Информационные технологии в корпоративных сетях

- Лектор – доцент Шеховцов Олег Иванович

- E-mail: clarahena@mail.ru

Информационные технологии в корпоративных сетях

- Раздел вводный

- **Введение в корпоративные сети**

Информационные технологии в корпоративных сетях

- **Введение в корпоративные сети**
- Корпоративная сеть - это сложная система, включающая тысячи самых разнообразных компонентов: компьютеры разных типов, начиная с настольных и кончая мейнфремами, системное и прикладное программное обеспечение, сетевые адаптеры, концентраторы, коммутаторы и маршрутизаторы, кабельную систему.

Введение в корпоративные сети

- Основная задача системных интеграторов и администраторов состоит в том, чтобы эта громоздкая и весьма дорогостоящая система как можно лучше справлялась с обработкой потоков информации, циркулирующих между сотрудниками предприятия и позволяла принимать им своевременные и рациональные решения, обеспечивающие выживание предприятя в жесткой конкурентоной борьбе.

Введение в корпоративные сети

- А так как жизнь не стоит на месте, то и содержание корпоративной информации, интенсивность ее потоков и способы ее обработки постоянно меняются. Последний пример резкого изменения технологии автоматизированной обработки корпоративной информации связан с беспрецедентным ростом популярности Internet в последние несколько лет.

Введение в корпоративные сети

- Изменения, причиной которых стал Internet, многогранны. Гипертекстовая служба WWW изменила способ представления информации человеку, собрав на своих страницах все популярные ее виды - текст, графику и звук. Транспорт Internet - недорогой и доступный практически всем предприятиям - существенно облегчил задачу построения территориальной корпоративной сети,

Введение в корпоративные сети

- одновременно выдвинув на первый план задачу защиты корпоративных данных при передаче их через в высшей степени общедоступную публичную сеть с многомиллионным "населением".

Информационные технологии в корпоративных сетях

- Защита информации в корпоративных сетях- **технология VPN**
- В современной практике для организации корпоративной сети между рассредоточенными подразделениями одной организации используется технология VPN (англ. Virtual Private Network — виртуальная частная сеть).

Технология VPN

- **VPN** — обобщённое название технологий, позволяющих обеспечить одно или несколько сетевых соединений (логическую сеть) поверх другой сети (например, Интернет). Суть её заключается в создании безопасной и защищённой сети поверх, как правило, публичной ненадёжной сети.
- Фактически, эта технология немногим отличается от понятия локальной сети.

Технология VPN

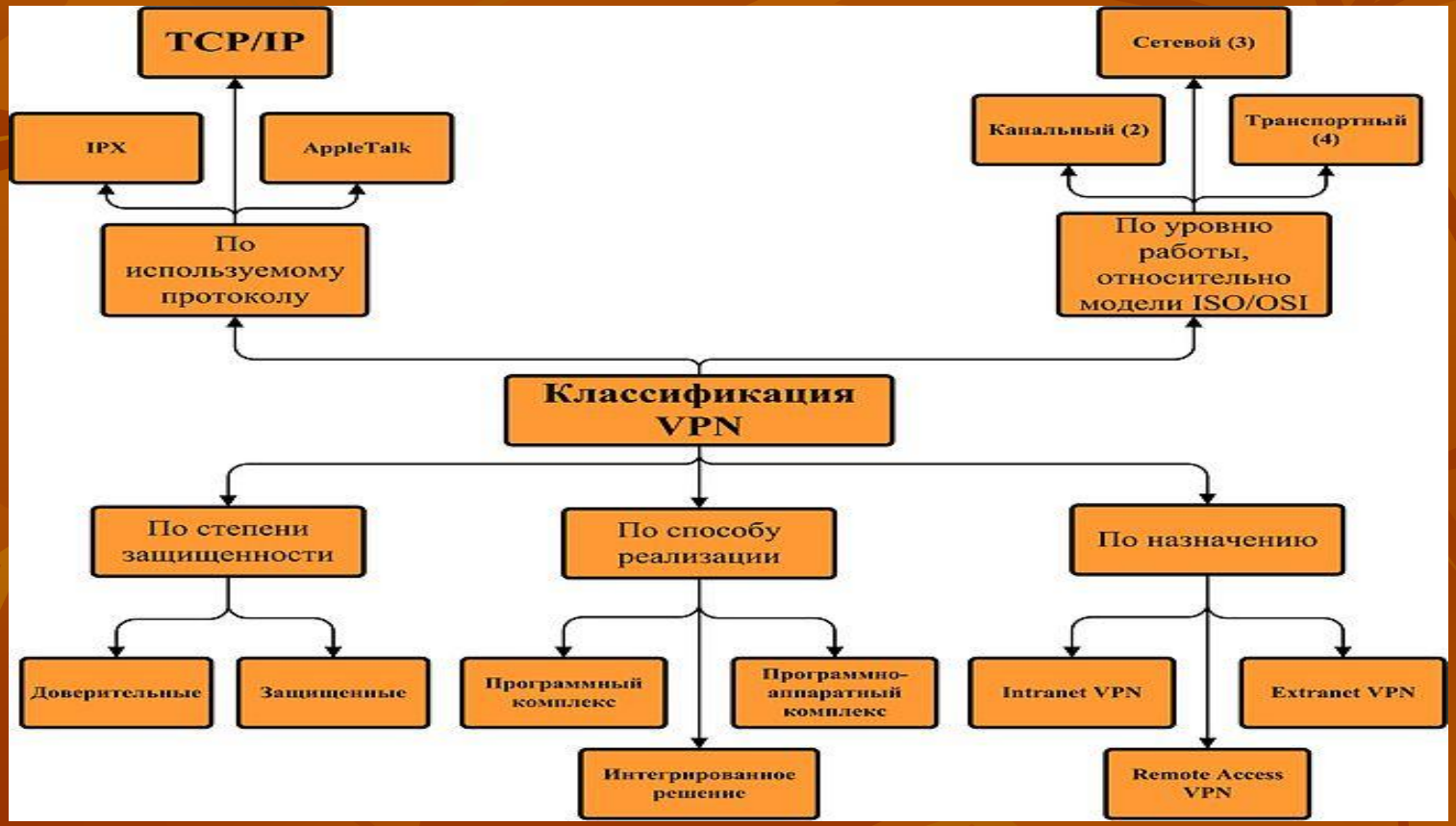
- Обычно VPN развёртывают на уровнях не выше сетевого, так как применение криптографии на этих уровнях позволяет использовать в неизменном виде транспортные протоколы (такие как TCP, UDP).Чаще всего для создания виртуальной сети используется инкапсуляция протокола PPP в какой-нибудь другой протокол — IP (такой способ использует реализация PPTP — Point-to-Point Tunneling Protocol).

Технология VPN

- Технология VPN в последнее время используется не только для создания собственно частных сетей, но и некоторыми провайдерами«последней мили» на постсоветском пространстве для предоставления выхода в Интернет.

