# 개발 사항

* 2가지 쿠폰 종류 개발 (public, scratch)
* 유저 메일을 통한 보상 지급
* db에서 보상 리스트 관리
  + 추후 운영툴에서 관리를 위해

# 쿠폰의 종류

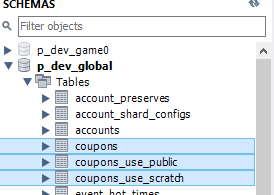
* public 쿠폰
  1. 유투브에 댓글이나 게임 이벤트 시, 공개한 쿠폰처럼

한 유저가 한번씩만 사용이 가능한 쿠폰

* scratch 쿠폰
  1. 한번 긁으면 사라지는 쿠폰

| public class BaseCoupon  {  */// <summary>*  */// 이미 사용한 쿠폰인지 체크*  */// </summary>*  */// <param name="db"></param>*  */// <param name="couponId"></param>*  */// <param name="playerId"></param>*  */// <returns></returns>*  public virtual bool CheckdAlreadyUsedCoupon(DatabaseSystem db, string couponId, long playerId)  {  return true;  }  */// <summary>*  */// 쿠폰 사용*  */// </summary>*  */// <param name="db"></param>*  */// <param name="couponId"></param>*  */// <param name="playerId"></param>*  */// <returns></returns>*  public virtual int Used(DatabaseSystem db, string couponId, long playerId)  {  return 0;  }  } |
| --- |

# DB



| CREATE TABLE `coupons` (  `id` bigint NOT NULL AUTO\_INCREMENT COMMENT '시퀸스',  `coupon\_id` varchar(12) NOT NULL COMMENT '쿠폰 번호',  `use\_type` int NOT NULL COMMENT '타입',  `expired\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '만료일',  `created\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '생성일',  PRIMARY KEY (`id`),  UNIQUE KEY `uidx\_coupon\_id` (`coupon\_id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=2713 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci COMMENT='쿠폰'; |
| --- |

| public interface ICoupon  {  */// <summary>*  */// 쿠폰이 유효한지 체크*  */// </summary>*  */// <param name="db"></param>*  */// <param name="couponId"></param>*  */// <param name="playerId"></param>*  */// <returns></returns>*  static abstract bool Valided(DatabaseSystem db, string couponId, long playerId);  */// <summary>*  */// 쿠폰 사용*  */// </summary>*  */// <param name="db"></param>*  */// <param name="couponId"></param>*  */// <param name="playerId"></param>*  */// <returns></returns>*  static abstract int Used(DatabaseSystem db, string couponId, long playerId);  } |
| --- |

## public 쿠폰 사용

| CREATE TABLE `coupons\_use\_public` (  `id` bigint NOT NULL AUTO\_INCREMENT COMMENT '시퀸스',  `coupon\_id` varchar(12) NOT NULL COMMENT '쿠폰 번호',  `player\_id` bigint NOT NULL COMMENT '플레이어 식별자',  `created\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '생성일',  PRIMARY KEY (`id`),  UNIQUE KEY `uidx\_coupon\_id\_player\_id` (`coupon\_id`,`player\_id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=12 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci COMMENT='public 쿠폰 사용'; |
| --- |

| public class PublicCoupon : BaseCoupon  {  */// <summary>*  */// 이미 사용한 쿠폰인지 체크*  */// </summary>*  */// <param name="db"></param>*  */// <param name="couponId"></param>*  */// <param name="playerId"></param>*  */// <returns></returns>*  public override bool CheckdAlreadyUsedCoupon(DatabaseSystem db,string couponId, long playerId)  {  return db.Global.CouponUsePublic.GetAsync(couponId, playerId).Result != null;  }  */// <summary>*  */// 쿠폰 사용*  */// </summary>*  */// <param name="db"></param>*  */// <param name="couponId"></param>*  */// <param name="playerId"></param>*  */// <returns></returns>*  public override int Used(DatabaseSystem db,string couponId, long playerId)  {  CouponUsePublicRow row = new()  {  player\_id = playerId,  coupon\_id = couponId,  };  return db.Global.CouponUsePublic.InsertAsync(row).Result;  }  } |
| --- |

## scratch 쿠폰 사용

| CREATE TABLE `coupons\_use\_scratch` (  `id` bigint NOT NULL AUTO\_INCREMENT COMMENT '시퀸스',  `coupon\_id` varchar(12) NOT NULL COMMENT '쿠폰 번호',  `player\_id` bigint NOT NULL COMMENT '플레이어 식별자',  `created\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '생성일',  PRIMARY KEY (`id`),  UNIQUE KEY `uidx\_coupon\_id` (`coupon\_id`)  ) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci COMMENT='scratch 쿠폰 사용'; |
| --- |

| public class ScratchCoupon : BaseCoupon  {  */// <summary>*  */// 이미 사용한 쿠폰인지 체크*  */// </summary>*  */// <param name="db"></param>*  */// <param name="couponId"></param>*  */// <param name="playerId"></param>*  */// <returns></returns>*  public override bool CheckdAlreadyUsedCoupon(DatabaseSystem db, string couponId, long playerId)  {  return db.Global.CouponUseScratch.GetAsync(couponId).Result != null;  }  */// <summary>*  */// 쿠폰 사용*  */// </summary>*  */// <param name="db"></param>*  */// <param name="couponId"></param>*  */// <param name="playerId"></param>*  */// <returns></returns>*  public override int Used(DatabaseSystem db, string couponId, long playerId)  {  var row = new CouponUseScratchRow  {  player\_id = playerId,  coupon\_id = couponId,  };  return db.Global.CouponUseScratch.InsertAsync(row).Result;  }  } |
| --- |

