데이터 교환

1. 개요

- 클라이언트와 서버의 Json 데이터 교환을 위해 작성한다.
- 변수명의 통일, 분업화의 효율성을 높이기 위해 작성한다.
- 굳이 JObject를 통해 값을 감싸는 경우는, 클라이언트 측에서 클래스로 바로 변환하기 위해서이다.

2. 사용된 표현 설명

- '(데이터타입) 데이터명' 을 통해 하나의 데이터를 표현한다.
- Json 형식을 시각적으로 보여주기 위해 ':'와 '{ }', '[ ]'를 그대로 사용한다.
- '(JArray) 데이터명 : [(JObject) [index] : { }]'에서 index는 JArray 안에서의 순서를 의미한다.

  Ex) C#에서는 '데이터명[0]["데이터명"]'와 같이 사용할 수 있다.

- (default)는 서버에서 값을 지정하지 않았을 때 또는 아직 값이 존재하지 않는 값을 때를 명시한다.

3. 테이블 종류

1) 통합

- 유저, 캐릭터정보, 스킬정보, 몬스터정보, 등록된 거래 물품 목록, 완료된 거래 목록,

게임 기록, 랭킹, 업적 테이블

2) 개별

- 유저 스킬, 유저 캐릭터, 유저 상인, 유저 업적 테이블.

 # 해당 테이블의 자세한 정보는 E-R 다이어그램 참고.

4. 데이터 교환 형식

1) 목록

(1) 회원가입 : /register

(2) 로그인 : /login

(3) 회원탈퇴 : /deleteAccount

(4) 유저 정보와 선택 정보 요청 : /RequestUserDataAndSelectedData

(5) 소유 캐릭터 리스트 요청 : /RequestPossessedCharacterList

(6) 소유 스킬 리스트 요청 : /RequestPossessedSkillsList

(7) 상점 상품 리스트 요청 : /RequestVenderProductList

(8) 상점 상품 구매 요청 : /RequestPurchaseVenderProduct

(9) 거래소 구매-등록-등록취소 버튼 권한 제어 : /RequestMarketAccessibility

(10) 거래소 상품 리스트 요청 : /RequestMarketProductList

(11) 거래소 상품 구매 요청 : /RequestPurchaseMarketProduct

(12) 거래소 상품 등록 수수료 정보 요청 : /RequestMarketProductRegistrationFee

(13) 거래소 상품 등록 요청 : /RequestRegisterMarketProduct

(14) 거래소 내 등록 상품 정보 요청 : /RequestMyMarketRegistrationRecord

(15) 거래소 내 구매 상품 정보 요청 : /RequestMyMarketPurchaseRecord

(16) 거래소 내 판매 상품 정보 요청 : /RequestMyMarketSalesRecord

(17) 거래소 내 등록 상품 등록 취소 요청 : /RequestCancelMyMarketRegistrationProduct

(18) 인증 번호 요청 : /RequestVerificationCode

(19) 인증 성공 여부 요청 : /RequestWhetherAuthenticationSucceeded

(20) 기존 계정으로 변경 요청 : /RequestLinkExisitingAccount

(21) 월렛 주소 등록 요청 : /RequestRegisterWalletAddress

(22) 닉네임 변경 : /RequestModifyUserNickname

(23) 유저 정보와 선택 정보 등록 요청 : /RequestSettingGameData

(24) 캐릭터 정보 요청 : /RequestCharacterData

(25) 적 정보 요청 : /RequestEnemiesData

(26) 스킬 정보 요청 : /RequestSkillsData

(27) 게임 결과 등록 요청 : /RequestGameResult

1) 로그인 Scene

(1) 회원가입

① 경로 : /register

② 요청 데이터

(JObject) requestData : {

(string) accountPW : // MACAddress 값

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

(JObject) accountData : {

(string) accountID : // MACAddress를 통한 랜던 ID 할당

(string) accountPW : // MACAddress 값 사용

}

}

④ 추신

- 유저 테이블에 유저 정보 삽입
- 해당 유저의 유저 스킬 테이블, 유저 캐릭터 테이블, 유저 상인 테이블, 유저 업적 테이블 생성.

