**Лекция 5**

#### Введение

CORBA представляет собой технологию построения распределенных систем, стандартизованную и поддерживаемую консорциумом OMG. Эта технология предоставляет механизмы для описания спецификаций интерфейсов взаимодействующих систем (предполагается, что они могут быть реализованы на различных языках программирования, с использованием различных платформ, и функционировать в различной сетевой инфраструктуре), а также обеспечивает некоторые важные сервисы - такие как сервисы именования, событий, транзакций, безопасности и т.д. Несмотря на то, что CORBA использует объектную идеологию, она может применяться в системах программирования, не поддерживающих объектный подход.

CORBA означает *Common Object Request Broker Architecture* (общая архитектура брокера объектных запросов). CORBA - это своего рода "клей", интегрирующая технология. Она позволяет программам на разных языках, работающих в различных узлах сети, взаимодействовать друг с другом так же просто, как если бы они находились в адресном пространстве одного процесса. Задача CORBA сделать возможной интеграцию изолированных систем. Клиенты, работающие с CORBA, получают преимущества в трех областях: прозрачность вызова ( invocation transparency ), прозрачность реализации ( implementation transparency ) и прозрачность локализации ( location transparency ). Прозрачность - основная цель CORBA. Для решения этой задачи CORBA использует все важнейшие достижения объектной технологии: инкапсуляцию, наследование, полиморфизм, динамическое связывание.

Прозрачность вызова определяет точку зрения клиента, посылающего сообщение серверу. Язык, используемый при реализации клиента, определяет, каким образом вызываемый объект получает сообщение, как параметры передаются в стек, как осуществляется обращение к методам данного объекта, как передаются возвращаемые значения. Если сервер создан в соответствии с правилами CORBA, то клиент может вызывать методы этого совместимого с CORBA объекта так, как и любого другого объекта, реализованного в используемом языке программирования. CORBA поддерживает ряд языков программирования (всего их 10), включая C, C++, Java, COBOL, Ada, Lisp, Smalltalk, PL/1, Python, IDLscript.

*Прозрачность реализации* - это инкапсуляция, примененная к распределенным системам. Клиенту о вызове метода известно три вещи: имя метода, параметры метода (если они есть) и тип возвращаемого значения. Клиент может предоставить серверу информацию относительно любых своих ограничений до того, как вызывать методы. Клиент вызывает метод средствами языка, используемого при разработке клиента, а код, вызываемый для выполнения метода, реализуется на любом языке, который поддерживают отображения OMG.

*Прозрачность локализации* позволяет клиенту вызывать CORBA -совместимый код, который может выполняться в любом узле сети. Прозрачность локализации основана на свойстве прозрачности реализации - клиенты вроде бы делают локальные вызовы, но на самом деле они вызывают код, написанный на разных языках и выполняющийся за пределами адресного пространства процесса клиента. Все это возможно благодаря концепциям объектной технологии (инкапсуляции, наследованию и полиморфизму), которые используются независимым от реализации образом.

Ответственным за CORBA является консорциум *Object Management Group* ( OMG ). Отличительным признаком CORBA по сравнению с другими технологиями распределенных систем является архитектура *Object Management Architecture* ( OMA ). ОМА (в качестве эталонной архитектуры) описывает систему, которая состоит из взаимодействующих сервисов, решающих прикладную задачу. Уровней модульности у такого решения больше, но концепция объектов (или компонентов), совместно работающих для достижения общей цели, остается той же самой.

Чтобы объекты могли взаимодействовать так, как если бы они "обращались" на одном языке, они должны использовать общее для языков, на которых они реализованы, внешнее представление. Это внешнее представление - промежуточный язык - должно быть простым в использовании, легким в изучении и обусловливать минимальные накладные расходы. Таким промежуточным языком является язык *OMG Interface Definition Language*,обычно называемый просто IDL.

IDL позволяет разработчикам описывать интерфейсы для тех типов данных, которые используются удаленно, независимо от языка реализации. IDL - это чисто описательный язык, IDL -файлы не включают никаких деталей реализации. IDL -компилятор принимает IDL -файл (этот файл имеет расширение **.idl)** и генерирует код, необходимый клиенту для вызова CORBA -совместимого объекта и CORBA -совместимому объекту - чтобы принимать и возвращать значения. Код, который реализует операции (т.е. обрабатывает запросы клиента к CORBA -объекту), называется сервантом (servant). В языке, не являющемся объектно-ориентированным, каждая операция реализуется отдельно. В объектно-ориентированном языке в одном серванте могут быть реализованы все необходимые операции. Клиенту ничего не известно об особенностях реализации, поэтому сервант может существовать только во время вызова метода, до тех пор, пока не прекратит работу сервер, на котором выполняется данный сервант, или в какое-то промежуточное время (время существования объекта контролируется объектным адаптером, через который проходят все обращения к сервантам).