## 쿠폰 보상(db테이블 생성)

| SELECT \* FROM p\_dev\_global.event\_ontime\_rewards; 참조 |
| --- |

| CREATE TABLE `coupons\_rewards` (  `id` bigint NOT NULL AUTO\_INCREMENT COMMENT '시퀸스',  `name` varchar(64) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL COMMENT '쿠폰 이름',  `desc` varchar(128) CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci NOT NULL COMMENT '쿠폰 설명',  `type` int NOT NULL COMMENT '쿠폰 타입',  `items` json NOT NULL COMMENT '쿠폰 보상 아이템 TBagSlotSerializer json',  `created\_at` datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP COMMENT '생성일',  PRIMARY KEY (`id`),  KEY `idx\_created\_at` (`created\_at`) ) ENGINE=InnoDB AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci COMMENT='쿠폰 보상 테이블'; |
| --- |

| [Table("coupons\_rewards")]  public class CouponsRewardRow  {  [Key] public long id { get; set; }  public string name { get; set; }  public string desc { get; set; }  public int type { get; set; } *// 1:public 쿠폰, 2:scratch 쿠폰*  public string items { get; set; } *//TBagSlotSerializer 사용.*  public DateTime created\_at { get; set; }  } |
| --- |

| */// <summary>*  */// 쿠폰 보상 DB*  */// </summary>*  public class CouponsRewardDbSet  {  private static readonly ILoggerEx Logger = LoggerFactoryEx.CreateLogger<CouponsRewardDbSet>();  private DbNode \_node;  public CouponsRewardDbSet(DbNode node)  {  \_node = node;  }  public async Task<int> InsertAsync(CouponsRewardRow row)  {  row.created\_at = DateTime.UtcNow;  row.id = await \_node.DoAsync(conn => conn.InsertAsync<int, CouponsRewardRow>(row));  return row.id > 0 ? 1 : 0;  }  public Task<IEnumerable<CouponsRewardRow>> GetAllAsync(IEnumerable<long> ids)  {  if (!ids.Any())  {  return Task.FromResult(Enumerable.Empty<CouponsRewardRow>());  }  var sql = @"SELECT \* FROM `coupons\_rewards` WHERE `id` IN @ids";  var param = new  {  ids = ids  };  return \_node  .DoAsync(conn => conn.QueryAsync<CouponsRewardRow>(sql, param));  }  } |
| --- |

# 

# 의문사항

1. 메타 값 관리는 어떻게 할것인가??
   1. 메타 데이터가 현재 없음
      1. ‘coupons’ db에 json 형태로 존재했으면 합니다.
   2. 해당 쿠폰에 대한 정보 (지급 아이템, 지급 타입, 지급 기간 등등)
      1. json 형태에 메타가 있었으면 합니다.
2. 이벤트 보상은 현재 db에 존재하는데 같은 방식으로 진행할지?

* 추후 운영툴에서 보상 관리 할수 있으므로

# 메타 데이터 설계

* couponid

| {  "id": 1001, // 쿠폰 아이디  "coupontype":"public", // 쿠폰 타입  "memo":"봄맞이 이벤트",  "startdate":"2031-1-1", // 쿠폰 시작일  "enddate":"2031-12-31", // 쿠폰 만료일  "items":{  {  "type": "item",  "index": 10001, // 장비 아이템 가정  "count": 1,  },  {  "type": "item",  "index": 10002, // 물약 아이템 가정  "count": 10,  },  {  "type": "durationbuff",  "index": 20001,  "duration": 100, // 지속 시간(초)  },  {  "type": "timeoverbuff",  "index": 30001,  "enddate": "2030-12-31", // 버프 종료 시간(초)  },  } } |
| --- |