- MACAddress를 이용한 자동 회원가입 기능이다.
- 회원가입 성공 시, 로그인을 위한 AccountData가 로컬에 저장된다.

(2) 로그인

① 경로 : /login

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : // 응답 성공 여부

}

④ 추신

- 로그인 성공 시, '(4)' 요청이 다음 요청으로 전달된다.

(3) 회원 탈퇴

① 경로 : /deleteAccount

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : // 응답 성공 여부

}

④ 추신

- 해당 유저의 스킬 테이블, 유저 캐릭터 테이블, 유저 상인 테이블, 유저 업적 테이블 삭제.
- 타 테이블에서 해당 유저의 ID와 연관된 값들을 삭제.

ㄴ 거래 기록 테이블에서는 사라진 유저의 정보는 default로 표시. // 가능하면 해줘...

- 회원탈퇴 성공 시, 로그인을 위해 생성해 두었던 AccountData가 로컬에 삭제된다.

(4) 유저 정보와 선택 정보 요청

① 경로 : /RequestUserDataAndSelectedData

② 요청 데이터

```
(JObject) requestData : {
        (JObject) accountData : {
                (string) accountID :
                (string) accountPW :
        }
}
```

③ 응답 데이터

```
(JObject) responseData : {
        (bool) success :                                    // 응답 성공 여부
        (JObject) userData : {
                (string) userNickname : (default) guest     // 프로필 이름
                (int) userImage : (default) 0               // 프로필 이미지 번호.
                (int) possessedCoin : (default) 0           // 일반 재화
                (int) possessedPaidCoin : (default) 0       // 유료 재화
        }
        (JObject) selectedData : {
                (int) selectedCharacterNumber : 0           // 선택 캐릭터 정보
                (JArray) selectedSkillNumber : []           // 선택 스킬 정보
        }
}
```

④ 추신

- 로그인 직후에서의 요청에서는, 유저가 선택한 정보가 없기에 기본값인 0과 [] 값을 리턴해준다.

2) 게임 로비 Scene

(5) 소유 캐릭터 리스트 요청

① 경로 : /RequestPossessedCharacterList

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

(JArray) characterList : [

(JObject) [index] : {

(int) characterNumber : // 유저 캐릭터 테이블 ForeignKey

},

(JObject) [index] : { }, … // 위와 같은 JObject가 반복된다.

]

}

④ 추신

- 'characterNumber'는 유저 캐릭터 테이블의 ForeignKey이자, 캐릭터 테이블의 PrimaryKey이다.
- 캐릭터의 자세한 정보는 DB의 캐릭터 테이블, 로컬의 캐릭터 Json파일로 기록되어 있다.
- 로컬 정보는 UI 출력을 위해 사용하며, DB 정보는 게임 내 로직에서 사용된다.

(6) 소유 스킬 리스트 요청

① 경로 : /RequestPossessedSkillsList

② 요청 데이터

```
(JObject) requestData : {
        (JObject) accountData : {
                (string) accountID :
                (string) accountPW :
        }
}
```

③ 응답 데이터

```
(JObject) responseData : {
        (bool) success : (default) false                    // 응답 성공 여부
        (JArray) skillsList : [
                (JObject) [index] : {
                        (string) uniqueSkillNumber :    // 유저 스킬 테이블의 PrimaryKey
                        (int) reaminCount :             // 남은 사용 횟수
                        (bool) isNFT : (default) false  // NFT화 여부
                        (int) skillNumber :             // 유저 스킬 테이블 ForeignKey
                },
                (JObject) [index] : { }, ...            // 위와 같은 JObject가 반복된다.
        ]
}
```

④ 추신

- 스킬의 자세한 정보는 DB의 캐릭터 테이블, 로컬의 캐릭터 Json파일로 기록되어 있다.
- 로컬 정보는 UI 출력을 위해 사용하며, DB 정보는 게임 내 로직에서 사용된다.

