Учебный курс

# Технологии и средства разработки корпоративных систем

Лекция 6

## Создание распределенных приложений по технологии Remoting

Лекции читает

**кандидат технических наук, доцент**

**Зыков Сергей Викторович**

# Содержание

# Общий принцип

**Объекты и/или модули распределенного приложения располагаются в нескольких процессах, выполняющихся на нескольких компьютерах**

# Основные понятия

**Web-services**: слабосвязная (loose coupled) независимая от операционной системы и языка программирования модель для работы с объектами без внутреннего состояния (stateless), основанная на общедоступных стандартах.

**.NET Remoting**: сильносвязная (tightly coupled) модель для эффективного взаимодействия объектов среды выполнения .NET, расположенных в разных процессах разных компьютеров.

# Распределенные приложения: терминология

**Loose coupled**: слабосвязная распределенное приложение опирается на минимальный набор требований для участвующих сторон. Используются широко распространенные протоколы (XML, HTTP) и самодостаточные наборы данных.

**Tightly coupled**: сильносвязное распределенное приложение подразумевает определенную однородность частей приложения и основано на заранее согласованных протоколах и наборах данных

**Stateless**: модель взаимодействия распределенных объектов без внутреннего состояния. Можно считать, что каждый вызов метода обрабатывает новый экземпляр объекта.

# Низкоуровневые средства для работы с сетью

System namespace

Net namespace

```
class EndPoint {…};
class IPEndPoint {…};
Class IrDAEndPoint {…};
class Dns {…}
…
```

Sockets namespace

```
class NetworkStream {…};
class Socket {…};
class TcpClient {…};
class TcpListener {…};
…
```