# UniTests 코드

## 쿠폰 관련 UnitTests 프로젝트

CouponMetaOptionTests

CouponSystemTests

RItemSetTests

| [Test]  public void TestUse()  {  using var entity1 = TestWorld.DummyGamePlayer.Create();  var player1 = entity1.GetComponent<GamePlayer>();  using var entity2 = TestWorld.DummyGamePlayer.Create();  var player2 = entity2.GetComponent<GamePlayer>();  TestWorld.WebCoupon.Generate(CouponType.Public.Value, 2, (DateTime.Now + TimeSpan.FromDays(1)).Ticks, out var publicCoupons);  TestWorld.WebCoupon.Generate(CouponType.Scratch.Value, 2, (DateTime.Now + TimeSpan.FromDays(1)).Ticks, out var scratchCoupons);  *// 존재하지 않는 쿠폰 사용해본다. 실패해야 함*  int status = TestWorld.WebCoupon.Use(player1.Psn.Uid, "ADGHWER234SR");  Assert.That(status, Is.EqualTo(StatusCodeEx.**InvalidCouponId**));  *// public 쿠폰 하나 사용해본다 성공해야 함*  status = TestWorld.WebCoupon.Use(player1.Psn.Uid, publicCoupons.ElementAt(0).CouponId);  Assert.That(status, Is.EqualTo(StatusCodeEx.**Success**));  *// public 쿠폰 사용한걸 또 사용해본다, 실패해야함*  status = TestWorld.WebCoupon.Use(player1.Psn.Uid, publicCoupons.ElementAt(0).CouponId);  Assert.That(status, Is.EqualTo(StatusCodeEx.**AlreadyUseCouponId**));  *// public 쿠폰 다른거 또 사용해본다 성공해야 함*  status = TestWorld.WebCoupon.Use(player1.Psn.Uid, publicCoupons.ElementAt(1).CouponId);  Assert.That(status, Is.EqualTo(StatusCodeEx.**Success**));  *// 1번 유저가 사용한 public 쿠폰 하나 사용해본다 성공해야 함*  status = TestWorld.WebCoupon.Use(player2.Psn.Uid, publicCoupons.ElementAt(0).CouponId);  Assert.That(status, Is.EqualTo(StatusCodeEx.**Success**));  *// scratch 쿠폰 하나 사용해본다 성공해야 함*  status = TestWorld.WebCoupon.Use(player1.Psn.Uid, scratchCoupons.ElementAt(0).CouponId);  Assert.That(status, Is.EqualTo(StatusCodeEx.**Success**));  *// scratch 쿠폰 사용한거 또 사용해본다 실패해야 함*  status = TestWorld.WebCoupon.Use(player1.Psn.Uid, scratchCoupons.ElementAt(0).CouponId);  Assert.That(status, Is.EqualTo(StatusCodeEx.**AlreadyUseCouponId**));  *// 1번 유저가 사용한 scratch 쿠폰 하나 사용해본다 실패해야 함*  status = TestWorld.WebCoupon.Use(player2.Psn.Uid, scratchCoupons.ElementAt(0).CouponId);  Assert.That(status, Is.EqualTo(StatusCodeEx.**AlreadyUseCouponId**));  *// 1번 유저가 사용하지 않은 scratch 쿠폰 하나 사용해본다 성공해야 함*  status = TestWorld.WebCoupon.Use(player2.Psn.Uid, scratchCoupons.ElementAt(1).CouponId);  Assert.That(status, Is.EqualTo(StatusCodeEx.**Success**));  } |
| --- |

# 메일 시스템 분석

* 쿠폰 보상은 메일을 통해 보상 지급

## 메일 쓰기

.\projectg\_server\src\Plutus.Server.Game\Worlds\Players\PlayerMailComponent.cs

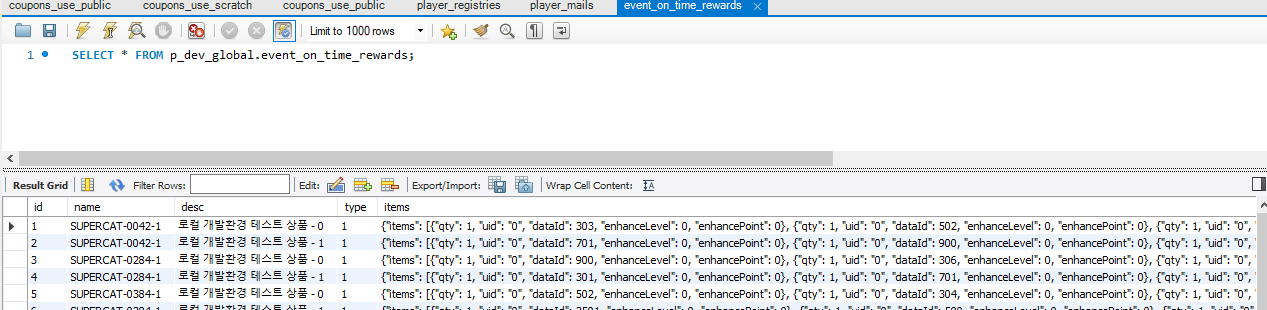
| public int Write(long receiverId, string text, IList<TPickedItem> attachments)  {  if (string.IsNullOrEmpty(text))  return StatusCodeEx.**NotFoundMailText**;  bool hasInvalidDataId = attachments  .GroupBy(x => x.Uid)  .Select(x => x.DistinctBy(y => y.DataId))  .Any(x => 2 <= x.Count());  if (hasInvalidDataId)  return StatusCodeEx.**InvalidDataId**;  var safeAttachments = attachments.GroupBy(x => x.Uid)  .Select(x =>  {  return new TPickedItem()  {  Uid = x.Key,  DataId = x.First().DataId,  Qty = x.Sum(y => y.Qty),  };  }).ToList();  *// 실제 아이템을 가지고 있는지 체크 한다.*  bool hasAll = safeAttachments.All(x => \_player.Bag.Contains(x.DataId, x.Uid, x.Qty));  if (!hasAll)  {  var jsons = string.Join(",", safeAttachments.Select(x => x.ToJson()));  Logger.Error($"invalid item - playerUid({\_player.Psn.Uid}) attacheItems({jsons})");  return StatusCodeEx.**NotFoundItem**;  }  var db = GetSystem<DatabaseSystem>();  var dashboardSvc = GetSystem<ServerDashboardSystem>();  var receiverRow = db.Global.PlayerRegistry.GetAsync(receiverId).Result;  if (null == receiverRow)  {  Logger.Error($"not found player - playerUid({\_player.Psn.Uid}) receiverId({receiverId})");  return StatusCodeEx.**NotFoundPlayer**;  }  *// 플레이어가 메일을 보낼 때, 나와 다른 월드에 보낼 순 없다(아이템 월드 이동 되기 떄문)*  if (dashboardSvc.WorldId != receiverRow.world\_id)  {  Logger.Error(  $"not match world - playerUid({\_player.Psn.Uid}) receiverId({receiverId}) worldId({dashboardSvc.WorldId}) receiver worldId({receiverRow.world\_id})");  return StatusCodeEx.**NotFoundPlayer**;  }  var writableItems = new List<TItem>();  foreach (var pickedItem in safeAttachments)  {  int status = \_player.Bag.TryRemove(pickedItem.DataId, pickedItem.Uid, pickedItem.Qty, out var removeItem);  if (!StatusCodeEx.IsSuccess(status))  {  Logger.Error(  $"item remove failed - playerUid({\_player.Psn.Uid}), itemDataId({pickedItem.DataId}), itemUid({pickedItem.Uid}), Qty({pickedItem.Qty}) status({status})");  continue;  }  writableItems.Add(removeItem);  }  var option = new MailOption  {  SenderId = \_player.Psn.Uid,  SenderName = \_player.Psn.Name,  ReceiverId = receiverId,  Text = text,  };  if (0 < writableItems.Count)  option.AttachedItem = writableItems;  var mailSvc = GetSystem<GameMailSystem>();  var affected = mailSvc.WriteAsync(option).Result;  if (0 >= affected)  return StatusCodeEx.**DbAffectedError**;  return StatusCodeEx.**Success**;  } |
| --- |