(7) 상점 상품 리스트 요청

① 경로 : /RequestVenderProductList

② 요청 데이터

```
(JObject) requestData : {
        (JObject) accountData : {
                (string) accountID :
                (string) accountPW :
        }
        (string) requestType :
                // "select"는 기존의 유저 상인 출력.
                // "update"는 유저 상인 테이블 갱신 후 출력.
}
```

③ 응답 데이터

```
(JObject) responseData : {
        (bool) success : (default) false                    // 응답 성공 여부
        (JArray) venderProductList : [
                (JObject) [index] : {
                        (int) productNumber :               // 유저 상인 테이블의 PrimaryKey
                        (int) price:                        // 가격
                        (int) skillNumber :                 // 유저 상인 테이블 ForeignKey
                },
                (JObject) [index] : { }, ...                // 위와 같은 JObject가 반복된다.
        ]
}
```

④ 추신

- '요청 데이터'의 requestType의 값에 따라서 서버에서 리턴값을 다르게 준다.
- 요청과 응답 값의 형태가 똑같아서 이런 방식으로 합쳐보았다.
- 클라이언트부분에서의 코드는 줄었지만, 서버에는 오히려 복잡함이 증가한 느낌이다.
- 이걸로 마지막으로 이런식의 코드는 사용하지 않도록 하자.....

(8) 상점 상품 구매 요청

① 경로 : /RequestPurchaseVenderProduct

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

(int) productNumber : // 유저 상인 테이블 PrimaryKey

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

}

④ 추신

- 유저 테이블에서 accountID를 이용하여 유저 소유 재화 정보를 가져온다.
- 유저 상인 테이블에서 productNumber를 이용하여 상품 가격 정보를 불러온다.
- 상품 구매 성공 시, 유저 스킬 테이블에서 스킬 정보를 추가한다.

(9) 거래소 구매-등록-등록취소 버튼 권한 제어

① 경로 : /RequestMarketAccessibility

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

}

④ 추신

- 유저 인증 정보가 없을 시, 에러창을 표시하는 부분이다.
- 유저 테이블의 '핸드폰 번호', '월렛 번호' 유무를 통해 판단한다.

(10) 거래소 상품 리스트 요청

① 경로 : /RequestMarketProductList

② 요청 데이터

(JObject) requestData : {

(JObject) productFilter : {

(string) skillRank : // 스킬 랭크

(string) searchString : // 검색된 문자열

(int) pageNumber : // 클라이언트에서 보여지는 페이지 번호.

}

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

(JArray) productList : [

(JObject) [index] : {

(int) registrationNumber : // 등록된 거래 물품 목록 테이블 PrimaryKey

(string) remainingTime : // 남은 거래 시간

(int) price : // 상품 가격

(int) skillNumber : // 등록된 거래 물품 목록 테이블 ForeignKey

}

(JObject) [index] : { }, ... // 위와 같은 JObject가 반복된다.

]

}

④ 추신

- '요청 데이터'의 'productFilter'의 값에 따라서 리턴되는 값의 범위가 달라진다.

- 'skillRank' -> 'searchString'순으로 필터링을 한 후, 'pageNumber'에 맞는 데이터들을 리턴한다.

(11) 거래소 상품 구매 요청

① 경로 : /RequestPurchaseMarketProduct

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

(int) registrationNumber : // 등록된 거래 물품 목록 테이블 PrimaryKey

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

}

④ 추신

- 상품 구매 성공 시, 유저 스킬 테이블에서 스킬 정보를 추가한다.
- 구매 및 판매한 유저의 재화 정보를 갱신한다.
- 완료된 거래 목록 테이블에 거래 정보를 추가한다.
- 등록된 거래 물품 목록 테이블의 값을 삭제한다.