Технология VPN

- Классификация VPN



Технология VPN

- Классифицировать VPN решения можно по нескольким основным параметрам:
- **По степени защищенности используемой среды**: Защищённые. Наиболее распространённый вариант виртуальных частных сетей. С его помощью возможно создать надежную и защищенную сеть на основе ненадёжной сети, как правило, Интернета. Примером защищённых VPN являются: IPSec, OpenVPN и PPTP.

Технология VPN

- Доверительные. Используются в случаях, когда передающую среду можно считать надёжной и необходимо решить лишь задачу создания виртуальной подсети в рамках большей сети. Проблемы безопасности становятся неактуальными. Примерами подобных VPN решений являются: Multi-protocol label switching (MPLS) и L2TP (Layer 2 Tunnelling Protocol)

Технология VPN

- **По способу реализации**
- Реализация VPN сети осуществляется при помощи специального комплекса программно-аппаратных средств. Такая реализация обеспечивает высокую производительность и, как правило, высокую степень защищённости.

Технология VPN

- В виде программного решения. Используют персональный компьютер со специальным программным обеспечением, обеспечивающим функциональность VPN.
- Интегрированное решение Функциональность VPN обеспечивает комплекс, решающий задачи фильтрации сетевого трафика, организации сетевого экрана

Технология VPN

- **По назначению**
- Intranet VPN используют для объединения в единую защищённую сеть нескольких распределённых филиалов одной организации, обменивающихся данными по открытым каналам связи.

Технология VPN

- Remote Access VPN. Используют для создания защищённого канала между сегментом корпоративной сети (центральным офисом или филиалом) и одиночным пользователем, который, работая дома, подключается к корпоративным ресурсам с домашнего компьютера, корпоративного ноутбука, смартфона

Технология VPN

- Extranet VPN используют для сетей, к которым подключаются «внешние» пользователи (например, заказчики или клиенты). Уровень доверия к ним намного ниже, чем к сотрудникам компании, поэтому требуется обеспечение специальных «рубежей» защиты, предотвращающих или ограничивающих доступ последних к особо ценной, конфиденциальной информации.

Технология VPN

- Internet VPN используется для предоставления доступа к интернету провайдерами, обычно если по одному физическому каналу подключаются несколько пользователей.
  Протокол PPPoE стал стандартом в ADSL-подключениях.

Технология VPN

- Client/Server VPN обеспечивает защиту передаваемых данных между двумя узлами (не сетями) корпоративной сети. Особенность данного варианта в том, что VPN строится между узлами, находящимися, как правило, в одном сегменте сети. Такая необходимость возникает в тех случаях, когда в одной физической сети необходимо создать несколько логических сетей.

Технология VPN

- **По типу протокола**
- Существуют реализации виртуальных частных сетей под TCP/IP, IPX и AppleTalk. Но на сегодняшний день наблюдается тенденция к всеобщему переходу на протокол TCP/IP, и абсолютное большинство VPN решений поддерживает именно
- IPSec (IP security) — часто используется поверх IPv4.

Технология VPN

- PPTP (point-to-point tunneling protocol) — разрабатывался совместными усилиями нескольких компаний, включая Microsoft.
- PPPoE (PPP (Point-to-Point Protocol) over Ethernet)
- OpenVPN SSL VPN с открытым исходным кодом, поддерживает режимы PPP, bridge, point-to-point, multi-client server

Корпоративные сети Интранет

- Интранет представляет собой технологию управления корпоративными коммуникациями, и в этом ее отличие от Интернет, который является технологией глобальных коммуникаций. В реализации коммуникаций выделяют три уровня: аппаратный, программный и информационный.

Корпоративные сети Интранет

- С точки зрения аппаратного и программного уровней коммуникации - это организация надежного канала соединения и передача информации без искажений, организация хранения информации и эффективный доступ к ней.
- В плане технической реализации этих уровней Интранет практически не отличается от Интернет.

Корпоративные сети Интранет

- **Главная отличительная особенность Интранет кроется в информационном уровне коммуникаций.**
- С информационной точки зрения коммуникации - это поиск и передача знаний.
- Здесь можно выделить минимум три уровня, без которых любое общение либо невозможно, либо бессмысленно:

Корпоративные сети Интранет

- Универсальный язык представления корпоративных знаний
- Модели представления.
- Фактические знания - это конкретные предметные знания

Корпоративные сети Интранет

- Универсальный язык представления корпоративных знаний - это такой язык описания, который не связан с конкретными предметными областями деятельности организации. Его использование: обеспечение унификации представления знаний;обеспечение однозначности толкования знаний всех уровней; сведение процессов обработки информации к простым процедурам.

Корпоративные сети Интранет

- Модели представления. Этот уровень определяет конкретную специфику предметов деятельности компании: понятия и символы предметной области, теоретические представления о предмете и самой организации
- Все три уровня образуют корпоративные знания и являются содержательным контекстом корпоративных коммуникаций.

Корпоративные сети Интранет

- **Интранет и методы управления**
- Точно так же, как не существует универсального лекарства или инструмента, нет и универсальной информационной технологии для поддержки корпоративного управления. У каждой такой технологии есть своя область эффективности Каждая информационная технология поддерживает определенные методы управления.

Корпоративные сети Интранет

- Метод управления определяет то, на что и как надо воздействовать управляющему для достижения ожидаемых результатов бизнеса.
- Можно выделить следующие три большие группы методов управления: а) ресурсами; б) процессами; в) корпоративными знаниями (коммуникациями).

Корпоративные сети Интранет

- Специфику метода управления можно также определить способами представления корпоративных знаний на уровне прикладных моделей и на уровне универсальных языков представления.

Корпоративные сети Интранет

- **Первая группа.** Модель, лежащая в основе этих методов, представляет организацию как систему ресурсов (финансов, материальных запасов, кадров), принадлежащих владельцам - юридическим лицам, структурным подразделениям, физическим лицам. Все процессы описываются как ***проводки***, отражающие перемещение ресурсов между владельцами

Корпоративные сети Интранет

- Метод управления хорошо описывается моделями, ставшими стандартами: модель бухгалтерского учета (например, GAAP), планирование производственных ресурсов (MRP II), планирование всех ресурсов предприятия (ERP). В качестве универсального языка представления используются балансовые модели с языком проводок.

Корпоративные сети Интранет

- **Вторая группа** представляет организацию как систему бизнес-процессов. Здесь центральными понятиями выступают процесс, функция, данные, событие. Основная цель управления для этих методов - обеспечение координации событий и функций. Ко второй группе можно отнести такие методы, как управление качеством (TQM - стандарт ISO9000),

Корпоративные сети Интранет

- управление процессами (Workflow - стандарты ассоциации Workflow Management Coalition)
- К этой же группе можно отнести управление проектами (семейство стандартов PMI), но лишь в той степени, в какой эти проекты можно считать типовыми, сведенными до уровня технологии.

Корпоративные сети Интранет

- Методы управления поддерживаются ПО, которое известно в России как системы управления проектами, документооборотом, технологическими процессами

Корпоративные сети Интранет

- **Третья группа** представляет организацию как систему небольших коллективов сотрудников, решающих общую задачу, а в роли организующих факторов выступают корпоративные знания и эффективные коммуникации. Главным корпоративным ресурсом управления становится база корпоративных знаний.