# Распределенные приложения – 1980-е (1):

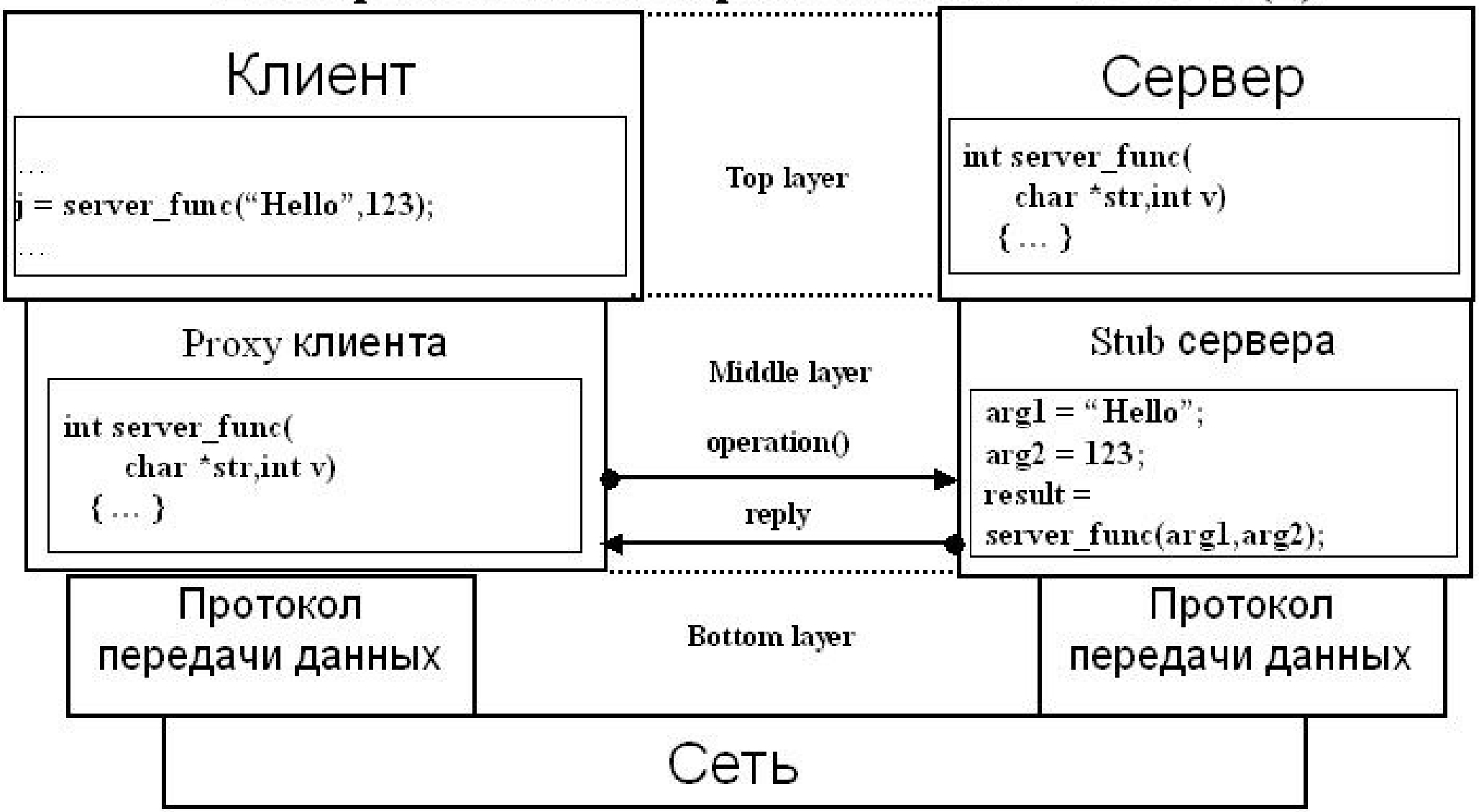
- ✓ **Передача сообщений** DCE

  (Distributed Computing Environment)

- ✓ **Удаленный вызов процедур**

  (Remote Procedure Call)

# Распределенные приложения – 1980-е (2)



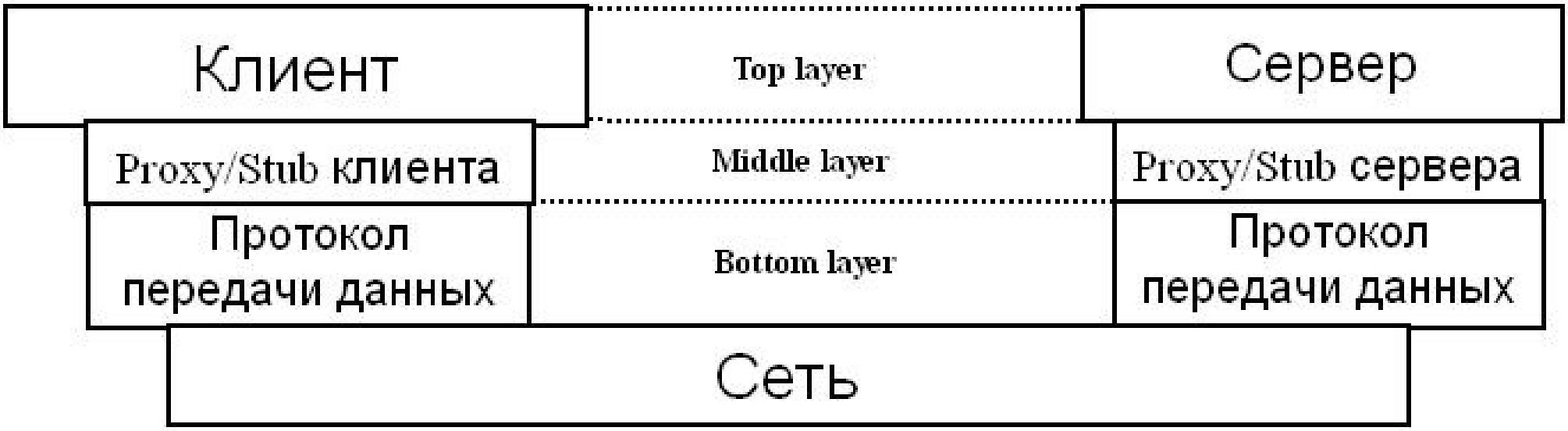
# Proxy и stub для удаленного вызова процедур – RPC(1)

✓ Proxy - подпрограмма, которую может вызывать клиент. Proxy передает серверу запрос на вызов подпрограммы с заданными параметрами и ожидает ответа, затем возвращает результат клиенту. Вызов proxy не отличается от вызова локальной подпрограммы.