## 이벤트에서 메일로 아이템 지급

| private bool TryGiveReward(EventOnTimeReward reward)  {  var state = GetOrDefault(reward.Uid);  int prevState = state.CompareAndExchange(1, 0);  if (0 != prevState)  {  return false;  }  var row = new PlayerEventOnTimeRewardRow();  row.player\_id = \_player.Psn.Uid;  row.reward\_id = reward.Uid;  row.rewarded = 1;  var db = GetSystem<DatabaseSystem>();  int affected = db.Game.PlayerEventOnTimeReward.InsertAsync(row).Result;  if (affected <= 0)  {  Logger.Error($"player event rewarded - playerUid({\_player.Psn.Uid}) rewardId({reward.Uid})");  return false;  }  *// 잡 익스큐터에 넣음으로써, 어쨋든 보내도록 한다*  var executor = GetSystem<ExecutorSystem>();  executor  .Job  .Fastable()  .Execute(() => \_player.Mail.WriteToMe(reward.Name, reward.Desc, reward.Items));  Logger.Info($"on time event rewarding - playerUid({\_player.Psn.Uid}) rewardUId({reward.Uid})");  return true;  } |
| --- |

* 이벤트 보상 아이템은 db에 존재

| SELECT \* FROM p\_dev\_global.event\_on\_time\_rewards; |
| --- |



| public class EventOnTimeSystem : XSystem  {  ……………….  private void LoadInternal(int worldId, out EventOnTimeModified modified)  {  var db = GetSystem<DatabaseSystem>();  var now = GameZonedDateTime.Now();  var start = GameZonedDateTime.FirstTimeOf(now.AddDays(-1));  var end = start.AddDays(2);  *// 어제 ~ 내일 까지 스케쥴 로드*  var temp3DaySchedules = db.Global.OnTimeSchedule  .GetAllAsync(worldId, start.ToUtc(), end.ToUtc())  .Result  .ToImmutableArray();  var temp3DayScheduleIds = temp3DaySchedules  .Select(x => x.reward\_id)  .Distinct();  *// 어제 ~ 내일 까지 리워드 로드*  var temp3DayRewards = db.Global.OnTimeReward  .GetAllAsync(temp3DayScheduleIds)  .Result  .ToDictionary(x => x.id, x => x);  var tempActivatedSchedules = temp3DaySchedules  .Where(x => x.IsActivated(now))  .ToDictionary(x => x.id, x => x);  *// 다시 로드한 스케쥴 - 이미 캐싱한 스케쥴 = 추가된 스케쥴*  modified = new();  modified.AddedEvent = tempActivatedSchedules.Values  .Where(r => !\_lastActivatedScheduleIds.Contains(r.id))  .ToImmutableArray();  *// 이미 캐싱한 스케쥴 - 다시 로드한 스케쥴 = 삭제 스케쥴*  modified.RemovedEventIds = \_lastActivatedScheduleIds  .Where(x => !tempActivatedSchedules.ContainsKey(x))  .ToImmutableArray();  \_lastActivatedScheduleIds = tempActivatedSchedules.Keys.ToHashSet();  \_last3DayRewards = temp3DayRewards.Values.ToDictionary(x => x.id, EventOnTimeReward.Of);  \_todayTOnTimes = temp3DaySchedules  .Where(s => s.IsActivatedOnDay(now))  .Select(s => ToTOnTime(s))  .Where(s => null != s)  .ToImmutableArray();  \_todayRewardIdsByType = temp3DaySchedules  .Where(s => s.IsActivatedOnDay(now))  .Select(s => \_last3DayRewards.GetValueOrDefault(s.reward\_id))  .Where(r => null != r)  .GroupBy(r => r.Type)  .ToDictionary(r => r.Key, r => r.Select(x => x.Uid).ToImmutableArray());  \_last3DaySchedules = temp3DaySchedules.ToImmutableArray();  } |
| --- |