Корпоративные сети Интранет

- Основная цель управления - обеспечение координации, коммуникации и быстрого поиска информации для самостоятельного принятия решения. Эта группа методов управления получила общее название "управления знаниями" (Knowledge Management).

Корпоративные сети Интранет

- Методы управления поддерживаются ПО класса GroupWare, информационно-поисковыми системами, Интранет-технологиями: Web-технологией, электронной почтой, телеконференциями. Системы GroupWare послужили отправной точкой технологии Интранет Ведущими производителями Интранет-систем стали: Lotus (IBM), Microsoft, Novell.

Корпоративные сети Интранет

- **Интранет в системе корпоративного менеджмента**
- Одной из известных точек зрения на систему менеджмента стала концепция "7С".
- Концепция "7С" выделяет семь базовых элементов управления: **с**тратегия, **с**труктура, **с**истема и процедура работы, **с**тиль, **с**остав персонала, **с**умма навыков и **с**овместно разделяемые ценности.

Корпоративные сети Интранет

- Названия каждого из них начинается с буквы "С" и все они взаимосвязаны.
- Все эти элементы управления могут быть разделены на две принципиально различающиеся группы: «жесткие» : стратегии, оргструктуры, системы и процедуры и «мягкие» : стиль управления, состав персонала, сумма навыков и совместно разделяемые ценности.

Корпоративные сети Интранет

- Технология Интранет произвела революцию в области "мягких" элементов управления.Методы управления ресурсами и процессами, как правило, находят свое применение в управлении "жесткими" элементами, а управление корпоративными знаниями (коммуникациями) - в управлении "мягкими" элементами.

Корпоративные сети Интранет

- Правильное сочетание информационных технологий, методов и элементов управления закладывает основу для построения сбалансированной стратегии компании

Информационные технологии в корпоративных сетях

Раздел 1

Сетевые трехнологии в автоматизации делопроизводства

СЕТЕВЫЕ ТЕХНОЛОГИИ В АВТОМАТИЗАЦИИ ДЕЛОПРОИЗВОДСТВА

- Термины и определения:
- **Документ** – зафиксированная на материальном носителе информация с реквизитами , позволяющими ее идентифицировать.
- **Делопроизводство** – специфическое направление деятельности, которое занимается составлением, оформлением документов, их обработкой и хранением.
- **Документооборот** – движение документов организации с момента их получения или создания до завершения исполнения или отправки

СЕТЕВЫЕ ТЕХНОЛОГИИ В АВТОМАТИЗАЦИИ ДЕЛОПРОИЗВОДСТВА

- С точки зрения автоматизации делопроизводства документ распадается на две части:
- **тело**, вся содержательная работа с которым ведется вне рамок системы автоматизации делопроизводства САД;
- **регистрационную карточку**, содержащую все реквизиты документа, которая и является предметом рассмотрения САД.

СЕТЕВЫЕ ТЕХНОЛОГИИ В АВТОМАТИЗАЦИИ ДЕЛОПРОИЗВОДСТВА

- Вся документация организации делится на 3 группы:
- Входящая,
- Исходящая,
- Внутренняя

  Технологическая цепочка обработки и движения документов включает следующие этапы:
- Прием и первичная обработка документа,
- Предварительное рассмотрение и распределение документов,
- Регистрация, контроль за исполнением,
- информационно – справочная работа,
- использование документов и их отправка

СЕТЕВЫЕ ТЕХНОЛОГИИ В АВТОМАТИЗАЦИИ ДЕЛОПРОИЗВОДСТВА

- САД охватывает следующие виды деятельности:
- Документирование;
- Организация документооборота;
- Систематизация архивного хранения.

СЕТЕВЫЕ ТЕХНОЛОГИИ В АВТОМАТИЗАЦИИ ДЕЛОПРОИЗВОДСТВА

Документирование включает:

подготовку, оформление, согласование, утверждение и выпуск документа ;

К организации документооборота относится : обеспечение движения, хранение, поиск и исполнение документом, контроль исполнения документов и их списание.

Систематизация архивного хранения включает: определение правил хранения, исполнения и уничтожения архивных документов; обеспечение поиска документов в архиве.

СЕТЕВЫЕ ТЕХНОЛОГИИ В АВТОМАТИЗАЦИИ ДЕЛОПРОИЗВОДСТВА

Структура АСД

СЕТЕВЫЕ ТЕХНОЛОГИИ В АВТОМАТИЗАЦИИ ДЕЛОПРОИЗВОДСТВА

- Документ характеризуется списком атрибутов. Атрибуты, общие для всех типов документов:
- Регистрационный № документа,
- Источник документа,
- отв. Исполнитель документа,
- код документа по номенклатуре для организации

СЕТЕВЫЕ ТЕХНОЛОГИИ В АВТОМАТИЗАЦИИ ДЕЛОПРОИЗВОДСТВА

Атрибуты входящих документов:

- контрольный срок исполнения,
- контролирующее лицо

Атрибуты внутренних документов:

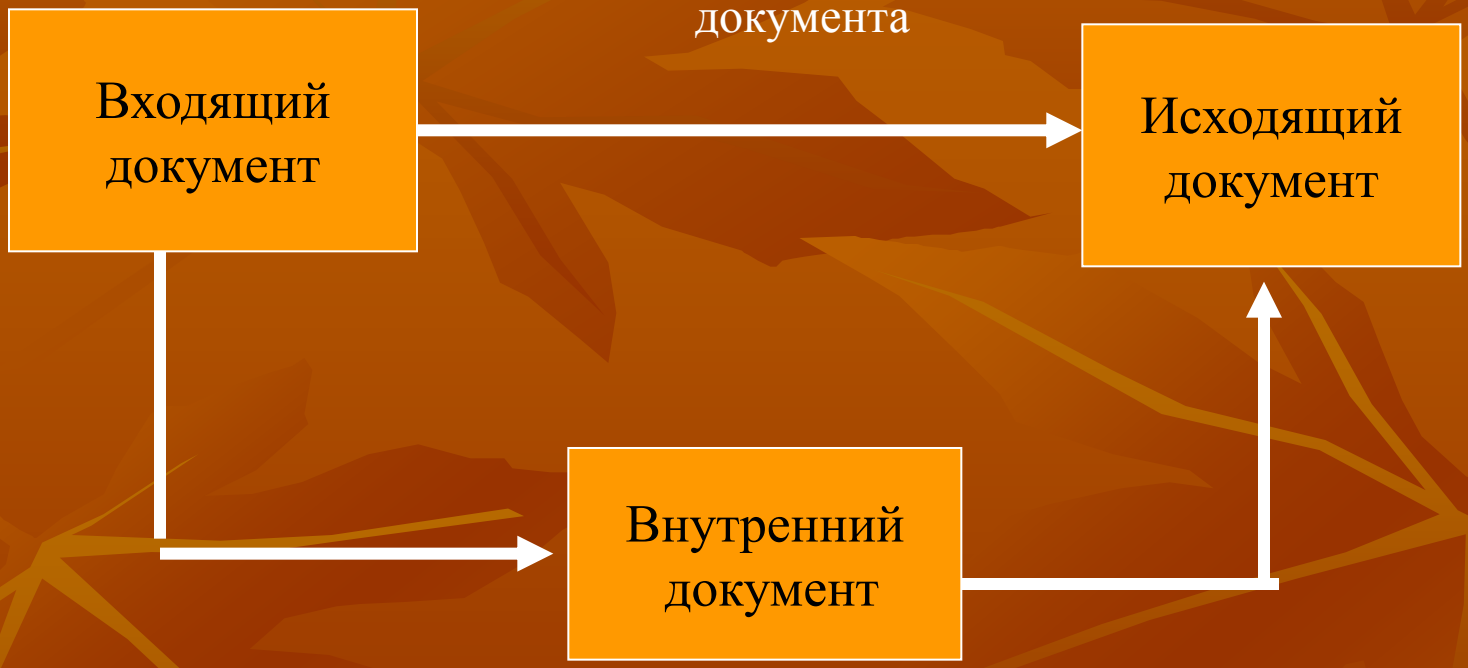
- список подразделений для ознакомления (список рассылки),
- контрольный срок ознакомления или исполнения,
- список исполнителей.