✓Stub - выполняющийся на сервере код. Получает запрос вызова заданной подпрограммы с заданными параметрами. Вызывает подпрограмму и отправляет результат по сети.

✓Proxy и stub создаются автоматически. Но для этого требуется язык описания интерфейсов клиента/сервера. Пример такого языка - IDL.

# Proxy и stub для удаленного вызова процедур – RPC(2)



# Объектные RPC – 1990-е (1)

- ✓ Скрывают различие между объектами в разных средах и языках
- ✓ Наиболее распространены DCOM и CORBA
- ✓ ORB = Object Request Broker
  - ✓ Получает указание вызвать заданный метод для заданного объекта
    - ✓ Находит объект, расположенный на некотором компьютере, вызывает указанный метод и возвращает результат клиенту
      - ✓ Клиент ничего не знает от языке и платформе, где расположен запрашиваемый объект

# Объектные RPC – 1990-е (2)

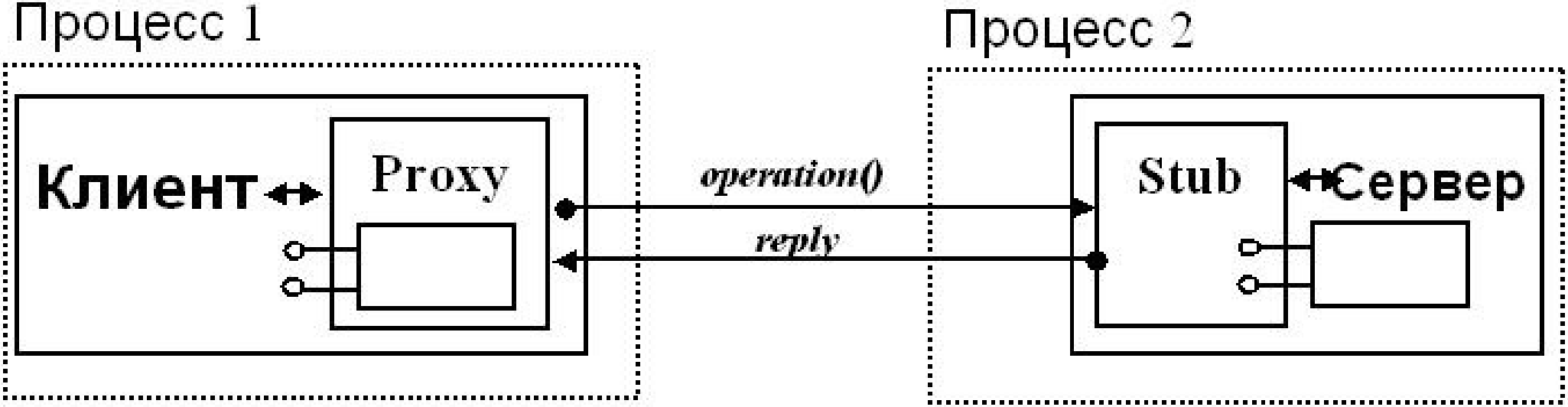
# Proxy и stub в объектных RPC (1)

✓ **Proxy** - для клиента выглядит как обычный объект. Но на самом деле он просто упаковывает параметры вызова и передает (marshal) упакованные параметры серверу. Ждет ответа от сервера, распаковывает его и передает клиенту.

✓**Stub** - получает и распаковывает параметры вызова (unmarshall) и вызывает требуемую функцию или метод на сервере. Упаковывает ответ и отправляет клиенту.

✓**Marshalling/unmarshalling** - передача объектов (в частности, параметров вызова) через границу процесса.