(12) 거래소 상품 등록 수수료 정보 요청.

① 경로 : /RequestMarketProductRegistrationFee

② 요청 데이터

(JObject) requestData : { }

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

(int) Fee : // 수수료 값.

}

④ 추신

- nft 등록 시, 필요한 수수료의 값을 명시하는 부분이다.
- 자체 코인을 사용하여 거래를 할 경우, 코인의 시세에 따라 수수료가 변경될 수도 있다.

  ㄴ 따라서 상품을 등록할 때마다 수수료 값을 서버에서 가져온다.

(13) 거래소 상품 등록 요청.

① 경로 : /RequestRegisterMarketProduct

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

(string) uniqueSkillNumber: // 유저 스킬 테이블 PrimaryKey

(int) price : // 판매하고자 하는 가격

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

}

④ 추신

- 등록에 성공하는 순간, 유저 스킬 테이블에서 해당 스킬 정보 삭제.
- 등록된 거래 물품 목록 테이블에 목록 추가.

(14) 거래소 내 등록 상품 정보 요청.

① 경로 : /RequestMyMarketRegistrationRecord

② 요청 데이터

```
(JObject) requestData : {
        (JObject) accountData : {
                (string) accountID :
                (string) accountPW :
        }
}
```

③ 응답 데이터

```
(JObject) responseData : {
        (bool) success : (default) false                    // 응답 성공 여부
        (JArray) productList : [
                (JObject) [index] : {
                        (int) registrationNumber :          // 등록된 거래 물품 목록 테이블 PrimaryKey
                        (string) remainingTime :            // 남은 거래 시간
                        (int) price :                       // 상품 가격
                        (int) skillNumber :                 // 등록된 거래 물품 목록 테이블 ForeignKey
                }
                (JObject) [index] : { }, ...                // 위와 같은 JObject가 반복된다.
        ]
}
```

④ 추신

- 등록된 거래 물품 목록 테이블에서 '등록자 속성'을 이용한다.
- 한 유저당 최대 10 개의 상품만 등록 가능. ➔ 응답해 줄 개수의 설정 필요 없음.

(15) 거래소 내 구매 상품 정보 요청.

① 경로 : /RequestMyMarketPurchaseRecord

② 요청 데이터

```
(JObject) requestData : {
        (JObject) accountData : {
                (string) accountID :
                (string) accountPW :
        }
}
```

③ 응답 데이터

```
(JObject) responseData : {
        (bool) success : (default) false                    // 응답 성공 여부
        (JArray) TransactionList : [
                (JObject) [index] : {
                        (int) TransactionNumber :           // 완료된 거래 목록 테이블 PrimaryKey
                        (string) TransactionTime :          // 거래가 수행된 시간
                        (int) price :                       // 구매 가격
                        (int) skillNumber :                 // 완료된 거래 목록 테이블 ForeignKey
                }
                (JObject) [index] : { }, ...                // 위와 같은 JObject가 반복된다.
        ]
}
```

④ 추신

- 완료된 거래 목록 테이블의 '구매자 속성'을 이용한다.

- 주기적으로 기록을 지울 계획이기에, 응답해주는 데이터의 길이는 지정하지 않아도 된다.

  ㄴ NFT 거래이기 때문에, 게임 내가 아니라 컨트렉션을 통해서도 확인이 가능하게 만들 계획이다.

(16) 거래소 내 판매 상품 정보 요청.