# WebAgent 로 메시지 보내기

* 쿠폰 보상은 메일을 통해 보상 지급

## webAgent 주소 가져오기

* 전체 서버 config 에서 hostname, use\_type =2 로 검색한다.

| *// 모든 서버 정보는 많아야 1천개 이하이므로, 모두 읽어 둔다*  private void RefreshForServerShardInternal()  {  var db = GetSystem<DatabaseSystem>();  \_serverConfigs = db.Global.ServerConfig  .GetAllAsync()  .Result  .ToDictionary(x => (int)x.id, x => x);  \_serverShards = db.Global.ServerShardConfig  .GetAllAsync()  .Result  .ToDictionary(x => x.server\_id, x => x);  } |
| --- |

# 치트키 추가

| *// 쿠폰*  public static readonly CheatCmd CouponCreate = new(nameof(CouponCreate), "쿠폰생성", "//쿠폰생성 {UseType} {Count} {ExpireTicks}");  public static readonly CheatCmd CouponUse = new(nameof(CouponUse), "쿠폰사용", "//쿠폰사용 {CouponId}"); |
| --- |

* 쿠폰 생성 관련 치트키 작업
* 아이템번호:수량,아이템번호:수량

## 아이템 생성

| *// 아이템 관련*  public static readonly CheatCmd Item = new(nameof(Item), "아이템", "//아이템 {ItemDataId} {수량} {강화레벨}");  public static readonly CheatCmd ItemDrop = new(nameof(ItemDrop), "드랍", "//드랍 {ItemDataId} {수량}", "아이템드랍", "드랍아이템"); |
| --- |

예시) 동동주 생성

| //아이템 1001 1 1 |
| --- |

| var text = "1001:1,1002:2,1003:3,1004:4,1005:5";  var pattern = @"\s\*,\s\*";  var subPattern = @"\s\*:\s\*";  string[] substrings = Regex.Split(text, pattern);  foreach (var substring in substrings)  {  var match = Regex.IsMatch(substring, subPattern);  if (match == false)  {  continue;  }  string[] tempSub = Regex.Split(substring, subPattern);  Console.WriteLine(tempSub);  } |
| --- |

| public class ResourceSystem : XSystem  {  public RItemSet Item => \_resContext.Item; |
| --- |

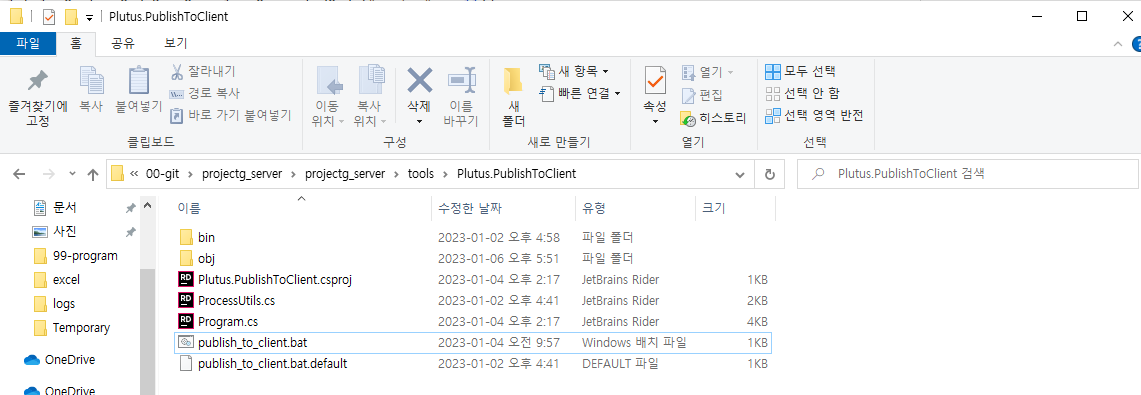
| *// 아이템 보상 지급용 json을 얻어온다. - ex)1001:1,1002:1,1003:1*  public static bool OfRewardJson(IEnumerable<RItem> resItems, string itemRewards, out string rewardJson)  {  rewardJson = string.Empty;  if (string.IsNullOrEmpty(itemRewards))  {  return false;  }  var createdItemPattern = @"\s\*,\s\*";  var subPattern = @"\s\*:\s\*";  var createdItems = Regex.Split(itemRewards, createdItemPattern);  var itemList = new List<TItem>();  try  {  foreach (var createdItem in createdItems)  {  if (Regex.IsMatch(createdItem, subPattern) == false)  {  return false;  }  var subItems = Regex.Split(createdItem, subPattern);  if (subItems.Length != 2)  {  return false;  }  var itemId = Int32.Parse(subItems[0]);  var qty = Int32.Parse(subItems[1]);  if (itemId <= 0 || qty <= 0)  {  return false;  }  if (TryGetItemProperty(resItems, itemId, out var rItemProperty) == false)  {  return false;  }  if (qty > rItemProperty.ItemStack) *// [아이템 중첩 수] 보면 오류*  {  return false;  }  itemList.Add(new TItem  {  DataId = itemId,  Qty = qty  });  }  }  catch (Exception e)  {  return false;  }  if (itemList.Any() == false)  {  *// 보상이 없다*  return false;  }  var serializer = new TBagSlotSerializer  {  Items = { itemList },  };  rewardJson = serializer.ToJson();  return true;  } |
| --- |

