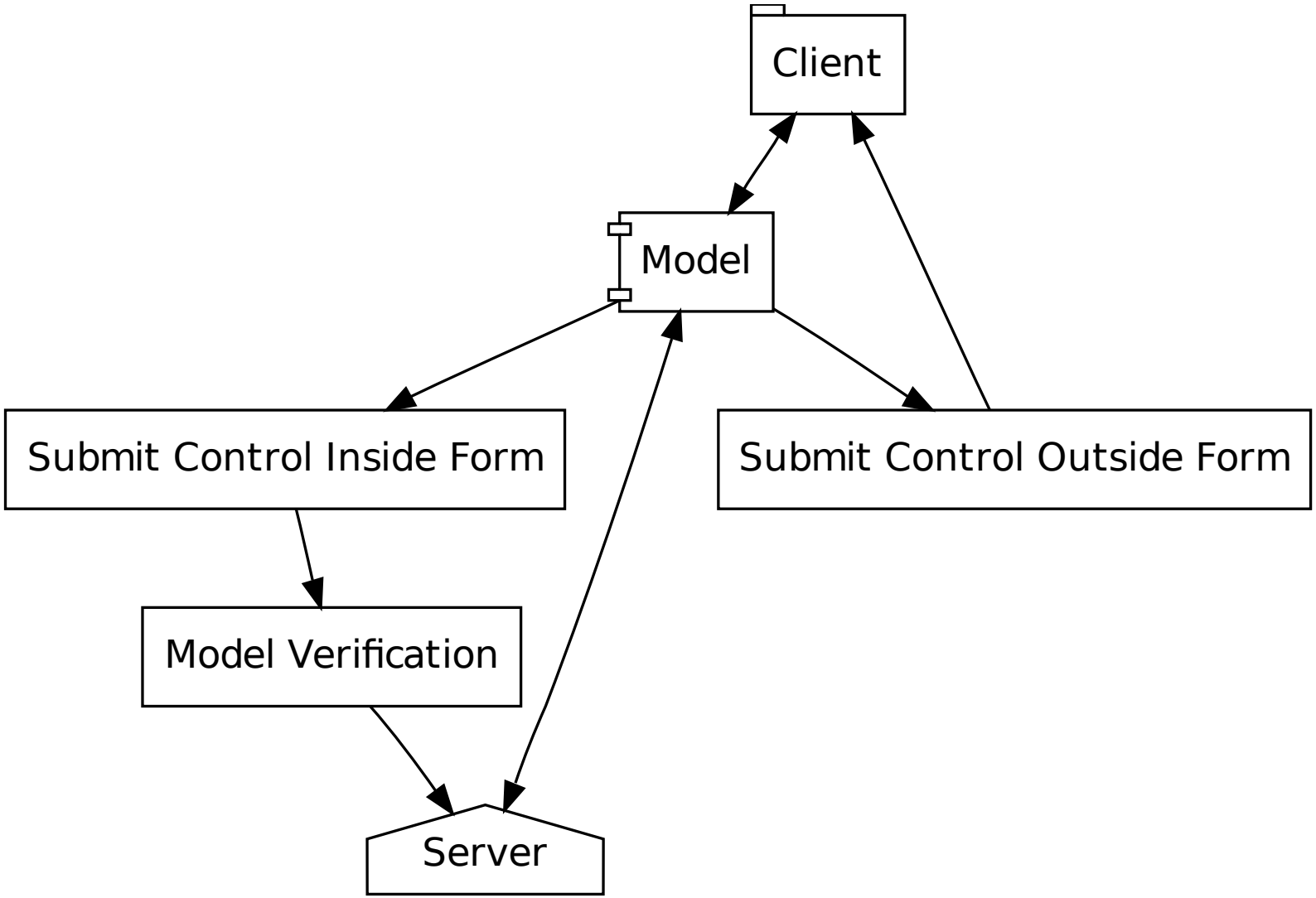
```
<fieldset><input type="hidden" value="id4127134"
name="#{controller.dppx_tabselid4127132}"/>
<legend><input type="submit"
name="#{controller.dppx_tabselid4127132}"
class="dppx-tab dppx-tab-left
dppx-tab-${controller.dppx_tabselid4127132
!= 'id4127134' ?
'un' : ''}selected"
accept="id4127134" value="First Tab" />
...
dppx-tab-body-${controller.dppx_tabselid4127132
!= 'id4127134' ? 'un' : ''}selected">
<p>Tab bodies can consist of arbitrary HTML and
   Jenner markup.</p>
<p>For example, here is the current value of the "name"
variable in the Store: ${name}</p>
</div>
```