① 경로 : /RequestMyMarketSalesRecord

② 요청 데이터

```
(JObject) requestData : {
        (JObject) accountData : {
                (string) accountID :
                (string) accountPW :
        }
}
```

③ 응답 데이터

```
(JObject) responseData : {
        (bool) success : (default) false                    // 응답 성공 여부
        (JArray) TransactionList : [
                (JObject) [index] : {
                        (int) TransactionNumber :           // 완료된 거래 목록 테이블 PrimaryKey
                        (string) TransactionTime :          // 거래가 수행된 시간
                        (int) price :                       // 구매 가격
                        (int) skillNumber :                 // 완료된 거래 목록 테이블 ForeignKey
                }
                (JObject) [index] : { }, ...                // 위와 같은 JObject가 반복된다.
        ]
}
```

④ 추신

- 완료된 거래 목록 테이블의 '판매자 속성'을 이용한다.

- 주기적으로 기록을 지울 계획이기에, 응답해주는 데이터의 길이는 지정하지 않아도 된다.

  ㄴ NFT 거래이기 때문에, 게임 내가 아니라 컨트렉션을 통해서도 확인이 가능하게 만들 계획이다.

(17) 거래소 내 등록 상품 등록 취소 요청.

① 경로 : /RequestCancelMyMarketRegistrationProduct

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

(int) registrationNumber : // 등록된 거래 물품 목록 테이블 PrimaryKey

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

}

④ 추신

- 등록된 거래 물품 목록 테이블에서 'registrationNumber'이 속한 컬럼을 삭제한다.
- 등록된 거래 물품 목록 테이블의 '판매자 속성'과 'uniqueSkillNumber' 값을 이용하여,

ㄴ 판매자 유저의 스킬 테이블에 스킬 정보를 넣어준다.

(18) 인증 번호 요청

① 경로 : /RequestVerificationCode

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

(string) phoneNumber : // 핸드폰 번호

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

}

④ 추신

- 핸드폰 번호를 입력 받고, 해당 번호로 인증 번호를 보내는 이벤트이다.
- 핸드폰 번호를 입력 받으면, 해당 번호로 인증 번호를 보낸다. (네이버 SENS Service 이용)

(19) 인증 성공 여부 요청

① 경로 : /RequestWhetherAuthenticationSucceeded

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

(string) phoneNumber : // 핸드폰 번호

(string) verificationCode : // 인증 번호

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

}

④ 추신

- '(18)', '(19)'는 연속되는 요청이므로 클라이언트에서 핸드폰 번호를 가지고 있을 수 있다.
- 인증 성공 여부를 반환하는 이벤트이다.
- 인증 성공 시, 유저 테이블의 PrimaryKey 의 값이 'accountID'인 컬럼의 '핸드폰 번호' 속성에 값이 추가된다.

(20) 기존 계정으로 변경하기.

① 경로 : /RequestLinkExisitingAccount

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

(string) phoneNumber : // 핸드폰 번호

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

}

④ 추신

- '(18)', '(19)'는 연속되는 요청이므로 인증 성공을 전제로 한다.
- 요청 데이터의 'accountData'는 익명으로 로그인 계정 정보이다.
- 응답 데이터의 'accountData'는 기존에 핸드폰 번호를 등록 시켜 놓은 계정 정보이다.
- '핸드폰 번호 속성'은 유저 테이블의 대체키이기 때문에, 기존 계정의 ID와 PW를 가져올 수 있다.
- 성공할 시, 요청 데이터로 전달 된 'accountData'의 정보는 유저 테이블에서 삭제한다.

(21) 월렛 주소 등록 요청.

① 경로 : /RequestRegisterWalletAddress

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

(string) walletAddress :

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

}

④ 추신

- 요청 데이터로 전달 받은 'accountID'를 PrimaryKey로 갖는 행에

'핸드폰 번호' 속성이 존재해야 성공한다.

