```
//
//  PlayGameRESTWebAPIModelAccessStrategy.swift
//  f30
//
//  Created by David on 05/02/2018.
//  Copyright © 2018 com.smartfoundation. All rights reserved.
//

import SFCore
import SFModel
import SFSocial
import SFSerialization
import SFNet
import f30Core
import f30Model

/// A strategy for accessing the PlayGame model data using a REST Web API
public class PlayGameRESTWebAPIModelAccessStrategy:
 RESTWebAPIModelAccessStrategyBase {

    // MARK: - Initializers

    private override init() {
        super.init()
    }

    public override init(connectionString: String,
                         storageDateFormatter: DateFormatter) {
        super.init(connectionString: connectionString,
                   storageDateFormatter: storageDateFormatter,
                   tableName: "PlayGames")

    }


    // MARK: - Private Methods

    fileprivate func runQuery(byID ID: String, loadRelationalTablesYN: Bool, into
     collection: ProtocolModelItemCollection, oncomplete
     completionHandler:@escaping ([String:Any]?, Error?) -> Void) {

        #if DEBUG

            if (ApplicationFlags.flag(key: "LoadPlayGamesDummyDataYN")) {

                self.selectDummy(byID: ID, loadRelationalTablesYN:
                 loadRelationalTablesYN, into: collection, oncomplete:
                 completionHandler)

                return

            }

        #endif

        // Create the dataWrapper
        let dataWrapper:            DataJSONWrapper = DataJSONWrapper()
```

```
    dataWrapper.setParameterValue(key: "\
     (PlayGameDataParameterKeys.LoadRelationalTablesYN)", value: "\
     (loadRelationalTablesYN)")

    // Create processResponse completion handler
    let processResponseCompletionHandler: (([String:Any]?, URLResponse?,
     Error?) -> Void) =
    {
        (data, response, error) -> Void in  // [weak self]

        // Call the completion handler
        completionHandler(data, error)
    }

    // Create processResponse
    let processResponse:        ((NSMutableData?, URLResponse?, Error?) ->
     Void) = self.getProcessResponse(oncomplete:
     processResponseCompletionHandler)

    // Create restApiHelper
    let restApiHelper:          RESTApiHelper = RESTApiHelper(processResponse:
     processResponse, mode: RESTApiHelperMode.CompletionHandler)

    // Get the Url
    var urlString:              String =
     NSLocalizedString("PlayGamesSelectByIDUrl", tableName:
     "RESTWebAPIConfig", comment: "")
    urlString                   = String(format: urlString, ID)

    // Call the REST Api
    restApiHelper.call(urlString: urlString, httpMethod: .POST, data:
     dataWrapper)

}

fileprivate func runQuery(byRelativeMemberID relativeMemberID: String,
 loadLatestOnlyYN: Bool, loadRelationalTablesYN: Bool, into collection:
 ProtocolModelItemCollection, oncomplete completionHandler:@escaping
 ([String:Any]?, Error?) -> Void) {

    #if DEBUG

        if (ApplicationFlags.flag(key: "LoadPlayGamesDummyDataYN")) {

            self.selectDummy(byRelativeMemberID: relativeMemberID,
             loadLatestOnlyYN: loadLatestOnlyYN, loadRelationalTablesYN:
             loadRelationalTablesYN, into: collection, oncomplete:
             completionHandler)

            return

        }

    #endif

    // Create the dataWrapper
    let dataWrapper:            DataJSONWrapper = DataJSONWrapper()
```

```
    dataWrapper.setParameterValue(key: "\
     (PlayGameDataParameterKeys.LoadLatestOnlyYN)", value: "\
     (loadLatestOnlyYN)")
    dataWrapper.setParameterValue(key: "\
     (PlayGameDataParameterKeys.LoadRelationalTablesYN)", value: "\
     (loadRelationalTablesYN)")

    // Create processResponse completion handler
    let processResponseCompletionHandler: (([String:Any]?, URLResponse?,
     Error?) -> Void) =
    {
        (data, response, error) -> Void in  // [weak self]

        // Call the completion handler
        completionHandler(data, error)
    }

    // Create processResponse
    let processResponse:        ((NSMutableData?, URLResponse?, Error?) ->
     Void) = self.getProcessResponse(oncomplete:
     processResponseCompletionHandler)

    // Create restApiHelper
    let restApiHelper:          RESTApiHelper = RESTApiHelper(processResponse:
     processResponse, mode: RESTApiHelperMode.CompletionHandler)

    // Get the Url
    var urlString:              String =
     NSLocalizedString("PlayGamesSelectByRelativeMemberIDUrl", tableName:
     "RESTWebAPIConfig", comment: "")
    urlString                   = String(format: urlString,
                                         relativeMemberID)

    // Call the REST Api
    restApiHelper.call(urlString: urlString, httpMethod: .POST, data:
     dataWrapper)

}

fileprivate func getPlayGameDataModelAdministrator() ->
 PlayGameDataModelAdministrator? {

    // Get modelAdministratorProvider
    let modelAdministratorProvider:     ProtocolModelAdministratorProvider? =
     self.delegate?.modelAccessStrategy(getModelAdministratorProvider: self)

    guard (modelAdministratorProvider != nil) else { return nil }

    // Get PlayGameDataModelAdministrator
    let pgdma:                          PlayGameDataModelAdministrator? =
     modelAdministratorProvider!.getModelAdministrator(key: "PlayGameData")
     as? PlayGameDataModelAdministrator

    return pgdma

}
```

```
// MARK: - Override Methods

public override func getRelationalDataWrapper(fromItem item:
 ProtocolModelItem) -> DataJSONWrapper? {

    let result:                     DataJSONWrapper? = DataJSONWrapper()

    // Create playGameDataWrappers
    let playGameDataWrappers:       DataJSONWrapper = DataJSONWrapper()
    playGameDataWrappers.ID = "PlayGameData"
    result?.Items.append(playGameDataWrappers)

    // Get PlayGameData item
    let pgd:                        PlayGameData? =
     self.getPlayGameDataModelAdministrator()!.collection!.getItem(propertyKey
     : "\(PlayGameDataDataParameterKeys.PlayGameID)", value: item.id) as?