Атрибуты исходящих документов:

- документ – основание,
- список рассылки,
- контрольный срок ответа.

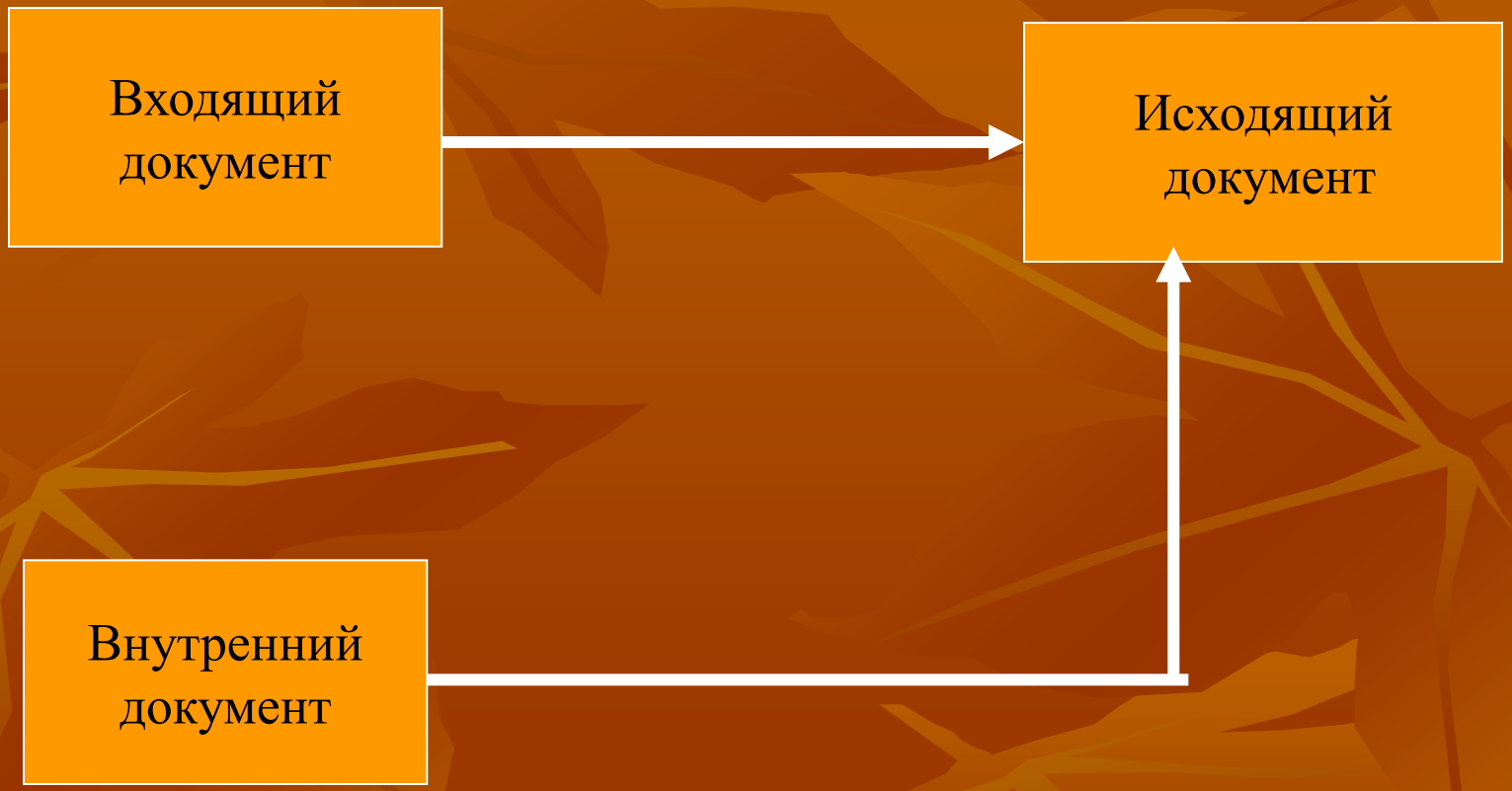
Схемы взаимосвязей документов (1)

- Вариант1



Схемы взаимосвязей документов (2)

