Mach-II Request, Event, and Exception Handling

For Mach-II for PHP version 1.0

Mach-II is a web-application framework that implements an event-based, implicit invocation architecture style. The framework's job is to handle web-requests, package request information into events, and process the events in a controlled manner.

Request Handling

Request handling in Mach-II is primarily handled by an instance of MachII_framework_RequestHandler. Each Mach-II application has one instance of a MachII_framework_RequestHandler that is created when the application is initialized. Usually, it will be initialized via the static method, MachII::cache().

An application's MachII_framework_RequestHandler is the entry point to the application controller. It is used to translate a web-request to an event that can be handled by the framework. Let's examine a typical request to Mach-II (in the form of a URL) to see how it is handled.

A simple example URL for Mach-II:
../index.php?event=doSomething&param=value1

../**index.php**?event=doSomething&param=value1
The index.php file in the URL represents the entry point into a Mach-II application. The index.php file could be any file that includes mach-ii.php, a file distributed with the framework code. This entry point file will begin the Mach-II request handling process by delegating the request to the application's MachII_framework_RequestHandler.

../index.php?**event=doSomething**&param=value1
The event parameter (which can also be passed via a form or URL parameter) tells the framework which event to handle. The parameter name (event) can be altered via the Mach-II config file (typically named mach-ii.xml) by setting the *eventParameter* property.

Example from a *mach-ii.xml* file:

```
<mach-ii>
     <properties>
          <!-- For this application, the request parameter 'event'
defines the event to handle. -->
          <property name="eventParameter" value="event" />
          ...
     <properties>
     ...
</mach-ii>
```

../index.php?event=doSomething&**param=value1**

Any parameters in the form or URL (_POST or _GET) arrays are copied to a framework array called *eventArgs*.

When the Mach-II framework handles a request, all request parameters (form and URL variables) are copied to a framework structure called *eventArgs*. If both a form and URL parameter have the same key, the precedence of the scope defined in the *parameterPrecedence* property of the Mach-II config file (mach-ii.xml) properties section will take precedence.

Example from a *mach-ii.xml* file:

```
<mach-ii>
     <properties>
          <!-- For this application, the form scope takes precedence
over the url scope. -->
          <property name="parameterPrecedence" value="form" />
          ...
     <properties>
     ...
</mach-ii>
```

The first event handled for the request is determined by getting the parameter from the eventArgs structure that matches the mach-ii.xml property defined in *eventParameter*. If the *eventParameter* property is not specified in mach-ii.xml then 'event' is used by default.

If an event was not specified in the request parameters then a default event will be announced. The *defaultEvent* property defined in the Mach-II config file (mach-ii.xml) specifies which event to handle by default.

Example from a *mach-ii.xml* file:

```
<mach-ii>
     <properties>
          <!-- For this application, if an event is not specified in
the request handle the 'showHome' event. -->
          <property name="defaultEvent" value="showHome" />
          ...
     <properties>
     ...
</mach-ii>
```

The following URL *http://{host}/index.php?event=registerUser&firstName=John&lastName=Smith&emailAddress=johnsmith@mach-ii.com* will result in the following event:

Event: registerUser	
firstName	John
lastName	Smith
emailAddress	johnsmith@mach-ii.com

Once the first event for Mach-II to handle is determined, and the request parameters are encapsulated in the event, the event will be announced to the framework.

Event Handling

In a Mach-II system, all framework actions are driven by events. Events represent an action and encapsulate the information necessary to perform that action. Each event is an instance of an MachII_framework_Event object. For each request, all events are handled sequentially by a queue.

For each request handled by Mach-II a MachII_framework_EventContext object will be created. The job of each MachII_framework_EventContext instance is to handle the event-queue mechanism for a request. All events for a single request will be handled by the same event-context instance.

Events are queued in the event-context by calling:
```
$eventContext->announceEvent( string eventName, [array eventArgs] )
```

The current Event being handled by the event-context can be accessed by calling:
```
$eventContext->getCurrentEvent()
```

After the first event is announced to event-context, the event-queue has one event to be handled. While handling any event, more events can be announced to the event-context where they will be placed at the end of the queue. The event-context will continue processing each event in the queue, in order, until there are no more events to be handled.

Determine if there are any more events in the queue by calling:
```
$eventContext->hasMoreEvents()
```

At any time the event-context's event queue can be cleared by calling:
```
$eventContext->clearEventQueue()
```

While there are events in the queue they are handled one at a time. To handle each event Mach-II will look up an event-handler in the mach-ii.xml config file based on the name of the event. The event-handler XML will define commands to execute to handle the event.

Each event-command will be executed one at a time and in order. Event-handler XML and event-commands are explained in detail in other documents, including the Mach-II Configuration guide.

Example from a *mach-ii.xml* file:
```
<mach-ii>
      ...
      <event-handlers>
            <event-handler name="eventName" access="public">
                  <!-- Define commands here. -->
                  ...
            </event-handler>
```

```
      </event-handlers>
      ...
</mach-ii>
```

A property can be set in the mach-ii.xml configuration file to limit the maximum number of events that can be processed for a single request. A typical use of this property is as a safeguard to ensure an infinite loop doesn't occur (where an event directly or indirectly announces itself). The *maxEvents* property defined in the Mach-II config file (mach-ii.xml) specifies the maximum number of events to handle per request.

Example from a *mach-ii.xml* file:

```
<mach-ii>
      <properties>
            <!-- For each request, no more than 10 events may be
processed. -->
            <property name="maxEvents" value="10" />
            ...
      <properties>
      ...
</mach-ii>
```

If the maximum number of events is exceeded an exception will be thrown. The exception will be handled by Mach-II in the manner described in the next section.

Note: This is a framework exception not a PHP5 exception!

Exception Handling

Exception handling in Mach-II makes use of the same event model explained in the previous section. Exceptions occurring in business logic should typically be caught and handled before reaching the framework level. However, the framework itself will throw some exceptions directly.

Some exceptions that may occur at the framework level:

- Attempting to announce an event from a web-request that is not publicly accessible.
- Attempting to announce an event without a defined event-handler.

Whether an exception is thrown by the framework itself or caught as a result of business logic error, it will be handled in the same fashion. When an exception is caught by Mach-II, the following occurs:

1. The exception information is packaged into an instance of MachII_util_Exception.
2. A new event is created with its name specified by the exceptionEvent property.
3. The exception object is placed in the new event's args with key 'exception'.
4. If the exception occurred while handling an event the exception causing event is placed in the new event's args with key 'exceptionEvent'.
5. The event-context is cleared of all queued events.
6. The exception event is announced.
7. The exception event is handled like any other event.

The *exceptionEvent* property defined in the Mach-II config file (mach-ii.xml) defines the event to announce when the framework catches an exception.

Example from a *mach-ii.xml* file:

```
<mach-ii>
     <properties>
          <!-- If an exception is caught by the framework, package
the exception and announce the 'exception' event. -->
          <property name="exceptionEvent" value="exception" />
          ...
     <properties>
     ...

     <event-handlers>
          <event-handler name="exception" access="private">
               <!-- Handle an exception here. -->
               ...
          </event-handler>
     </event-handlers>
</mach-ii>
```