# 캐릭터 초기화 처리

| *// 준비*  public PlayerActor Prepare()  {  if (null != \_actor)  {  *// \_actor 가 null 이 아니면 재접속*  Logger.Info(  $"exist actor maybe reconnected - accountUid({\_player.Psn.AccountId}) playerUid({\_player.Psn.Uid})");  return \_actor;  *// // TODO : 이제 이 곳으로 들어오지 않는 것 같지만*  *// // Logging 해 놓고 체크.*  *// Logger.Error($"already exist actor - accountId:{\_player.Psn.AccountId}, playerId:{\_player.Psn.Uid}");*  *// return \_actor;*  }  int job = \_player.Psn.Job;  int lv = \_player.Exp.GetLevel();  var rSys = GetSystem<ResourceSystem>();  var actorSys = GetSystem<GameActorSystem>();  var option = new PlayerActorOption();  var actorDataId = ActorRules.JobTypeToActorDataId(job);  option.ResActor = rSys.Actor.GetById(actorDataId);  option.ResCharLevel = rSys.Level.GetLevelData(job, lv);  option.Position = \_player.PreservedActor.Position;  *// option.Status = player.Preserve.Status;*  *// TODO : 테스트를 위해*  option.Status = ActorStatus.**NORMAL**;  option.Hp = \_player.PreservedActor.Hp;  option.Mp = \_player.PreservedActor.Mp;  *// 확장값*  option.Player = \_player;  var tempActor = actorSys.CreateActor(option) as PlayerActor;  \_actor = tempActor;  *// Player Equip Item Value 적용*  \_player.Mount.Prepare();  *// 버프 적용*  \_player.Buff.Prepare();  var args = new FxPlayerActorBound()  {  Instant = DateTime.UtcNow,  };  \_player.Flux.Emit(args);  return tempActor;  } |
| --- |

| *public class RActorSet*  *{*  *private ImmutableDictionary<int, RActor> \_actors;*  *private ImmutableDictionary<int, RActorPortal> \_portals;*  *private ImmutableDictionary<int, RActorPlayer> \_players;*  *private ImmutableDictionary<int, RActorMonster> \_monsters;*  *private ImmutableDictionary<int, RActorNpc> \_npcs;*  *private ImmutableDictionary<int, RActorPet> \_pets;*  *// 플레이어 액터 생성/병합*  foreach (var element in loader.LoadPlayerData())  {  var rPlayer = RActorPlayer.OfPlayer(element.Value);  if (null == rPlayer)  {  throw new DataException($"not found player actor in base actors - id({element.Value.Id})");  }  tempPlayers.Add(element.Value.Id, rPlayer);  tempActors.Add(element.Value.Id, rPlayer);  }  \_actors = ImmutableDictionary.CreateRange(tempActors);  \_portals = ImmutableDictionary.CreateRange(tempPortals);  \_players = ImmutableDictionary.CreateRange(tempPlayers);  \_monsters = ImmutableDictionary.CreateRange(tempMonsters);  \_npcs = ImmutableDictionary.CreateRange(tempNpcs);  \_pets = ImmutableDictionary.CreateRange(tempPets); |
| --- |