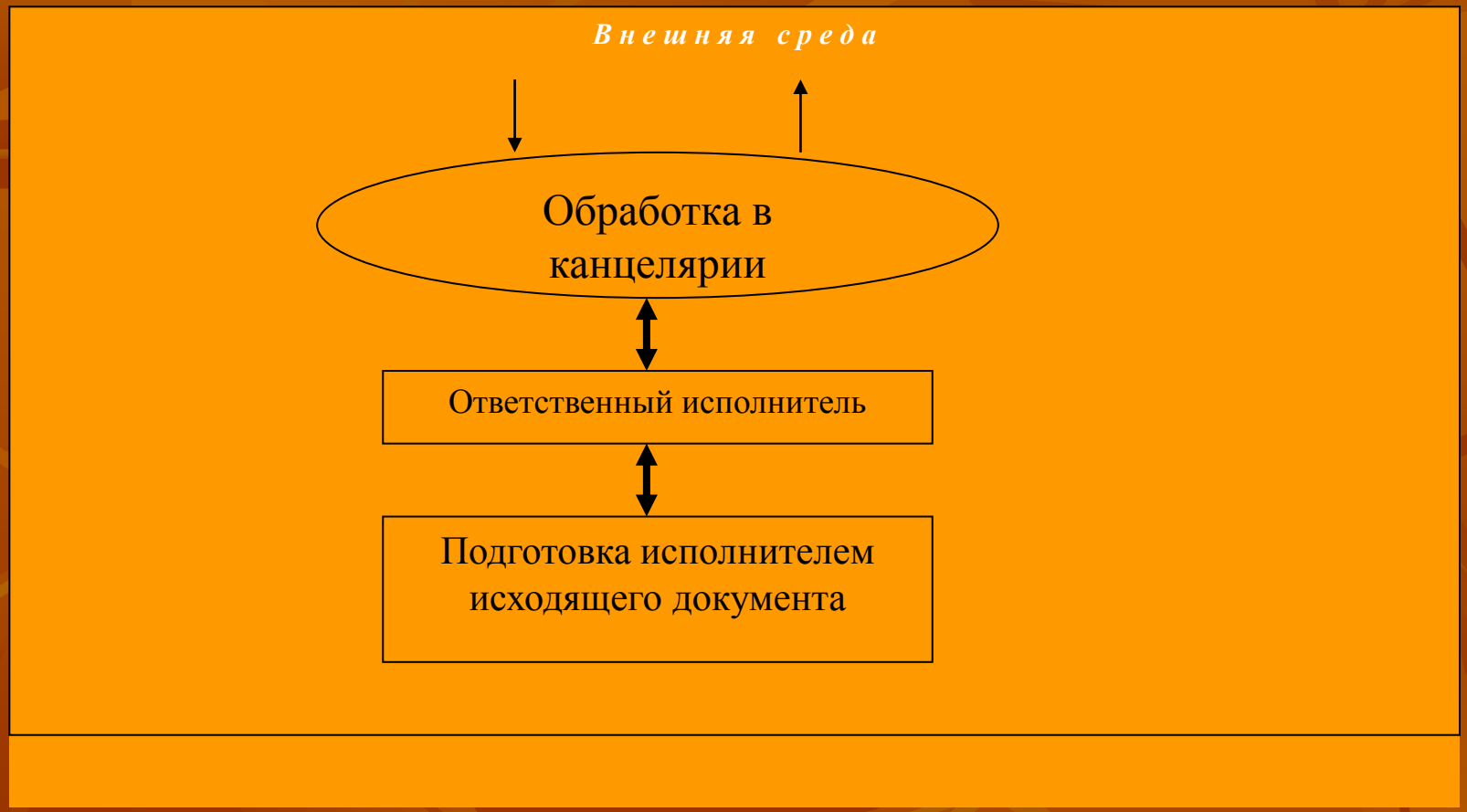
- Генерация исходящего документа



Схемы взаимосвязей документов (3)

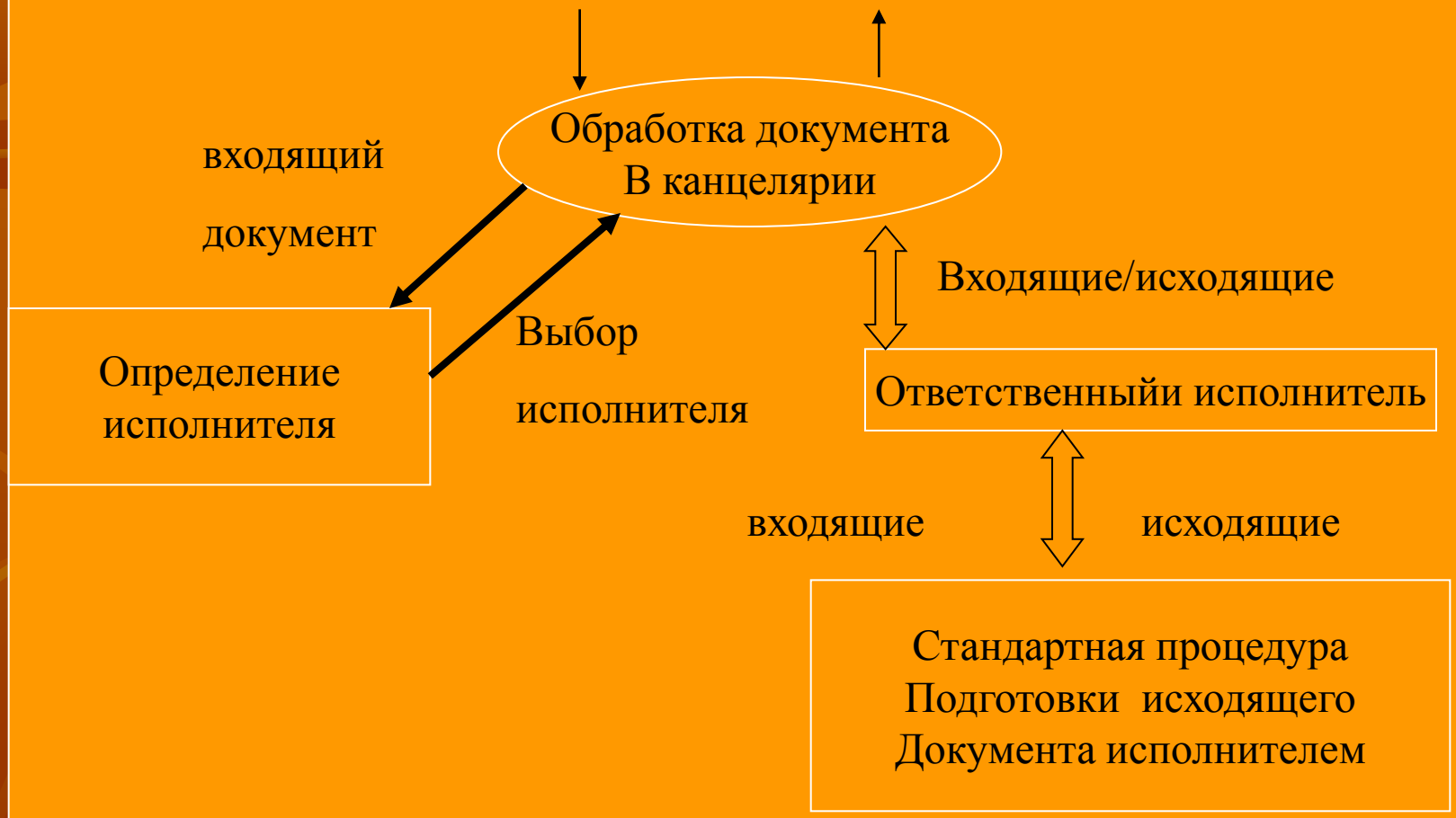
- Исходящий документ на контроле

Типовые процессы обработки документов (1)



Типовые процессы обработки

- *Внешняя среда*



Классификация АСД и технологий групповой деятельности

АСД делятся на два класса:

- системы управления электронными архивами;
- системы управления электронным документооборотом

поддерживаемые технологии групповой работы:

- groupware;
- workflow.

Технология groupware

- ориентирована на небольшие рабочие группы и характеризуется поддержкой выполнения одной коллективной задачи и отсутствием требований структуризации в организации работ. Поддержка ограничивается обеспечением коллективного доступа к информации с помощью различных методов, в том числе сетевого доступа к файлам и БД, локальной и глобальной электронной почты и др.

