player\_dict：玩家进程字典；

跨服服务端API：

%% @doc 发消息给游戏服  
%% 返回：undefined-失败，ok-成功  
send\_to\_game(Bin) -> %%Bin为发送数据  
 GameList = mod\_cross\_server:get\_game\_list(),  
 Fun = **fun**(#ets\_cross\_server{send\_pid = SendPid}) ->  
 cross\_server:send\_to\_sid(SendPid, Bin)  
 **end**,  
 lists:foreach(Fun,GameList)**.**send\_to\_game(ServerId, Bin) -> %%ServerId为发送进程  
 **case** mod\_cross\_server:get\_send\_pid(ServerId) **of** undefined ->  
 ?ERROR\_MSG("Not send\_pid:~p", [{ServerId,Bin}]),  
 undefined;  
 SendPid ->  
 cross\_server:send\_to\_sid(SendPid, Bin),  
 ok  
 **end.**

%% 玩家record定义

-record(r\_player\_info, {  
 player\_id = 0 % 玩家id  
 ,user\_id = 0 % 帐号id  
 ,user\_name = "" % 帐号名  
 ,name = "" % 角色名  
 ,union\_id = 0 % 军团ID  
 ,sex = 0 % 性别  
 ,exp = 0 % 经验值  
 ,level = 1 % 等级  
 ,avatar\_id = 0 % 头像  
 ,fight = 0 % 战斗力  
 ,create\_time = 0 % 创建时间  
 ,reset\_time = 0 % 重置时间  
 ,offline\_time = 0 % 下线时间  
 ,last\_login\_time = 0 % 最后一次登录时间  
 ,ip = "" % 用户ip  
 ,heat\_error = 0 % 心跳错误次数(跨天重置为0)  
 ,state = 0 % 状态  
 ,ban\_chat\_date=0 % 禁言时间  
 ,login\_day = 0 % 登录天数  
 ,vip\_lev = 0 % vip等级  
 ,channel\_id = "" % 渠道id  
 ,is\_merge\_first\_login = 1 % 是否是合服第一次登陆 -- 用来做一些合服处理 1 是 0 否  
 ,platform = 0 % 0=其他 1=android 2=ios  
})**.**

**Union\_id 军团Id；**

**增加军团的经验值函数：**

add\_union\_exp(AddExp) ->  
 #r\_player\_info{union\_id = UnionId} = player\_dict:get\_player\_info(),  
 ?IF(UnionId == 0, skip, mod\_union:cast({add\_union\_exp, misc:get\_player\_sid(?PLAYER\_ID), UnionId, AddExp})),  
 ok**.**

**mod\_cache.erl**

%% 保存标识  
-define(SAVE\_FLAG\_KEY(PlayerId), {?MODULE, save\_flag, PlayerId})**.**get\_save\_flag(PlayerId) -> erlang:get(?SAVE\_FLAG\_KEY(PlayerId))**.**set\_save\_flag(PlayerId, Flag) -> erlang:put(?SAVE\_FLAG\_KEY(PlayerId), Flag)**.**del\_save\_flag(PlayerId) -> erlang:erase(?SAVE\_FLAG\_KEY(PlayerId))**.**

do\_cast({clear\_offline\_player, IsForce}, #state{ets\_name = EtsName} = State) ->  
 erlang:statistics(wall\_clock),  
  
 NowSec = misc\_timer:now\_seconds(),  
 ClearPlayers = ets:foldl(  
 **fun**(EtsPlayer, Acc) ->  
 **case** EtsPlayer#ets\_player**.**offline\_time **of** 0 -> Acc;% 玩家在线  
 OfflineTime ->

%% 未到清除时间并且没有进行强制清除  
 **case** (NowSec - OfflineTime) > ?CACHE\_CLEAR\_TIME **of** false **when** IsForce =/= 1 ->  
 % 还没到清除时间  
 Acc;  
 \_ ->  
 PlayerId = EtsPlayer#ets\_player**.**player\_id,  
 % 清除时存档  
 **case** get\_save\_flag(PlayerId) **of** true -> save\_cache\_player(EtsName, EtsPlayer);  
 \_ ->  
 skip  
 **end**,  
  
 ets:delete(EtsName, PlayerId),  
 [PlayerId | Acc]  
 **end  
 end  
 end**, [], EtsName),

**模块说明：**

**玩家缓存模块：**

**mod\_cache 玩家缓存操作函数模块；**

**api\_cache 玩家缓存接口模块，通过接口对数据库进行相应的持久化操作（使用数据库代理db\_agent实现）；**

**项目包说明：**

**db\_agent:游戏数据库代理模块；**

**mod\_player/lib ,lib\_xxx用于实现具体的算法和也无逻辑；**