     PlayGameData

    guard (pgd != nil) else { return nil }

    playGameDataWrappers.Items.append(pgd!.copyToWrapper())

    return result

}

public override func setRelationalItems(fromWrapper relationalDataWrapper:
 DataJSONWrapper, for item: ProtocolModelItem, originalID: String) {

    // Get playGameDataWrappers
    var playGameDataWrappers:   DataJSONWrapper? = nil

    // Go through each item
    for item in relationalDataWrapper.Items {

        if (item.ID == "PlayGameData") {

            playGameDataWrappers = item

        }

    }

    guard (playGameDataWrappers != nil) else { return }

    #if DEBUG

        if (ApplicationFlags.flag(key: "SaveDummyDataYN")) {

            // Go through each item
            for pgd in
             self.getPlayGameDataModelAdministrator()!.collection!.items! {

                let pgd = pgd as! PlayGameData

                pgd.playGameID  = item.id
```

```
                pgd.status      = .unmodified

            }

            return

        }

    #endif

    // Go through each item
    for pgdw in playGameDataWrappers!.Items {

        // Get OriginalID
        let oid:            String? = pgdw.getParameterValue(key:
         "OriginalID")

        guard (oid != nil) else { continue }

        // Get PlayGameData item
        let pgd:            PlayGameData? =
         self.getPlayGameDataModelAdministrator()!.collection!.getItem(id:
         oid!) as? PlayGameData

        guard (pgd != nil) else { continue }

        // Update PlayGameData
        pgd!.id             = pgdw.ID
        pgd!.playGameID     = item.id
        pgd?.status         = .unmodified

    }

}


// MARK: - Dummy Data Methods

fileprivate func selectDummy(byID ID: String, loadRelationalTablesYN: Bool,
 into collection: ProtocolModelItemCollection, oncomplete
 completionHandler:@escaping ([String:Any]?, Error?) -> Void) {

    let responseString  = NSLocalizedString("byID", tableName:
     "PlayGamesDummyRESTWebAPIResponse", comment: "")

    // Convert the response to JSON dictionary
    let data:           [String:Any]? = JSONHelper.stringToJSON(jsonString:
     responseString) as? [String:Any]

    // Process the data
    let returnData:     [String:Any]? =
     self.processRESTWebAPIResponse(responseData: data!)

    // Call the completion handler
    completionHandler(returnData, nil)

}
```

```
    fileprivate func selectDummy(byRelativeMemberID relativeMemberID: String,
     loadLatestOnlyYN: Bool, loadRelationalTablesYN: Bool, into collection:
     ProtocolModelItemCollection, oncomplete completionHandler:@escaping
     ([String:Any]?, Error?) -> Void) {

        let responseString  = NSLocalizedString("byRelativeMemberID", tableName:
         "PlayGamesDummyRESTWebAPIResponse", comment: "")

        // Convert the response to JSON dictionary
        let data:           [String:Any]? = JSONHelper.stringToJSON(jsonString:
         responseString) as? [String:Any]

        // Process the data
        let returnData:     [String:Any]? =
         self.processRESTWebAPIResponse(responseData: data!)

        // Call the completion handler
        completionHandler(returnData, nil)

    }


}

// MARK: - Extension ProtocolPlayGameModelAccessStrategy

extension PlayGameRESTWebAPIModelAccessStrategy:
 ProtocolPlayGameModelAccessStrategy {

    // MARK: - Public Methods

    public func select(byID ID: String, loadRelationalTablesYN: Bool, collection:
     ProtocolModelItemCollection, oncomplete completionHandler:@escaping
     ([String:Any]?, ProtocolModelItemCollection?, Error?) -> Void) {

        // Create completion handler
        let runQueryCompletionHandler: (([String:Any]?, Error?) -> Void) =
         self.getRunQueryCompletionHandler(collection: collection, oncomplete:
         completionHandler)

        // Run the query
        self.runQuery(byID: ID, loadRelationalTablesYN: loadRelationalTablesYN,
         into: collection, oncomplete: runQueryCompletionHandler)

    }

    public func select(byRelativeMemberID relativeMemberID: String,
     loadLatestOnlyYN: Bool, loadRelationalTablesYN: Bool, collection:
     ProtocolModelItemCollection, oncomplete completionHandler:@escaping
     ([String:Any]?, ProtocolModelItemCollection?, Error?) -> Void) {

        // Create completion handler
        let runQueryCompletionHandler: (([String:Any]?, Error?) -> Void) =
         self.getRunQueryCompletionHandler(collection: collection, oncomplete:
         completionHandler)
```

```
        // Run the query
        self.runQuery(byRelativeMemberID: relativeMemberID, loadLatestOnlyYN:
         loadLatestOnlyYN, loadRelationalTablesYN: loadRelationalTablesYN, into:
         collection, oncomplete: runQueryCompletionHandler)

    }

}
```