**REVIEW**

1. **Описание структуры приложения**

Приложение разработано на С# c применением ASP.NET WebForms.

При работе с выбранной технологией любой запрос обрабатывается одним конкретным обработчиком HttpHandler. Исходя из этого реализация **FrontController** возможна с помощью собственного HttpHandler, заменяющего обработчик по умолчанию. Заменяя обработчик по умолчанию другим, мы теряем функциональность, такую как генерацию разметки ASP.NET. Такая замена не позволяет использовать элементы управления ASP.NET WebForms. Поэтому попытка реализовать **FrontController** представлена **ResultFrontController**, где HttpHandler реализован для одной конкретной страницы result.aspx, где нет элементов управления.

Данные сессии пользователя хранятся в специальном объекте ***Session***. В ранних версиях приложения реализован механизм меток синзронизации.

**ResultFrontCоntroller** вызывает соответствующий метод **GenerateView**() класса **TransformView**. Обработчик GenerateView() генерирует разметку страницы result.aspx на основе **xml** и **xslt**. Остальные страницы представляют собой веб-формы с элементами управления ASP.NET и обрабатываются стандартным HttpHandler, не использующим xslt.

Система фильтров реализована на основе класса **HttpModule**. В этом модуле есть 2 цепочки фильтров (пре- и постфильтры). Фильтры реализованы с помощью шаблона **Decorator**. Каждый фильтр наследуется от базового класса **FilterBase**, реализующего созданный интерфейс **IFilter**.

Пользовательский интерфейс (GUI) создан на основе нескольких связных страниц ASP.NET WebForms. Главная форма **main** связывает все остальные, а именно: формы создания, редактирования и удаления записей и форма просмотра. Создание новой записи о клиенте основано на последовательности из трех форм ввода данных.

Операции с записями о клиентах (добавление, удаление, редактирование) реализованы при помощи набора команд. Команды реализованы в виде классов, реализующих созданный интерфейс **ICommand**.

При запуске приложения в коде обработчиков событий уровня приложения (Global.asax) создается новая сессия, после чего инициализируется экземпляр класса **UnitOfWork**.

Для контролем в за всеми изменениями данных, которые мы производим с доменной моделью в рамках бизнес-транзакции, используется шаблон **Unit of Work**. При этом в классе UnitOfWork реализованы методы: Сommit, Rollback, RegisterNew, RegisterDeleted, RegisterDirty.

За вызов Commit отвечает команда **CommitCommand**, изменяющая модель приложения. Вызовы вышеизложенных методов UnitOfWork осуществляются в классе **Customer** при помощи соответствующих функций, помечающих объект Customer новым, измененным или удаленным (MarkNew(), MarkDirty() и MarkRemoved(), соответственно).

В методе Commit класса UnitOfWork вызываются требуемые методы класса , отвечающего за операции доступа и изменения данных, хранимых в XML.

1. **Анализ приложения**

Приложение было проанализировано с помощью программы NDepend.

В NDepend заложено 130 правил проверки кода :

* Code Quality
* Object Oriented Design
* Design, Architecture and Layering
* Dead Code
* Visibilty
* Purity –Immutability–Side-Effects,
* Naming Conventions.

При первом запуске аналитической программы получили результат 24/106.

Затем был произведен рефакторинг, итог – 13/117.

Обратим внимание на значения метрик в отредактированном коде:

1. **Types Metrics : Code Quality**

* **Afferent Coupling at type level (Ca):** The Afferent Coupling for a particular type is the number of types that depends directly on it.

*Все типы, кроме 3х не привышают норму, 3 – привышеные незначитальное.*

* **Cyclomatic Complexity -** The CC metric is language dependent

*UnitOfWork, DAO, Customer привышают норму незначительно, все остальные классы удоблетворяют норме.*

#### Types Metrics : Code Members and Inheritance

* **Depth of Inheritance Tree (DIT):** The Depth of Inheritance Tree for a class or a structure is its number of base classes (including the*System.Object* class thus DIT >= 1).

*Глубина 1-4 уровня*

#### Types Metrics : Lack Of Cohesion Of Methods and Association Between Classes

* **Lack of Cohesion Of Methods (LCOM):** The single responsibility principle states that a class should not have more than one reason to change.
* LCOM takes its values in the range [0-1]. The LCOM HS takes its values in the range [0-2].
  + LCOM = 1 – (sum(MF)/M\*F)
  + LCOM HS = (M – sum(MF)/F)(M-1)

Where:

* M is the number of methods in class (both static and instance methods are counted, it includes also constructors, properties getters/setters, events add/remove methods).
* F is the number of instance fields in the class.
* MF is the number of methods of the class accessing a particular instance field.
* Sum(MF) is the sum of MF over all instance fields of the class.

#### The underlying idea behind these formulas can be stated as follow: a class is utterly cohesive if all its methods use all its instance fields, which means that sum(MF)=M\*F and then LCOM = 0 and LCOMHS = 0.

#### *Все оценки LCOM и LCOM HS в пределах указанных норм, лишь LCOM BaseFilter =1.*

#### Association Between Class (ABC): The Association Between Classes metric for a particular class or structure is the number of members of others types it directly uses in the body of its methods.

#### *Этот показатель в пределах норамы для всех, кроме UoF, DAO, Update, но отклонения незначительные.*

#### Assemblies Metrics

#### 

*При разработке дизайна необходимо стремиться к высокой cohesion и низкой coupling, что в данном случае выполняется*

1. **Демонстрация SAP** 