# Proxy и stub в объектных RPC (2)

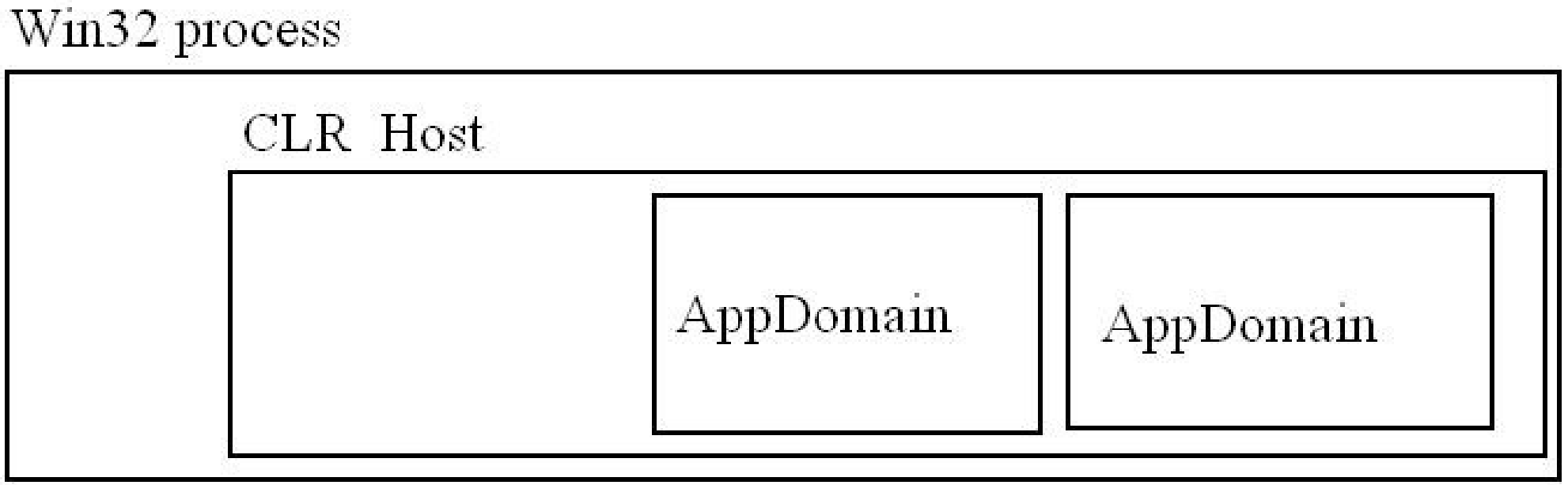


# Границы процессов и домены приложений (1)

✓ При выполнении программы, написанной для среды .NET, загрузчик сначала создает хост CLR (“виртуальную машину”) и домен приложения (AppDomain) по умолчанию, в который загружает требуемые сборки и передает управление.

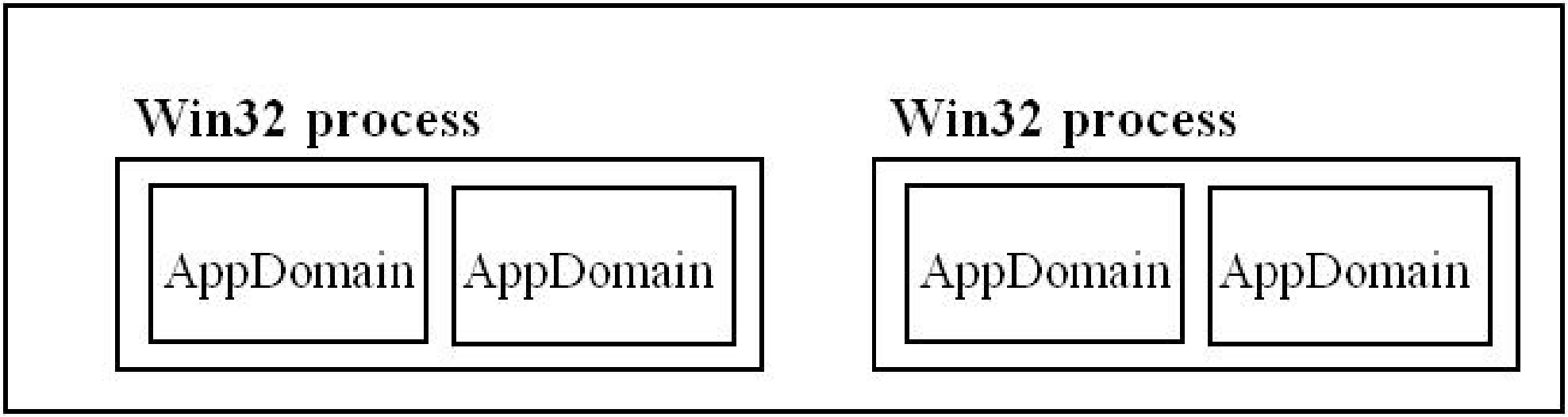
✓ В некоторых случаях, в одном процессе может быть несколько доменов приложений. Средства среды выполнения CLR обеспечивают совершенно независимое выполнение программ в нескольких AppDomain одного процесса.