# google protobuff 수정

| message CouponCreateWebRequest {  int32 use\_type = 1;  int32 count = 2;  int64 expire\_ticks = 3;  int64 reward\_id = 4;  } |
| --- |

| new pbr::GeneratedClrTypeInfo(typeof(global::Plutus.Proto.Messages.CouponCreateWebRequest), global::Plutus.Proto.Messages.CouponCreateWebRequest.Parser, new[]{ "UseType", "Count", "ExpireTicks" }, null, null, null, null), |
| --- |

## .proto 빌드



* bat 파일을 실행하여 .proto 파일 빌드

# ssl connection 오류 개선

위에 코드 오류가 있어 수정 필요

| System.Net.WebException: The SSL connection could not be established, see inner exception.  ---> System.Net.Http.HttpRequestException: The SSL connection could not be established, see inner exception.  ---> System.Security.Authentication.AuthenticationException: The remote certificate is invalid according to the validation procedure: RemoteCertificateNameMismatch  at System.Net.Security.SslStream.CompleteHandshake(SslAuthenticationOptions sslAuthenticationOptions)  at System.Net.Security.SslStream.ForceAuthenticationAsync[TIOAdapter](Boolean receiveFirst, Byte[] reAuthenticationData, CancellationToken cancellationToken)  at System.Net.Http.ConnectHelper.EstablishSslConnectionAsync(SslClientAuthenticationOptions sslOptions, HttpRequestMessage request, Boolean async, Stream stream, CancellationToken cancellationToken)  --- End of inner exception stack trace ---  at System.Net.Http.ConnectHelper.EstablishSslConnectionAsync(SslClientAuthenticationOptions sslOptions, HttpRequestMessage request, Boolean async, Stream stream, CancellationToken cancellationToken)  at System.Net.Http.HttpConnectionPool.ConnectAsync(HttpRequestMessage request, Boolean async, CancellationToken cancellationToken)  at System.Net.Http.HttpConnectionPool.CreateHttp11ConnectionAsync(HttpRequestMessage request, Boolean async, CancellationToken cancellationToken)  at System.Net.Http.HttpConnectionPool.AddHttp11ConnectionAsync(QueueItem queueItem)  at System.Threading.Tasks.TaskCompletionSourceWithCancellation`1.WaitWithCancellation(CancellationToken cancellationToken)  at System.Threading.Tasks.TaskCompletionSourceWithCancellation`1.WaitWithCancellationAsync(Boolean async, CancellationToken cancellationToken)  at System.Net.Http.HttpConnectionPool.HttpConnectionWaiter`1.WaitForConnectionAsync(Boolean async, CancellationToken requestCancellationToken)  at System.Net.Http.HttpConnectionPool.SendWithVersionDetectionAndRetryAsync(HttpRequestMessage request, Boolean async, Boolean doRequestAuth, CancellationToken cancellationToken)  at System.Net.Http.HttpMessageHandlerStage.Send(HttpRequestMessage request, CancellationToken cancellationToken)  at System.Net.Http.DiagnosticsHandler.SendAsync(HttpRequestMessage request, Boolean async, CancellationToken cancellationToken)  at System.Net.Http.RedirectHandler.SendAsync(HttpRequestMessage request, Boolean async, CancellationToken cancellationToken)  at System.Net.Http.HttpMessageHandlerStage.Send(HttpRequestMessage request, CancellationToken cancellationToken)  at System.Net.Http.SocketsHttpHandler.Send(HttpRequestMessage request, CancellationToken cancellationToken)  at System.Net.Http.HttpClient.Send(HttpRequestMessage request, HttpCompletionOption completionOption, CancellationToken cancellationToken)  at System.Net.HttpWebRequest.SendRequest(Boolean async)  at System.Net.HttpWebRequest.GetResponse()  --- End of inner exception stack trace ---  at System.Net.HttpWebRequest.GetResponse()  at Plutus.Server.Common.Worlds.Systems.WebSystem.Post(String baseUrl, String relativeUrl, String jsonParameters, String& reponseText, Int32& rtnStatus) |
| --- |

**개선법 1)**

| if (\_initCertificate == false)  {  *// .Net Core Console 개발시, Https 접근을 시도하면 나타나는 예외에 대한 처리*  *//https://stackoverflow.com/questions/2675133/c-sharp-ignore-certificate-errors*  ServicePointManager.ServerCertificateValidationCallback += (sender, cert, chain, sslPolicyErrors) => true;  \_initCertificate = true;  } |
| --- |

