Zend_Apjax Proposal - Steven Brown

<ac:macro ac:name="note"><ac:parameter ac:name="title">Under Construction</ac:parameter><ac:rich-text-body>
This proposal is under construction and is not ready for review. </ac:rich-text-body></ac:macro>

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Apjax Component Proposal

Proposed Component Name	Zend_Apjax
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Apjax
Proposers	Steven Brown
Revision	0.1 - 8 February 2008: Created. (wiki revision: 4)

Table of Contents

1. Overview

Zend_Apjax is proposed to act in a similar fashion to the Aptana Jaxer system. Apjax stands for Asynchronous PHP, Javascript And Xml.

Apjax will allow developers to call javascript functions from the server side, or to call server side functions from javascript using AJAX.

2. References

- Aptana Jaxer

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** allow developers to call server-side PHP "functions" from
- This component **will** allow server-side functions called using Apjax to in turn call client-side Javascript functions
- This component **will not** run server-side Javascript

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Json

5. Theory of Operation

- The developer will set up classes/methods and register these in the "functions" list
- Zend_Apjax outputs Javascript code with the view that creates a Javascript object that provides access to the server-side functions
- The developer calls methods of the Javascript object using Javascript code where necessary
- The developer can call client-side Javascript functions from within any registered server-side "functions"
- Zend_Apjax will capture responses that require further client-side function calls, or can detect and return the result of a function
- Zend_Apjax responses can include Javascript code to be executed on the client side, this code would be in a string and as a result could be loaded from a file

Some potential limitations and issues exist:

- While there is a seamless communication channel, the speed of communication can vary and may degrade the user experience if not handled appropriately
- Circular function calls could be created by the developer that never resolve
- Capturing and handling errors could be difficult and may break code execution

6. Milestones / Tasks

- Milestone 1: Design notes completed
- Milestone 2: Working prototype checked into the incubator allowing client to call server-side functions
- Milestone 3: Working prototype checked into the incubator allowing server-side functions to call client-side functions
- Milestone 4: Working prototype checked into the incubator allowing server-side functions to call custom client-side code
- Milestone 5: Unit tests exist, work, and are checked into SVN.
- Milestone 6: Initial documentation exists.

7. Class Index

- Zend_Apjax_Exception
- Zend_Apjax
- Zend_Apjax_Function
- Zend_Apjax_Client

8. Use Cases

Registering a sever-side "function":

```
Zend_Apjax::register('functionName', 'someClass', 'someMethod', $argumentNames);
Zend_Apjax::register('functionName', new Zend_Apjax_Function('someClass',
'someMethod', $argumentNames));
```

Calling a client-side function from a server-side function:

```
Zend_Apjax_Client::call('someFunction', $arguments);
```

Client-side Javascript calling a sever-side function:

```
Zend_Apjax::someFunctionName(argument1, argument2...);
```

9. Class Skeletons

```
class Zend_Apjax_Exception extends Zend_Exception {}

class Zend_Apjax {}

class Zend_Apjax_Function {}

class Zend_Apjax_Client {}
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>