Технология workflow

- ориентирована на автоматизацию документооборота в средних и крупных организациях.
- Для нее характерна
- поддержка многопользовательской работы с несколькими задачами одновременно;
- четкая структуризация выполнения работ по ролям и документам с контролем исполнения..

Общие требования к АСД (1)

- **Масштабируемость.** АСД должнв поддерживать как 5, так и 500 пользователей и ее способность наращивать свою мощность должна ограничиваться только возможностями аппаратного обеспечения.
- **Распределенность**. Основная проблема работы с документами возникает в территориально – распределенных организациях; поэтому архитектура АСД должна поддерживать взаимодействие распределенных процессов.

Общие требования к АСД (2)

- **Модульность**. Так как заказчику может потребоваться внедрение не всех компонентов АСД, а только некоторой ее части, то система должна состоять из отдельных модулей взаимосвязанных между собой.
- **Открытость**. АСД не может и не должна существовать в отрыве от других систем автоматизации управленческой деятельности. Следовательно она должна иметь открытые интерфейсы для возможной доработки и интеграции с другими системами.

Системы управления электронными архивами

- DOCS OPEN. Краткая характеристика:
- архитектура «клиент - сервер». Включает:
- библиотечный сервер для хранения карточек документов,
- сервер документов, на котором находятся сами документы,
- сервер полнотекстового индекса.

Системы управления электронными архивами

- DOCS OPEN:
- реализована иерархическая схема хранения документов в сетевой файловой системе ;
- поддерживает распределенную обработку документов непосредственно на местах хранения.
- В то же время позволяет редактировать документ с временным копированием на локальный диск.

Системы управления электронными архивами

- DOCS OPEN:
- Поиск документов основан на интерфейсе QBE (Query by Example);
- Для поиска заполняется предполагаемая карточка документа;
- средства защиты документов:поддерживаются права доступа как к карточке документа, так и к его телу.

Системы управления электронными архивами

- DOCS OPEN: интеграция с приложениями
- имеется модуль обмена информацией с Lotus Notes: Interchange for Lotus Notes.
- интеграция с АСД Action WorkFlow.

Системы управления электронными архивами

- Excalibur EPS:
- оригинальная технология адаптивного распознавания образов AFRP основанная на механизме нейронных сетей ;
- имеет русский интерфейс и позволяет работать с русским текстом.
- интегрируется с САД Staffware

Системы управления электронными архивами

- Excalibur EPS:поддерживает режимы поиска:
- Нечеткий поиск по полному содержанию документа;
- Нечеткий поиск по названию документа;
- Ассоциативный поиск по всему тексту с заданными синонимами;
- Прямой доступ к файлам по пиктограммам;
- Запросы в стиле обычной базы данных;
- Поиск по ключевым словам в полном тексте, именах и названиях;
- Логический поиск по всему тексту.

Средства автоматизации документооборота

- *Action WorkFlow. Основные особенности*
- Методология учитывает «человеческий» фактор: в качестве адресов используются не фамилии конкретных сотрудников, а их должности;
- Созданная карта процесса статически проверяет себя на замкнутость;
- Поддерживается четкая система контроля исполнения как по исполнителям, так и срокам исполнения с определением штрафных санкций;
- На рабочих местах исполнители оперируют на своем профессиональном ОЕЯ языке;
- Поддерживается санкционированный доступ пользователей к документам;

Средства автоматизации документооборота

- Система включает 3 части:
- Aws Manager – представляет ядро системы;
- AWS Builder - поддерживает проектирование карт процесса, их реинжиниринг, а также статическую проверку карт на замкнутость процесса;
- AwsAnalist – реализует моделирование деловых процессов, существующих на предприятии.

Средства автоматизации документооборота

- Action WorkFlow имеет модульную структуру и включает электронный архив и систему контроля исполнения;
- информационная модель базы данных, удовлетворяет следующим требованиям:
  - Поддержка иерархического построения с распределениям по разным серверам с ограничением прав доступа;
  - классификация документов по их типам: входящие, внутренние, исходящие;
- Ведение истории жизни документа;
- Объединение документов во временные иерархические группы с различным уровнем доступа без изменения физического размещения документов в архиве.

Средства автоматизации документооборота

- *Lotus Notes/ Domino. Представляет собой платформу с архитектурой «клиент сервер», обеспечивающую разработку и размещение прикладных программ группового обеспечения работ.*

Lotus Notes/ Domino

- Lotus Notes – это база документов. Ее основным элементом является документ. Структура документа определяется его *формой*, содержащей ряд полей,
- Документы в БД Notes могут иметь как структурированный, так и не структурированный формат. Поэтому Notes может хранить и обрабатывать такие массивы данных обработка которых реляционными СУБД сопряжена с большими сложностями

Lotus Notes/ Domino

- Благодаря документной модели Notes предоставляет пользователям управление и распределение такой информации как :
- таблицы,
- отформатированный текст,
- страницы WWW,
- графика,
- связанные или внедренные объекты,
- информацию мультимедиа,
- сканированные изображения и факсы,
- звуковые и видеофрагменты.
- Таким образом Notes выступает в качестве центральной точки доступа ко всей корпоративной информации;

Lotus Notes/ Domino

- Поддержка функции полнотекстового поиска, которая позволяет индексировать документы и проводить их поиск по запросам.
- Управление версиями документов с отслеживанием множества изменений, вносимых в документ различными пользователями
- Поддержка ссылок на документы на основе гипертекста.
- Поддержка технологии репликации, которая позволяет рабочим группам расположенным в различных географических точках синхронно обрабатывать информацию

Полнотекстовый индекс.Определение

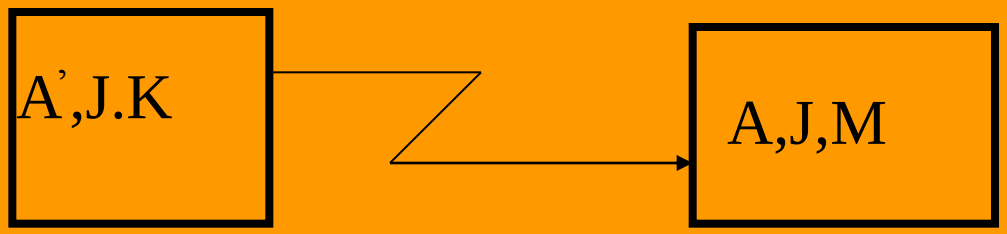
- **Полнотекстовый индекс** (Full text index) - результат полнотекстового индексирования документов в базе данных. Полнотекстовый индекс содержит информацию о том, в каком документе какая лексическая единица содержится. Если в полнотекстовом индексе учитывается словарное окружение лексической единицы, такой индекс называется **контекстно-зависимым.** Контекстно-зависимый индекс обеспечивает исполнение поисковых запросов с контекстными операторами (фраза, следование, близость). Контекстно-зависимые полнотекстовые индексы различают также по основанию деления - единице структурирования текста (слова, слова и предложения, слова и предложения и абзацы).