# 스킬 관련

| public class ActorSkillComponent : XComponent  {  public ActorSkillComponent()  {  \_skills = new();  }  /////////////////////////////////  public int PreSkill(IEnumerable<long> targetIds, int skillId, IEnumerable<TPosition> positions, long mainTargetID, out ActorSkill skill)  {  *// 1.스킬 보유 체크*  skill = GetSkillById(skillId);  if (null == skill)  {  Logger.Error($"not found player skill - playerUid({\_actor.Uid}) skillId({skillId})");  return StatusCodeEx.**NotFoundSkill**;  }  *// 1.쿨타임 체크*  if (!skill.CheckCoolTime())  {  Logger.Error($"skill coll time is over - playerUid({\_actor.Uid}) skillId({skillId})");  return StatusCodeEx.**SkillCoolTime**;  }    *// Set coolTime*  skill.Reset();  *// 2.마나 소모 체크*  var status = skill.SkillCostSpend();  if (!StatusCodeEx.IsSuccess(status))  {  Logger.Error($"player mp is not enough - actorUid({\_actor.Uid}) skillId({skill.RSkill.Spec.Id})");  return status;  }  *// var needMp = skill.RSkill.Spec.MpNeedValue;*  *// if (0 < needMp && TryGetComponent<ActorManaComponent>(out var mana))*  *// {*  *// int status = mana.MpSpend(needMp);*  *// if (StatusCodeEx.Success != status)*  *// {*  *// Logger.Error($"player mp is not enough - actorUid({\_actor.Uid}) skillId({skill.RSkill.Spec.Id})");*  *// return status;*  *// }*  *// }*  *//공격자가 도적이면 투명 버프 상태인지 체크 후 해제*  *// 상태 처리.*  *//추후 상태 에서 invisible에 관계된 버프를 날릴것인지 체크해야함.*  if(\_actor.TryGetComponent<ActorStateComponent>(out var state))  {  *//string 를 날려야 할지 고민해보자.*  if (state.Has(ActorStateType.Invisible))  {  state.Clear(ActorStateType.Invisible);  }  }  *// BC*  var notify = new ActorPreSkillNotify()  {  AttackerId = \_actor.Uid,  TargetId = { targetIds },  DataSkillId = skillId,  Positions = { positions },  MainTargetId = mainTargetID  };  \_actor.Near.ToShowns(notify, true);  return StatusCodeEx.**Success**;  } |
| --- |

| public class RActorPlayer : RActor  {  public RActorPlayerProperty Player { get; private set; }  public static RActorPlayer OfPlayer(FBDataActorPCElement element)  {  var actor = new RActorPlayer();  actor.MergeFrom(element);  return actor;  }  public void MergeFrom(FBDataActorPCElement element)  {  Spec = AddProperty(new RActorProperty(element.Id, element.Name, element.Type));  Collider = AddProperty(new RColliderProperty(element.TouchColSize, element.ColliderSize));  Player = AddProperty(new RActorPlayerProperty(element));  }  public override List<int> GetSkills()  {  return Player.Skills;  } |
| --- |

| *// 플레이어 생성 옵션*  public class PlayerActorOption : ActorOption  {  public GamePlayer Player { get; set; }  public RLevelProperty ResCharLevel { get; set; }  *// PlayerSaved Data*  public GVector3 Position { get; set; }  public ActorStatus Status { get; set; }  public long Hp { get; set; }  public long Mp { get; set; }  public PlayerActorOption() : base(ActorType.Player)  {  }  } |
| --- |

| public GameActorSystem()  {  \_lock = new();  \_actors = new();  \_constructors = new();  \_constructors.Add(ActorType.Player, (entity, opt) => PlayerActor.Of(entity, opt as PlayerActorOption));  \_constructors.Add(ActorType.Monster, (entity, opt) => MonsterActor.Of(entity, opt as MonsterActorOption));  \_constructors.Add(ActorType.Npc, (entity, opt) => NpcActor.Of(entity, opt as NpcActorOption));  \_constructors.Add(ActorType.Portal, (entity, opt) => PortalActor.Of(entity, opt as PortalActorOption));  \_constructors.Add(ActorType.Item, (entity, opt) => ItemActor.Of(entity, opt as ItemActorOption));  \_constructors.TrimExcess();  } |
| --- |

| public static PlayerActor Of(XEntity entity, PlayerActorOption option)  {  actor.Skill.Initialize(option.ResActor); |
| --- |

## 모든 기술 초기화

* 플레이 중인 캐릭터가 습득한 기술 모두 미 습득 상태로 변경

//기술초기화

| public int CheatSkillInit(GamePlayer player, ImmutableArray<string> args) |
| --- |

## 현재 레벨 모든 기술 습득

* 플레이 중인 캐릭터 레벨까지 배울 수 있는 기술 모두 습득됨

//기술습득

| public int CheatSkillAcquire(GamePlayer player, ImmutableArray<string> args) |
| --- |