Project Refactoring

Team

- Seth Perry
- Sudeep Galgali
- Christina Nguyen

Title

The Third Meal

Refactoring

For our refactoring, we simplified our design by removing the key and unique item class that all objects inherited from. We also removed the redundant controllers and repository. Finally, we changed the way in which repositories will be provided to the pages that use them.

Strategy Design Pattern

We use the strategy design pattern frequently in our project. We have numerous data item repositories that are represented as interfaces, which are given concrete implementations. This is done to provide a layer of abstraction between the front-end and the backend. By adding an abstract layer, different storage types can be swapped out by replacing the concrete implementations of the repositories.

Dependency Injection

We refactored our project to use a dependency injection method for the front-end pages. Previously, the pages directly referenced the repositories that they required. Under our new system, we have introduced the IRepositoryProvider, which is responsible for providing the repositories to the page. This is the only strict dependency taken by each page. This helps to decouple the pages from the repositories they require by putting an intermediate object in between the two. Any number of repositories can be consumed by each page, and the page's constructors will not need to be modified to consume a new repository. Furthermore, this allows the possibility for lazy repository creation, where it is only generated when it is needed.

Factory Design Pattern

The Repository creation will use the RepositoryFactory, which will be responsible for instantiating new repositories based on their interface. This will be consumed by the IRepositoryProvider dependency injector, which will lazily create the repositories as they are requested.

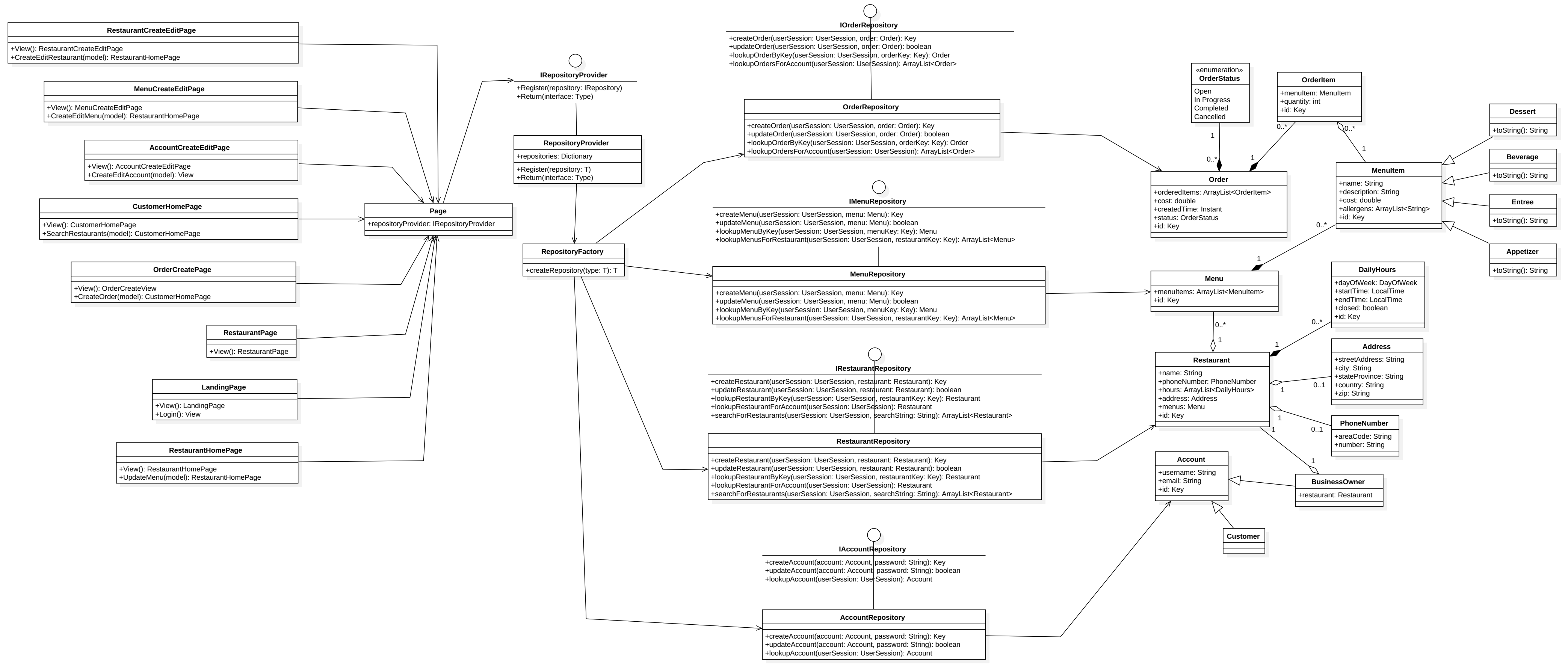
Flyweight Design Pattern

This design pattern is inherent in the IRepositoryProvider. This class will keep a dictionary of repositories. If a page requests a repository that is not found in this dictionary, a new one will be created and added to the dictionary. If the page requests a repository that is found, the repository that was found will be returned to the page.