# Границы процессов и домены приложений (2)

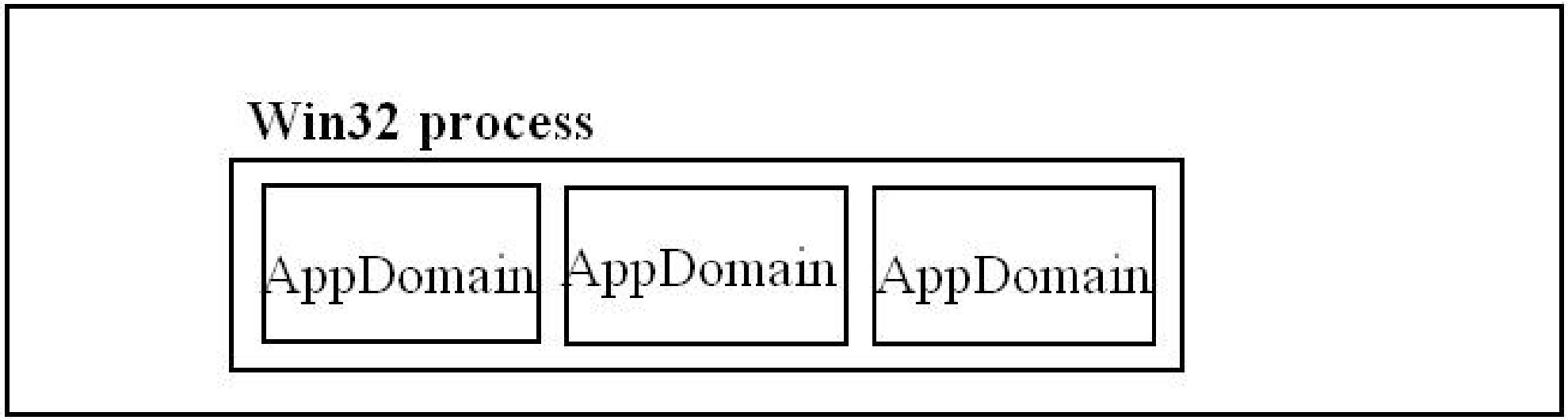


# Процессы и домены приложений (1)

*Компьютер*



# Процессы и домены приложений (2)



# Работа с удаленными объектами: `value` и `ref`

- ✓ `MarshalByValue`: передать от сервера к клиенту объект целиком. Вызывать методы объекта на стороне клиента.
  - ✓ Целесообразно для простых, редко изменяющихся во времени объектов.
- ✓ `MarshalByRef`: передавать от сервера к клиенту параметры вызываемого метода и вызывать методы объекта на стороне сервера.
  - ✓ Объект содержит ссылки на системные ресурсы, специфичные для процесса/компьютера.
  - ✓ Объект содержит ссылки на большое количество других объектов на стороне сервера
  - ✓ Объект часто изменяет свое состояние

# Явное создание объекта (1)

✓ Объект на сервере создается явно.

```
ServerObject obj = new ServerObject();
RemotingServices.Marshal(obj,"srvobj");
```

