介绍 CouchDB 的 couch_event_sup

1. erlang 的 error_logger & gen_event

erlang 的 error_logger 模块是一个 gen_event，它默认实现了一个日志处理的 handler，我们也可以自定义自己的日志处理 handler，实现日志不同格式的输出，或者把日志同步输入到文件中等.

例如(下面代码实现一个简单的 gen_event handler 模块，它将错误日志按照自定义格式输出):

```
-module(my_log).
-behaviour(gen_event).
%% gen_event callbacks
-export([init/1, handle_event/2, handle_call/2,
         handle_info/2, terminate/2, code_change/3]).
init(Args) ->
    io:format("***my_log init***: ~p~n", [Args]),
    {ok, []}.
handle_event(ErrorMsg, State) ->
    io:format("***my_log error***: ~p~n", [ErrorMsg]),
    {ok, State}.
handle_call(_Request, State) ->
    Reply = ok,
    {ok, Reply, State}.
handle_info(_Info, State) ->
    {ok, State}.
terminate(Reason, _State) ->
    io:format("***my_log terminate***: ~p~n", [Reason]),
    ok.
code_change(_OldVsn, State, _Extra) ->
    {ok, State}.
```

测试:

```
error_logger:error_msg("test msg ~n", []).    %% error_logger 的默认行为
=ERROR REPORT==== 16-Aug-2010::10:42:38 ===
test msg
ok
gen_event:add_handler(error_logger, my_log, "init args"). %% 添加自定义的 handler
***my_log init***: "init args"
ok
error_logger:error_msg("test msg ~n", []).   %% 会输出自定义的 error message
***my_log error***: {error,<0.26.0>,{<0.37.0>,"test msg ~n",[]}}
=ERROR REPORT==== 16-Aug-2010::10:43:05 ===
test msg
ok
gen_event:delete_handler(error_logger, my_log, "delete reason"). %% 删除自定义的 handler
```

```
***my_log terminate***: "delete reason"
ok
error_logger:error_msg("test msg ~n", []).       %% 回复 error_logger 的默认行为
=ERROR REPORT==== 16-Aug-2010::10:43:31 ===
test msg
ok
```

注意:
我们使用这两个 APIs 来添加或者删除 handler

```
gen_event:add_handler(EventMgrRef, Handler, Args) -> Result
gen_event:delete_handler(EventMgrRef, Handler, Args) -> Result
```

同调用下面两个 error_logger 模块封装好的 APIs 效果是一样的:

```
error_logger:add_report_handler(Handler, Args) -> Result
error_logger:delete_report_handler(Handler) -> Result
```

2. gen_event:add_handler/3 和 gen_event:add_sup_handler/3 的区别

```
<1> add_handler(EventMgrRef, Handler, Args) -> Result
```

添加一个新的 handler 到 event manager，将会调用 Module:init 来初始化这个新的 handler.

```
<2> add_sup_handler(EventMgrRef, Handler, Args) -> Result
```

同 add_handler/3 功能类似，也是添加一个新的 handler 到 event manager，并且调用 Module:init 来初始化新的 handler，不同点是会监控 event handler 和调用进程(call process)之间的 connection.

a. 如果调用进程(call process) terminates with Reason，则 event manager 将通过调用

Module:terminate/2，并且以{stop，Reason}作为参数来删除这个 handler.

b. 如果新的 handler 被删除，则 event manager 将发送{gen_event_EXIT,Handler,Reason}消息到调用进程(call process).

3. couch_event_sup 模块:
这个模块实质上的作用是把 gen_event handler 和它的调用进程(call process)结合起来，我们可以通过调用:

```
{ok, CallProcessPid} = couch_event_sup:start_link(error_logger, my_log, Args)
```

来代替调用

```
ok = gen_event:add_sup_handler(error_logger, my_log, Args)
```

在调用 couch_event_sup:start_link/3 之后，会启动一个 gen_server 进程，这个新的进程就是 gen_event handler my_log 对应的调用进程(call process)，通过这种方式把 gen_event handler 和它对应的 call process 结合起来，这样 gen_event handler 就可以作为进程树的一部分像

gen_server 一样被监控，可以被随意的 start, restart, shutdown.

例如:
如果 event handler 停止，则对应的 gen_server 由于 gen_event_EXIT 消息也会停止.
停止 gen_server，则 event handler 对于对应的 calll process 停止也会被删除.
启动 gen_server, event handler 会被添加.

具体实现代码:

```
-module(couch_event_sup).
-behaviour(gen_server).
-include("couch_db.hrl").
-export([start_link/3,start_link/4, stop/1]).
-export([init/1, terminate/2, handle_call/3, handle_cast/2, handle_info/2,code_change/3]).

start_link(EventMgr, EventHandler, Args) ->
    gen_server:start_link(couch_event_sup, {EventMgr, EventHandler, Args}, []).

start_link(ServerName, EventMgr, EventHandler, Args) ->
    gen_server:start_link(ServerName, couch_event_sup, {EventMgr, EventHandler, Args}, []).

stop(Pid) ->
    gen_server:cast(Pid, stop).

init({EventMgr, EventHandler, Args}) ->
    ok = gen_event:add_sup_handler(EventMgr, EventHandler, Args),
    {ok, {EventMgr, EventHandler}}.

terminate(_Reason, _State) ->
    ok.

handle_call(_Whatever, _From, State) ->
    {ok, State}.

handle_cast(stop, State) ->
    {stop, normal, State}.

handle_info({gen_event_EXIT, _Handler, Reason}, State) ->
    {stop, Reason, State}.

code_change(_OldVsn, State, _Extra) ->
    {ok, State}.
```