Singleton Design Pattern

The IRepositoryProvider will also implement this design pattern, and will be shared by all pages. Only one provider per application is needed.

Model::Main


RestaurantCreateEditPage
+View(): RestaurantCreateEditPage
+CreateEditRestaurant(model): RestaurantHomePage
MenuCreateEditPage
+View(): MenuCreateEditPage
+CreateEditMenu(model): RestaurantHomePage
AccountCreateEditPage
+View(): AccountCreateEditPage
+CreateEditAccount(model): View
CustomerHomePage
+View(): CustomerHomePage
+SearchRestaurants(model): CustomerHomePage
OrderCreatePage
+View(): OrderCreateView
+CreateOrder(model): CustomerHomePage
RestaurantPage
+View(): RestaurantPage
LandingPage
+View(): LandingPage
+Login(): View
RestaurantHomePage
+View(): RestaurantHomePage
+UpdateMenu(model): RestaurantHomePage
Page
+repositoryProvider: IRepositoryProvider
IRepositoryProvider
+Register(repository: IRepository)
+Return(interface: Type)
RepositoryProvider
+repositories: Dictionary
+Register(repository: T)
+Return(interface: Type)
RepositoryFactory
+createRepository(type: T): T
IOrderRepository
+createOrder(userSession: UserSession, order: Order): Key
+updateOrder(userSession: UserSession, order: Order): boolean
+lookupOrderByKey(userSession: UserSession, orderKey: Key): Order
+lookupOrdersForAccount(userSession: UserSession): ArrayList<Order>
OrderRepository
+createOrder(userSession: UserSession, order: Order): Key
+updateOrder(userSession: UserSession, order: Order): boolean
+lookupOrderByKey(userSession: UserSession, orderKey: Key): Order
+lookupOrdersForAccount(userSession: UserSession): ArrayList<Order>
IMenuRepository
+createMenu(userSession: UserSession, menu: Menu): Key
+updateMenu(userSession: UserSession, menu: Menu): boolean
+lookupMenuByKey(userSession: UserSession, menuKey: Key): Menu
+lookupMenusForRestaurant(userSession: UserSession, restaurantKey: Key): ArrayList<Menu>
MenuRepository
+createMenu(userSession: UserSession, menu: Menu): Key
+updateMenu(userSession: UserSession, menu: Menu): boolean
+lookupMenuByKey(userSession: UserSession, menuKey: Key): Menu
+lookupMenusForRestaurant(userSession: UserSession, restaurantKey: Key): ArrayList<Menu>
IRestaurantRepository
+createRestaurant(userSession: UserSession, restaurant: Restaurant): Key
+updateRestaurant(userSession: UserSession, restaurant: Restaurant): boolean
+lookupRestaurantByKey(userSession: UserSession, restaurantKey: Key): Restaurant
+lookupRestaurantForAccount(userSession: UserSession): Restaurant
+searchForRestaurants(userSession: UserSession, searchString: String): ArrayList<Restaurant>
RestaurantRepository
+createRestaurant(userSession: UserSession, restaurant: Restaurant): Key
+updateRestaurant(userSession: UserSession, restaurant: Restaurant): boolean
+lookupRestaurantByKey(userSession: UserSession, restaurantKey: Key): Restaurant
+lookupRestaurantForAccount(userSession: UserSession): Restaurant
+searchForRestaurants(userSession: UserSession, searchString: String): ArrayList<Restaurant>
IAccountRepository
+createAccount(account: Account, password: String): Key
+updateAccount(account: Account, password: String): boolean
+lookupAccount(userSession: UserSession): Account
AccountRepository
+createAccount(account: Account, password: String): Key
+updateAccount(account: Account, password: String): boolean
+lookupAccount(userSession: UserSession): Account
«enumeration»
OrderStatus
Open
In Progress
Completed
Cancelled
OrderItem
+menuItem: MenuItem
+quantity: int
+id: Key
Order
+orderedItems: ArrayList<OrderItem>
+cost: double
+createdTime: Instant
+status: OrderStatus
+id: Key
MenuItem
+name: String
+description: String
+cost: double
+allergens: ArrayList<String>
+id: Key
Dessert
+toString(): String
Beverage
+toString(): String
Entree
+toString(): String
Appetizer
+toString(): String
Menu
+menuItems: ArrayList<MenuItem>
+id: Key
DailyHours
+dayOfWeek: DayOfWeek
+startTime: LocalTime
+endTime: LocalTime
+closed: boolean
+id: Key
Restaurant
+name: String
+phoneNumber: PhoneNumber
+hours: ArrayList<DailyHours>
+address: Address
+menus: Menu
+id: Key
Address
+streetAddress: String
+city: String
+stateProvince: String
+country: String
+zip: String
PhoneNumber
+areaCode: String
+number: String
Account
+username: String
+email: String
+id: Key
BusinessOwner
+restaurant: Restaurant
Customer
1
0..*
0..*
1
0..*
1
0..*
1
0..*
1
0..*
1
1
0..1
1
0..1
1
1