Курс: Разработка приложений с использованием WCF

ЗАДАНИЕ №4

Надо модифицировать службу Converter, созданную в предыдущем домашнем задании. Теперь служба не должна хоститься в консольном приложении, а должна быть создана в виде dll. Для хостинга такой службы пока можно использовать Visual Studio.

Подумайте, надо ли изменять клиента этой службы, созданного в предыдущем домашнем задании?

Сделайте так, чтобы этот клиент начал работать с нашей службой. Пришлось что-то изменять в клиенте или нет?

Курс: Разработка приложений с использованием WCF

ЗАДАНИЕ №5

Создать простую WCF службу, использующую дуплексный контракт. Контракт службы должен быть создан в интерфейсе и содержать методы:

```
void  Login(string name);
void  SendText(string text);
void  Logout(string name);
```

В callback контракте должен находиться метод:

```
void  TextForUsers(string text);
```

Сначала клиент должен зарегистрироваться на службе, вызвав метод ***Login()*** и передав в нем свое имя. Затем служба сможет получать от зарегистрированного клиента какое-либо строковое значение, когда клиент вызовет метод ***SendText()***. Клиент может отключиться от службы вызовом метода ***Logout()***. Служба не должна принимать текст от клиента, который не выполнил регистрацию *(не вызвал метод Login()).*

С помощью дуплексного контракта служба должна будет пересылать текст, полученный от зарегистрированного клиента всем другим зарегистрированным клиентам, которые в этот момент подключены к службе.Служба должна быть создана в виде dll. Для хостинга такой службы пока можно использовать Visual Studio. Клиент должен быть WinForms или WPF приложением.

Домашнее задание №5

На стороне клиента создайте прокси-класс. Прокси-класс создается командой «Добавить ссылку на службу».

Создайте приятный интерфейс для ввода значений и для вывода полученных от службы результатов в окне клиентского приложения.

Протестируйте созданные службу и клиента. Запустите службу и три копии клиента. Зарегистрируйте каждого клиента на службе, для чего вызовите от имени клиента метод ***Login().*** Затем одним из клиентов передайте на службу строку текста, вызывав метод ***SendText().*** Убедитесь, что все клиенты получили этот текст обратно от службы. Выполните одним из клиентов выход *(вызовите метод Logout()),* но не закрывайте окно этого клиента. Снова, каким-нибудь из двух оставшихся активных клиентов отправьте на службу другой текст. Убедитесь, что два клиента получили этот текст от службы, а отключившийся не получил.

Курс: Разработка приложений с использованием WCF

ЗАДАНИЕ №6

Создать WCF службу с дуплексным контрактом, условно называемую «Банкомат». Контракт службы должен быть создан в интерфейсе и содержать методы:

```
int  CreateAccount(int m);
void  Withdraw(int m, int number);
int  GetAccountInfo(int number);
```

В callback контракте должен находиться метод:

```
int  Balance();
```

Кроме этих методов служба должна еще хранить создаваемые аккаунты клиентов. Как это делать, придумайте самостоятельно.

Метод ***CreateAccount()*** предназначен для создания клиентского аккаунта. В этом методе клиент передает в параметре m сумму денег, которую он хочет хранить в создаваемом аккаунте. Служба создает новый аккаунт, заносит на его баланс сумму m и возвращает клиенту номер созданного аккаунта. Теперь клиент сможет снимать деньги со своего аккаунта, вызывая метод ***Withdraw(m, number***) и передавая в параметре m сумму, которую он хочет снять, а в параметре number — номер своего аккаунта. В любой момент клиент может посмотреть свой текущий баланс, вызвав метод ***GetAccountInfo(number),*** который вернет сумму денег для аккаунта с именем number.

Домашнее задание №6

Для чего нужен callback метод ***Balance()***? Этот метод должен вызываться службой для клиента, который только что выполнил вызов метода ***CreateAccount()*** или ***Withdraw()*** и отображать в окне клиента текущий баланс клиента. Т.е., клиент создал аккаунт, положил туда m денег и служба вызывает для этого клиента метод Balance(), сообщая его текущий баланс. Клиент снял какую-то сумму денег со своего аккаунта и служба снова присылает ему его измененный баланс.

Клиент должен быть WinForms или WPF приложением. На стороне клиента создайте прокси-класс. Прокси-класс создается командой «Добавить ссылку на службу».

Для простоты договоримся, что клиент не может пополнять свой баланс. Аккаунт автоматически уничтожается, когда его баланс становится равным нулю. При снятии денег необходимо проверять доступность требуемой клиентом суммы.

И самое главное. В данном приложении все клиенты должны работать с одним и тем же экземпляром службы.