(22) 닉네임 변경

① 경로 : /RequestModifyUserNickname

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

(string) nickname :

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

}

④ 추신

3) 인 게임 Scene

(23) 유저 정보와 선택 정보 등록 요청

① 경로 : /RequestSettingGameData

② 요청 데이터

(JObject) requestData : {

(JObject) accountData : {

(string) accountID :

(string) accountPW :

}

(JObject) selectedData : {

(int) selectedCharacterNumber : 0 // 선택 캐릭터 정보

(JArray) selectedSkillNumber : [] // 선택 스킬 정보

}

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

(JObject) gameData : {

(int) recordNumber : // 게임 기록 테이블의 PrimaryKey

(float) RecordTime : (default) 0 // 게임 플레이 시간

(int) earnedScore : (default) 0 // 획득 점수

(int) earnedCoin : (default) 0 // 획득 코인

}

④ 추신

- 게임 로비에서 선택한 정보들을 게임 시작 시, 서버로 전송한다.
- 요청 데이터로 받은 데이터들을 게임 기록 테이블에 기록한다.
- 게임 기록 테이블에 기록된 컬럼의 PrimaryKey인 recordNumber를 응답 데이터로 전달한다.

(24) 캐릭터 정보 요청

① 경로 : /RequestCharacterData

② 요청 데이터

(JObject) requestData : {

(int) recordNumber :

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false // 응답 성공 여부

(JObject) characterData : { // 캐릭터 정보.

(int) characterNumber :

(float) maxHP :

(float) speed :

}

}

④ 추신

- 캐릭터 정보는 서버에도 로컬에도 존재한다. 하지만, 로컬의 데이터는 악의적인 사용자가 수정할 수 있기에 게임 내에서 사용되는 데이터는 서버에서 가져와 사용한다.

- '(23)' 요청을 수행한 후, 진행하기에 recordNumber 값을 클라이언트가 알고 있을 수 있다.

- 요청 데이터로 전달하는 'recordNumber'의 값을 통해 선택한 정보를 파악할 수 있다.

(25) 적 정보 요청

① 경로 : /RequestEnemiesData

② 요청 데이터

(JObject) requestData : {

(int) recordNumber :

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false, // 응답 성공 여부

(JArray) enemyData : [

(JObject) [index] : { // 적 정보.

(int) enemyNumber : // 몬스터 정보 테이블 PrimaryKey

(string) createType : // 몬스터 생성 방식

(float) spawnTime : // 재 생성되는 시간 간격

(float) maxHP :

(float) damage :

(float) speed :

(float) experience :

(float) score :

(float) coin :

},

(JObject) [index] : { }, ... // 위와 같은 JObject가 반복된다.

],

(Jarray) BossData : [

(JObject) [index] : { // 적 정보.

(int) enemyNumber : // 몬스터 정보 테이블 PrimaryKey

(string) createType : // 몬스터 생성 방식

(float) spawnTime : // 재 생성되는 시간 간격

(float) maxHP :

(float) damage :

(float) speed :

(float) experience :

(float) score :

(float) coin :

},

(JObject) [index] : { }, ... // 위와 같은 JObject가 반복된다.

]

}

④ 추신

- 몬스터 테이블에서 관련 정보를 찾아 전달해 준다.
- 게임 내에서 사용되는 몬스터의 정보가 많지 않기에, 사용되는 모든 값들을 한번에 가져간다.

(26) 스킬 정보 요청

① 경로 : / RequestSkillsData

② 요청 데이터

(JObject) requestData : {

(int) recordNumber :

}

③ 응답 데이터

(JObject) responseData : {

(bool) success : (default) false, // 응답 성공 여부

(JArray) selectedSkillNumber : [

(JObject) [index] : { // 스킬 정보.

},

(JObject) [index] : { }, ... // 위와 같은 JObject가 반복된다.

]

}

④ 추신

- 스킬 정보는 서버에도 로컬에도 존재한다. 하지만, 로컬의 데이터는 악의적인 사용자가 수정할 수 있기에 게임 내에서 사용되는 데이터는 서버에서 가져와 사용한다.
- '(23)' 요청을 수행한 후, 진행하기에 recordNumber 값을 클라이언트가 알고 있을 수 있다.
- 요청 데이터로 전달하는 'recordNumber'의 값을 통해 선택한 스킬 정보를 파악할 수 있다.