The above macro as rendered by the browser:



Pixaxe

Adding Input Handling to Jenner



- Pixaxe uses traditional HTML form controls for input, but augments some of their functionality.
- By placing expressions inside the `name` and `value` attributes of controls, controls can be linked to values in the model.
- For example, below is an `input` element that is linked to the `person.name` member of the model:

```
<input name="#{person.name}" value="${person.name}" />
```

- By placing an expression in the `accept` attribute of a control, that control's value can be validated or manipulated before being applied to the model.
- For example, the input control below can be set to automatically uppercase the first letter of its input before it is synchronized with the model:

```
<input name="#{person.name}" value="${person.name}"
accept="${str:titlecase($value)}" />
```

Pixaxe

Synchronizing the Model With the Server

- Pixaxe overloads the meaning of HTML `form` elements.
- Submit controls inside form elements result in the synchronization of the model with the server, if the form's `enctype` attribute is set to `text/javascript`.
- Forms can also validate the model before it is synchronized with the server by evaluating an expression placed in the form's `accept` attribute; this expression must evaluate to true for the model to be synchronized.
- It is important to note that this is not HTML form submission; only the model is serialized and then synchronized; the page is re-rendered entirely locally with the freshly synchronized model.

Pixaxe

Interacting With JavaScript

- Pixaxe allows for a simple foreign function interface with JavaScript code in the browser.
- Jenner expressions can call JavaScript functions exported to the Jenner runtime, and JavaScript code can easily interact with Pixaxe - either indirectly by manipulating the model, or directly by explicitly evaluating Jenner expressions.
- Jenner comes with a large standard library of functions for a variety of tasks, including text manipulation, cookie storage, mathematical functions, alert boxes, and so on.
- Functions can be namespaced similarly to XML Namespaces, preventing collisions.

Underlying Technology

Kouprey

- Pixaxe is built on top of a complete parser combinator library written in pure JavaScript, called Kouprey.
- Kouprey is usable separately from Pixaxe, and provides a useful tool for the development of parsers running in web browsers and other JavaScript environments.
- Grammars are specified inline using normal JavaScript statements, but in such a way as to resemble EBNF.

Underlying Technology

Esel

- Esel is a powerful expression and query language, useful for querying JSON and other hierarchical datasets.
- Esel is a perfect subset of Jenner; Jenner is in fact Esel with the addition of XML element types and syntax.
- Esel's parser, compiler, and virtual machine are written entirely in JavaScript and run entirely within the browser.
- Esel's virtual machine is Turing complete and easily extensible, with an efficient code representation.
- Esel is useful as an embedded expression language in web applications, for example, in a web-based spreadsheet.

Potential Uses

- Pixaxe is useful as an application toolkit for small and rapidly-developed web applications.
- It also proves useful in situations where server resources are extremely limited or not under the control of the application developer.
- Pixaxe is useful in situations where the developer is more familiar with HTML and CSS than with JavaScript or programming.
- Pixaxe, with it's traditional form- and page-oriented design and extremely flexible server interface, may be useful for developing web based interfaces to legacy applications.
- Kouprey greatly eases the creation of parsers in web browsers, easing the development of application-specific languages in web applications (think expressions in web-based spreadsheets, for example).

The Future

- Kouprey 2 is already finished, and has proven to be considerably faster, with a simpler grammar specification syntax. It is currently just awaiting documentation.
- Esel 2 is nearing completion, and adds useful features such as nested comments, an *n*-way case statement, optional assignment, and improved facilities for runtime analysis of expressions.
- Pixaxe 2 is being planned on top of these features, and will have a much more “reactive programming” feel.

Conclusion

- Pixaxe is available for use today, under the GNU General Public License.
- Pixaxe is still under development and has some bugs, but it has been used in production environments for relatively large and complex projects.
- For more information, please contact Rob King at either:
  - `http://www.deadpixi.com`
  - jking@deadpixi.com
  - rob.r.king@hp.com