Lotus Notes/ Domino

- Создание дубликата

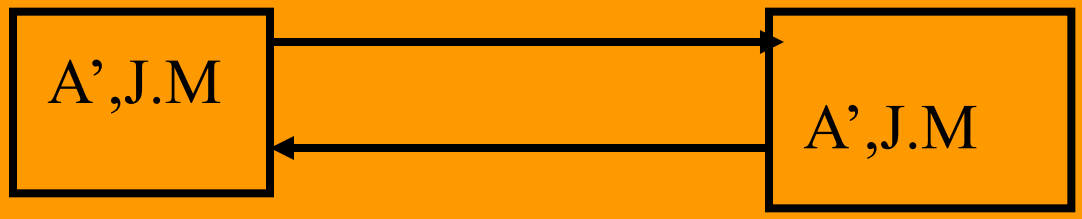
- Подключающийся время от времени пользователь

- Дополнения,
- Изменения,
- Удаления
- На сервере



Дополнения, изменения, удаления на локальном Серваере

- Двунаправленная репликация



Lotus Notes/ Domino

Репликация в LN обладает следующими свойствами:

- Двунаправленностью;
- Эффективностью. Репликация осуществляется на уровне отдельного поля, что оптимизирует ресурсы и обеспечивает самую короткую продолжительность цикла синхронизации БД при выполнении репликации.
- Репликация для клиента.Notes поддерживает не только взаимодействие между серверами, но и между сервером и мобильным клиентом.
- Выборочность репликации.
- Фоновый режим репликации. Такой режим позволяет мобильному пользователю не прерывать решение других задач

Lotus Notes/ Domino

- **Обеспечение безопасности.**
- В Notes используется 4 уровня защиты данных:
- двунаправленная аутентификация;
- управление доступом;
- секретность на уровне поля;
- цифровые подписи.

Lotus Notes/ Domino

- ***Система передачи сообщений*** *используется:*
- *Для связи между людьми,*
- *для совместной работы членов рабочих групп,*
- *в качестве наиболее важного компонента приложения рабочих потоков*
- *в качестве платформы для ведения календаря и планирования*

Lotus Notes/ Domino

- Notes поддерживает единый каталог и адресную книгу Name& Address Book .Адресная книга управляет всей информацией о ресурсах каталога от адресов электронной почты отдельных пользователей до записей о соединениях, определяющих моменты проведения репликаций.

Lotus Notes/ Domino

- ***Среда разработки приложений*** **Notes** представляет собой интегрированную среду, предоставляющую средства разработки пользователям с различным опытом. Платформа содержит встроенные средства программирования.
- **Notes** поддерживает следующие средства программирования:формул, Lotus Script, JavaScript

Информационные технологии в корпоративных сетях

- Раздел 2

- СЕТЕВЫЕ ТЕХНОЛОГИИ ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

СЕТЕВЫЕ ТЕХНОЛОГИИ ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

- ***дидактическая система*** **-** совокупность методов, выделяющихся по некоторым признакам, а также средств и процессов, направленных на обеспечение всеобъемлющего и полноценного образования.
- ***Дидактическая система*** *– система взаимосвязанных элементов, к которым принято относить цели обучения, принципы его организации, содержание, организационные формы, а также методы обучения.*

СЕТЕВЫЕ ТЕХНОЛОГИИ ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

- Содержание дидактической системы

Дидактическая задача и технология обучения

- Составляющие дидактической системы

Категории дидактики

- ***образование***. *Образованием принято называть целенаправленный процесс и конечный результат приобретения способов деятельности, познавательных навыков и научных знаний.*
- ***обучение***. *Обучением называется целенаправленный процесс познания окружающего мира в результате взаимодействия учащегося и учителя при достижении определенных целей образования*
- ***преподавание***. *Это процесс деятельности учителя в рамках обучения.*
- ***учение.*** *Учением принято называть познавательную деятельность учащегося в период обучения.*

Категории дидактики

- ***Принципы обучения:*** принципы доступности, наглядности, прочности и многие другие
- ***Знания*** . Знаниями принято называть научные факты, понятия, схемы, образы, правила, законы, теории, которые нашли свое отражение в сознании и сохранились в памяти учащегося.
- ***умения***. Умениями называют способы применения приобретенных знаний и жизненного опыта на практике. Умения можно сформировать при помощи упражнений.
- ***навыки***, т. е. действия, которые выполняются человеком почти автоматически, поскольку они доведены до абсолютного совершенства. Навыки человек приобретает в результате постоянного повторения.

Модели преподавания

- **Модель, ориентированная на преподавателя.**
- Традиционная модель обучения при помощи уроков-лекций, которую называют ориентированной на преподавателя, чаще всего используется, когда целью обучения является простая передача информации и знаний.

Модели преподавания

- **Модель, ориентированная на учащегося.** Суть модели состоит в том, что каждый учащийся должен не просто получать информацию, а интерпретировать ее для создания новых знаний..При таком подходе студенты учатся методом проб и ошибок и могут одновременно контролировать ход своего обучения. знаний. Только такое обучение можно считать развивающим.

Модели преподавания

Модель преподавания с применением учебных групп.

- Модель, создает среду, в которой новые знания появляются и распространяются как результат коллективной работы учащихся в учебных группах. В рамках учебной группы опыт и существующие знания явным образом включаются в процесс передачи знаний, приводя, таким образом, к появлению новых знаний.

Технологии дистанционного обучения

- Технология дистанционного обучения -- Есть совокупность методов, форм (модели преподавания) и программно-технических средств обучения и администрирования учебных процедур, обеспечивающих проведение учебного процесса на расстоянии.

Классификация технологий(1)

- **Репродуктивные технологии**. технологии распространения учебных материалов чаще всего основаны на модели преподавания, ориентированной на преподавателя. Классической реализацией репродуктивной технологии является ТВ-технология. Современным вариантом реализации репродуктивной технологии является использование сетевых средств для доставки учебно-методических материалов

- ***Интерактивные технологии обучения***. основаны на личностно-ориентированной модели преподавания и ориентированы на приобретения навыков и умений в качестве цели обучения. Одним из наиболее известных вариантов интерактивной технологии дистанционного обучения является кейс-технология. Кейс-технология основана на использовании наборов (кейсов) текстовых, аудиовизуальных и мультимедийных учебно-методических материалов и рассылке их для самостоятельного изучения учащимися при организации регулярных консультаций у преподавателей-тьюторов (очно или заочно).

Классификация технологий(2)

- ***Компьютерные курсы*** доставляются учащимся по сети и позволяют им контролировать ход своего обучения, выполняя требуемые упражнения и задания для самотестирования, общаться с преподавателем по электронной почте.

- ***Технологии совместного обучения.*** ориентированы на цель обучения «Изменение модели мышления» и предоставляют возможность преподавания с применением учебных групп.

- ***Технологии обучения в реальном режиме времени.*** Данный класс технологий обучения является в наибольшей степени дистанционным, т.к. благодаря техническим средствам виртуальная учебная среда максимально похожа на традиционную очную. Это позволяет проводить обучение практически без ограничений по используемым формам и методам

Технологические платформы ДО

- Под технологической платформой ДО будем понимать совокупность программно-технических средств, направленных на предоставление услуг дистанционного обучения, включая администрирование учебных процедур и проведение учебного процесса на расстоянии.