IDL -компилятор создает для каждого компилируемого IDL -файла целый ряд файлов, ориентированных на конкретные языки программирования. Проекту потребуется столько IDL -компиляторов, сколько языков программирования используется при разработке системы. Для создания на Java интерфейса к унаследованной COBOL -системе потребуется два IDL -компилятора: один - для создания клиентской программы на языке Java, второй - для создания серверной программы на языке COBOL. Например, внешнему интерфейсу на языке Java потребуются следующие файлы, сгенерированные на языке Java, для взаимодействия с серверной системой, которая называется **BillingService**:

BilingService.java

BillingServiceOperations.java

BillingServiceHolder.java

BillingServiceHelper.java

\_ BillingServiceStub.java

**BilingService.java** содержит основное определение удаленного сервера. Исходный файл **BillingServiceOperations** содержит определения методов (операций), поддерживаемых CORBA -объектом. Класс **BillingServiceStub** отвечает за взаимодействие на стороне клиента, а файл **BillingServiceHelper** определяет класс, используемый клиентом для выполнения задач, которые связаны с CORBA.

Эти файлы скрывают механизм реализации вызова от вызывающей программы. **BillingService.java** и **BillingServiceOperations.java** определяют интерфейсы для использованиями классами, связанными с реализацией; **\_BillingServiceStub.java** определяет код CORBA -вызова для осуществления вызовов удаленных методов на сервере. Файлы **BillingServiceHelper.java** и **BillingServiceHolder.java** - вспомогательные служебные файлы.

Поставщик серверного компонента предоставляет IDL -файл, описывающий API своего серверного объекта. Используя этот файл и IDL -компилятор, разработчик может создать клиентские файлы, необходимые для непосредственного обращения к этому серверному объекту.

Если бы в написанном на С или любом другом языке графическом пользовательском интерфейсе системы **BillingService** потребовалось бы посылать сообщения серверным объектам, написанным на Java ( CORBA -совместимые Java -объекты), с помощью IDL -компилятора потребовалось бы создать следующие файлы, чтобы серверные Java -объекты могли получать сообщения от любых вызывающих программ:

BillingService.java

BillingServiceOperations.java

BillingServicePOA.java

Файлы **BillingService.java** и **BillingServiceOperations.java** - те же, что и раньше (интерфейсы для использования классами, связанными с реализацией), а **BillingServicePOA.java** представляет собой код, используемый сервером для приема вызовов методов и передачи возвращаемых значений через сеть.

Дополнительно к объявлению Java -интерфейсов, применяемых разработчиками, и клиентские, и серверные файлы, определенные выше, реализуют базовые классы *посредников (proxies)* - "заместителей" других объектов. Посредники создают для клиента видимость того, что он посылает сообщение одному объекту, в то время как на самом деле этот клиент посылает сообщение другому объекту. CORBA определяет двух взаимосвязанных посредников: *заглушку (stub)* и *скелет (skeleton)*.Заглушка является посредником на стороне клиента, скелет - на стороне сервера. Оба посредника скрывают использование ORB и от клиента, и от сервера.

*Object Request Broker* - это важнейший элемент технологии CORBA, связанный с брокером объектных запросов. Основным назначением ORB является функциональная совместимость. Все CORBA -совместимые объекты должны применять какой-либо брокер объектных запросов для передачи или приема запросов методов, но сами объекты редко имеют дело непосредственно с брокером объектных запросов. Способность к взаимодействию, заложенная в CORBA, остается одной и той же независимо от конкретной реализации CORBA.

Когда клиент запрашивает операцию, реализуемую распределенным объектом, брокер объектных запросов данного клиента для выполнения вызова использует объектную ссылку. Объектная ссылка, используемая клиентом (поддерживаемая открытым API вызываемого объекта, описанным в IDL -файле), содержит непрозрачную сетевую ссылку (непрозрачную - так как ни клиент, ни сервер не могут извлечь из этой ссылки информацию о деталях реализации). Клиент запрашивает операцию, реализуемую распределенным объектом посредством объектной ссылки, поскольку *Interoperable Object Reference* ( IOR ) ссылается именно на объектную ссылку, структура которой хорошо известна брокерам ORB при условии использования OMG -совместимых протоколов (например, IIOP ). Источником объектной ссылки является объектный адаптер, указывающий на удаленный объект. Объектная ссылка содержит три важных фрагмента информации: местоположение распределенного объекта; ссылку на адаптер, создавший объектную ссылку; идентификатор объекта серванта. Все CORBA -совместимые объектные брокеры "знают", что такое объектные ссылки и как их использовать для установления соединений клиентов с сервантами. Последнее, что необходимо для обеспечения функциональной совместимости, - это создание и синтаксический анализ IOR в протоколе для отправки запроса через сеть. Чтобы сделать возможным надежное и уверенное взаимодействие брокеров объектных запросов различных производителей, консорциум OMG разработал спецификацию протокола *Internet Inter ORB Protocol* ( IIOP ). IIOP - это протокол обмена сообщениями между объектными брокерами.