* 위와 같이 처리하면 전체 서버 ssl에 영향을 주므로 패킷 단위로 하는 방식으로 변경 필요

**개선법2)**

아래와 같이 해결

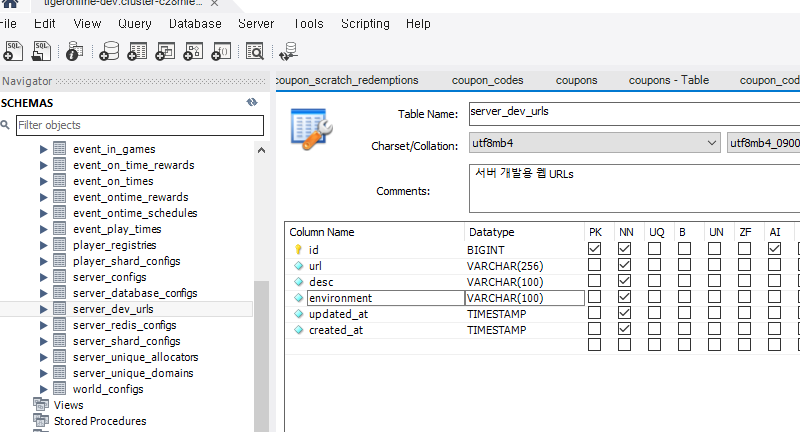
| var webRequest = (HttpWebRequest)WebRequest.Create(uri);  webRequest.Method = "POST";  webRequest.Timeout = 300; *// 300 msec*  webRequest.ContentType = "application/json";  *//https://stackoverflow.com/questions/12506575/how-to-ignore-the-certificate-check-when-ssl*  *// ssl 때 인증서 확인을 무시하는 방법*  webRequest.ServerCertificateValidationCallback  += (sender, certificate, chain, sslPolicyErrors) => true;  var postContentBytes = Encoding.UTF8.GetBytes(jsonParameters);  webRequest.ContentLength = postContentBytes.Length;  using (var reqStream = webRequest.GetRequestStream())  {  reqStream.Write(postContentBytes, 0, postContentBytes.Length);  }  *// Response 처리*  using (var resp = webRequest.GetResponse())  {  var respStream = resp.GetResponseStream();  using (var sr = new StreamReader(respStream))  {  reponseText = sr.ReadToEnd();  var reponseJObject = JObject.Parse(reponseText);  if (reponseJObject["status"] == null)  {  reponseText = string.Empty;  rtnStatus = StatusCodeEx.**NoResponseOnGameServer**;  return false;  }  rtnStatus = reponseJObject["status"].Value<int>();  }  } |
| --- |

# 코드리뷰

## 1차

| 리뷰  - 모든 테이블은 특수한 경우가 아니라면, 복수형으로 끝나야 합니다. 관용어구 입니다.  1. coupons table  - 이 테이블이 쿠폰 메타 정보 테이블이여야 합니다.(수정완료)  - coupons\_rewards 의 이름이 coupons 여야 합니다. (수정완료)  - 쿠폰 메타 정보 이므로, 이름, 설명, 기간, 타입, 보상 칼럼을 정리 해야 합니다. (수정완료)  2. coupon\_codes tables  - 쿠폰 코드들을 담고 있는 테이블 입니다. (수정완료)  - 기존 coupon\_id -> coupon\_code(수정완료)  - 기존 reward\_id -> coupon\_id(수정완료)  - expired\_at 삭제 (수정완료)  유저가 쿠폰을 사용 하는 절차  1. 유저가 쿠폰을 어딘가에서 받는다  2. 유저가 쿠폰 코드를 입력한다.  구조는  1. 쿠폰 코드로 쿠폰 메타 정보에 접근해야 한다.(수정완료)  2. 쿠폰 메타정보에는 쿠폰의 이름/사용기간/설명/보상 등의 정보가 기록 되어 있다.(수정완료)  3. 쿠폰 메타정보를 확인해서, 쿠폰을 사용하고, 각 쿠폰 타입에 맞게 처리 한다.(수정완료)  웹서버로 쿠폰 사용을 게임서버가 전달 하는데 몇가지 문제가 있어요  1. 로컬/개발/스테이지/라이브일 경우, 각 환경마다 서버 URL 이 달라요(수정완료)  2. 머신이 여러개일 때, 호스트 네임이 다를 수 있어요(수정완료)  3. 그러므로, 서버 실행 환경에 따라, 하드 코드로 URL 을 기입하는게 좋을거 같아요(4번 방식으로 수정완료)  4. 구조적으로 문제를 해결한다면, server\_dev\_urls 에 환경 컬럼 추가하여, 각 환경별 url 가저오면 되(수정완료)  쿠폰 사용과 생성은 처리 주체가 다름을 프로그래밍 해야 할거 같아요  1. 쿠폰 시스템에서 생성과 사용을 처리 합니다.  2. 웹 에이전트에서는 쿠폰 사용만 처리 합니다.  3. 게임 서버에서는 쿠폰 시스템을 이용하여, 생성만 처리 합니다.(수정완료)  4. 호크(매:운용툴) 서버에서는 게임서버와 동일하게 생성만 처리 합니다. 아직은 호크 서버 틀만 있어요  5. 그러므로, 쿠폰 시스템은 Server.Common 으로 올려야 된다고 생각 합니다.(수정완료) |
| --- |