✓ Объект на сервере существует пока на него есть ссылки,

в т.ч. пока не вызван метод
`RemotingServices.Disconnect(Object obj)`

# Явное создание объекта (2)

✓ Клиент может получить ссылку на объект двумя способами:

```
ServerObject obj = (ServerObject)Activator.GetObject(
                   typeof(ServerObject),
                   "http://localhost:8080/srvobj");
```

```
RemotingConfiguration.RegisterWellKnownClientType(
                typeof(ServerObject),
                "http://localhost:8080/srvobj");
               …
ServerObject obj = new ServerObject();
```

# Объекты, активируемые сервером (1)

✓ Момент создания объекта определяется сервером. В этом случае имя присваивается не экземпляру объекта, а типу объекта.

✓ Для обработки каждого вызова удаленного метода может создаваться свой экземпляр объекта.

```
RemotingConfiguration.
    RegisterWellKnownServiceType(typeof(ServerObject),
                "srvobj",
                 WellKnownObjectMode.SingleCall);
```

# Объекты, активируемые сервером (2)

✓ Все вызовы удаленного метода может обрабатывать

```
RemotingConfiguration.
    RegisterWellKnownServiceType(typeof(ServerObject),
                "srvobj",
                 WellKnownObjectMode.Singleton);
```

✓ Клиент работает с удаленным объектов полностью аналогично предыдущему случаю

# Объекты, активируемые клиентом (1)

✓ Момент создания объекта определяется клиентом - на сервере может быть создано много объектов. В этом случае сервер объектов однозначно определяется именем

```
RemotingConfiguration.
    RegisterActivatedServiceType(typeof(ServerObject));
```

# Объекты, активируемые клиентом (2)

✓ Клиент должен зарегистрировать тип объекта как расположенный на сервере, затем создавать объекты (на самом деле - proxy) при помощи оператора `new`:

```
RemotingConfiguration.RegisterActivatedClientType(
            typeof(ServerObject),
            "http://localhost:8080");
…
ServerObject obj = new ServerObject();
ServerObject obj2 = new ServerObject();
```

# Распределенная «сборка мусора» (1)

✓ Классический распределенный сборщик мусора подсчитывает количество ссылок на серверный объект, Для этого все клиенты время от времени опрашиваются.

✓ .NET Remoting использует другую схему – механизм «аренды». Каждому серверному объекту ставится в соответствие объект, предоставляющий интерфейс `ILease`:

# Распределенная «сборка мусора» (2)

```
public class MarshalByRefObject {
    …
    virtual object InitializeLifetimeService();
    …
}

interface ILease {
    …
    TimeSpan InitialLeaseTime { get; set; }
    TimeSpan RenewOnCallTime { get; set; }
    void Renew();
    void Register(ISponsor sponsor);
    …
}
```

# Библиография (1)


- Федоров А. Продукты и технологии Microsoft 2006.– М.: «Русская редакция», 2005.– 126 с.
- Мюррей К. Новые возможности системы Office 2007.– М.: «ЭКОМ», 2007.– 256 с.
- Зыков С.В. Проектирование корпоративных порталов.– М.: МФТИ, 2005.– 258 с.
- Технологическая платформа Microsoft .NET: www.microsoft.com/net

# Библиография (2)

1. http://msdn.microsoft.com/net
2. Nathan A. .NET and COM: The Complete Interoperability Guide. Sams, 2002, 1608 pp.
3. Box D. Essential .NET, Vol.1: The Common Language Runtime. Addison Wesley, 2002, 432 pp.
4. Grimes F. Microsoft .NET for Programmers. Manning Publications, 2002, 386 pp.
5. J. Richter. Applied Microsoft .NET Framework Programming. Microsoft Press, 2002, 556 pp.
6. Зыков С.В. Проектирование корпоративных порталов.– М.: МФТИ, 2005.– 258 с.

# Благодарю за внимание!

Вопросы?


- http://zykov.altweb.ru
- szykov@hotmail.com
- sergey.zykov@tekama.com