* server\_dev\_urls 에 environment 컬럼 추가



## 2차

쿠폰 2차 리뷰

dev/23.1.3.001/added-coupon-2st

1. 디비 테이블명과 애플리케이션의 DBSet 이름은 맞추고 있어요(수정완료)

예) CouponUsePublicDbSet -> CouponPublicRedemptionDbSet

2. 쿠폰 메타 정보도 쿠폰 시스템에서 생성하는 로직 만들어 주세요(수정완료)

CouponSystem.Create(CouponMetaOption option) …

3. 쿠폰 메타의 타입이 퍼블릭, 스크래치 중 하나이므로, 쿠폰 코드 생성은 반드시 한 계열로만 생성 되어야 합니다. (수정완료)

- 왜냐하면 각 타입별로 사용이 다르기 때문에, 같은 쿠폰 메타로 다른 타입으로 생성 될 수 없어요

3. 유닛테스트 작성시, 하나의 테스트는 하나의 역활만 수행하게 하면 좋아요 (수정완료)

예) TestGenerate 에서는 Generate 관련 테스트를 묶고, TestUse 에서는 Use 만 묶는 형태

1. 로그 남길 때, 상수영역과 변수영역으로 나뉘어야 하고, 예제는 다음과 같습니다 (수정완료)

Logger.Error($"skill coll time is over - playerUid({\_actor.Uid}) skillId({skillId})");

- 상수부는 소문자로 구성

- 변수부는 카멜케이스로 $"({변수})" 형태로로 남깁니다.

- 상수부와 변수부는 - 로 구분합니다.

| Logger.Info($"coupon code generating prepared - couponUid({couponId}) count({begin + count}) added({count})"); |
| --- |

2. 쿠폰 메타 정보 인서트 기능 구현 필요함(수정완료)

3. 샤드 시스템에서 디버깅용 \_serverDevUrls 를 로드하지 않아요 (수정완료)

- 왜냐하면, 샤드 시스템은 온리 샤드 정보만 담아야 하고,

- 치트에서만 사용되는 코드는 정규 시스템에 포함시키지 않기 때문이에요

4. ServicePointManager.ServerCertificateValidationCallback(수정완료)

- 시스템 전 영역에서 컨트롤하기 위한 제어보다

- 요청 단위로 인증 무시하는거로 바꿔야 할거 같아요

## 3차

1. 유닛 테스트의 실패/성공 케이스에 대한 작업 필요(수정완료)

* 유닛테스트시 초기엔 실패들에 대한 처리

2. 퍼블릭 쿠폰의 단수 생성 기능 제약 필요(수정완료)

3. 비동기 함수 만들때, 접미사로 Async 를 붙여야 함(수정완료)

## 4차

유닛 테스트 작성시

1. 파라미터는 되도록 컨텍스트 없이 구성해주세요.(수정완료)

2. 기본값이 있는 것이라면, 기본값을 생성 후, 특정값을 망가뜨리고 만들어서 해도 좋아요(수정완료)

쿠폰 메타 생성의 시간 체크 더 추가

1. 시작 시간이 끝 시간 보다 미래일 수 없다 (수정완료)

2. 끝 시간이 현재 시간보다 과거 일 수 없다 (수정완료)

3. 시작 시간이 기본값은 이포치 타임의 시작 값인게 안전하다 (수정완료)

에러 스트링은 아웃 파람으로 빼지 않아요 (수정완료)

- 왜냐하면 로거 출력 스트링은 컨텐츠랑 독립적이여야 하기 때문이에(수정완료)

CouponMetaOption 안에서 메타 옵션을 체크하는게 좋아 보여요 (수정완료)

- 특정 객체의 멤버 변수로만 체크 되어야 하는 것은 그 객체에 완전한 종속이 되었기 때문이에(수정완료)

쿠폰 코드 제네레이트 테스트 할 때

1. 퍼블릭 쿠폰 코드 제네레이트시 같은 쿠폰 코드로 생성하면, 쿠폰 코드가 계속 제네레이트 되는 버그가 있어요 (수정완료)

쿠폰 사용 테스트 할 때

1. 존재하지 않는 플레이어로 쿠폰 사용시 사용되는 버그가 있어요(수정완료)

2. 없는 쿠폰을 사용 할 때, 사용 안되는 테스트가 없어요(수정완료)

ps

1. unit-test 사용할떄 -1로 사용하는 것이 아니고 0값으로 사용 시도 (수정완료)
2. 없는 scratch 쿠폰 사용 시도(수정완료)
3. public 쿠폰 없는 유저로 사용 시도??? <- 이게 가능해?????

## 5차

1. OfRewardJson 함수의 경우, RItemSet 쪽으로 옮겨주세요(수정완료)

2. RItemSetTests 클래스 추가하여, 1번 함수 테스트 필요 (수정완료)

3. CouponMetaOptionTests 추가하여, 테스트 필요 (수정완료)

4. coupons 에 seed 중복시 생성 불가 로직 필요함, 개별 유니크 (수정완료)

5. coupons 에 이름 중복시 생성 불가 로직 필요함, 개별 유니크 (수정완료)

## 6차

1. CouponMetaOption.IsValidate 에 파라미터 db 제거(수정완료)

* db 체크는 밖에서 체크하게 변경

2. rewardItems Json 가져오는 방식 변경(수정완료)