Основные виды технологических платформ:

- ТВ-технология и, соответственно, платформа
- кейс-технология и, соответственно, платформа ,
- сетевые технологии и платформы

Разновидности *Сетевых технологий*

- Computer-Based Training (CBT) – индивидуальное обучение с использованием локальных компьютерных обучающих программ с различной степенью интерактивности между преподавателем и учащимся.
- Web-Based Training (WBT) – индивидуальное и коллективное обучение с использованием локальных и сетевых компьютерных обучающих программ с различной степенью интерактивности

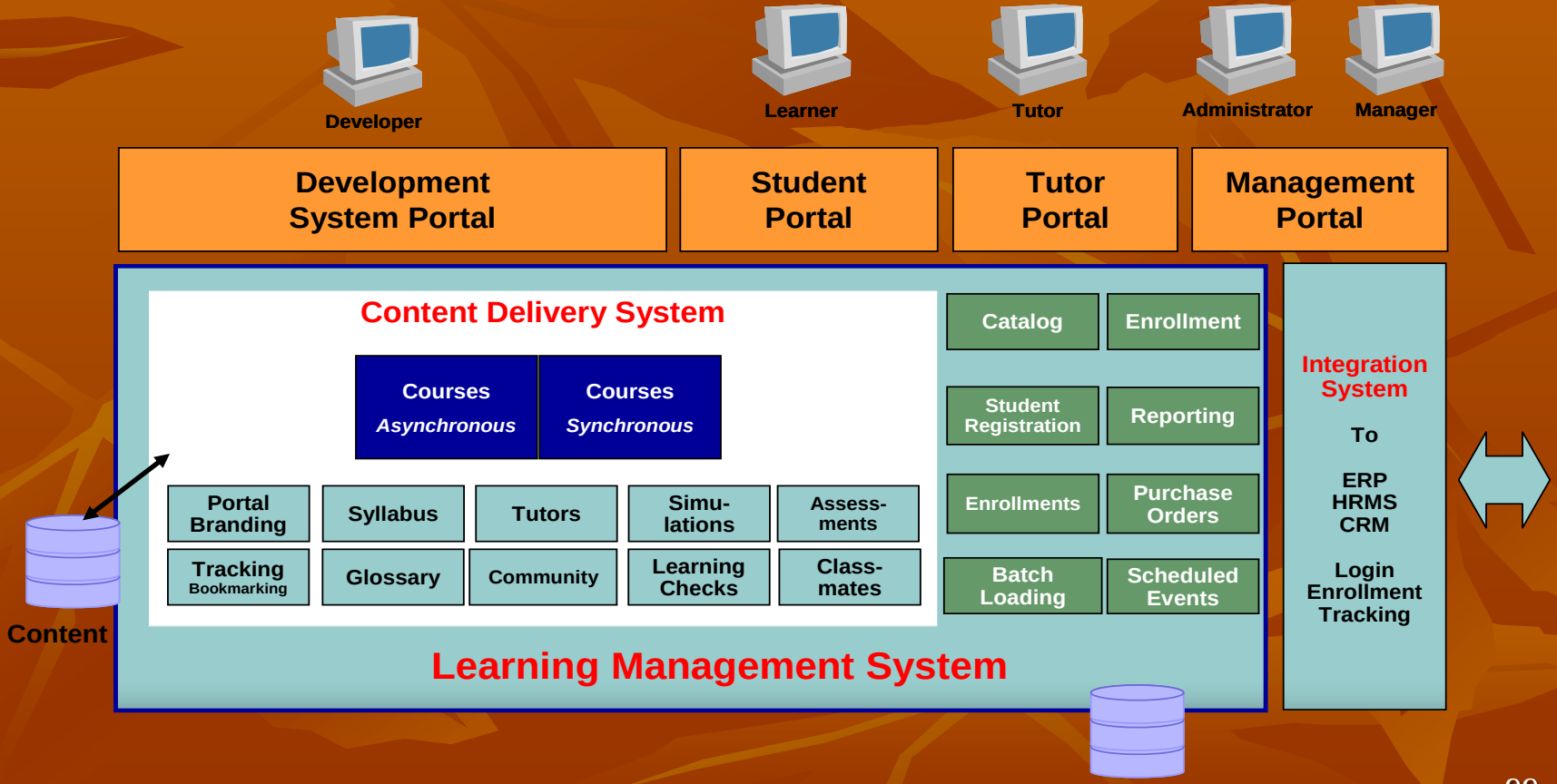
Разновидности *Сетевых технологий*

- **Синхронные** сетевые технологии обучения реализуют истинно дистанционное обучение, когда учащиеся и преподаватели территориально удалены друг от друга. Синхронные технологии предполагают создание виртуальных классов с использованием средств видеоконференц-связи и дополнительных инструментов совместной работы. Эти технологии требуют присутствия всех участников учебного процесса в одно и тоже время.

Архитектура системы электронного обучения

- Здесь приведена обобщенная архитектура

The Modern E-learning System



система электронного обучения IBM Learning Space.

Основные свойства:

- использование распределенной среды обучения;
- сокращение продолжительности циклов разработки учебных программ и курсов;
- поддержание на занятиях высокого уровня интенсивности работы;
- использование лучших моделей и практических примеров индивидуального и группового обучения;
- нацеленность на успех образовательного процесса за счет активных форм обучения и взаимодействия обучаемых с различными схемами общения

система электронного обучения IBM Learning Space

- Упрощенная архитектура системы Learning Space включает следующую совокупность модулей:
- базовый модуль Core;
- вспомогательный модуль Collaboration - работет только совместно с базовым;
- сервер БД;
- почтовый сервер;
- контрольный сервер Traking server$
- Web server – сервер, содержащий учебно – методические материалы

система электронного обучения IBM Learning Space

- Learning Space включает 5 специализированных баз:
- **БД Schedule** – центральный модуль системы, позволяющий участникам просматривать учебные материалы, выполнять упражнения, решать задачи, участвовать в тестировании и проводить исследования. Данная база отражает структуру курса, разработанного преподавателем.
- **БД Media Center** – хранит статьи, новости, главы книг, и пр. Через нее можно получить доступ в сети WWW и к другим внешним источникам информации. БД может хранить информацию, необходимую для проведения различных исследований.

система электронного обучения IBM Learning Space

- **БД Course Room** – интерактивная среда, в которой студенты ведут дскуссии и совместно решают задачи.
- **БД Profiles** – содержит информацию о студентах и преподавателях, их профили и др., а также сведения о ходе обучения.
- **БД Assessment Manager**. Является средством , позволяющим преподавателям оценивать работу каждого студента и сообщать ему результаты. Материалы для контрольных работ, зачетов и экзаменов напра вляются студентам через БД Schedule, а выполненные работы по электронной почте передаются для проверки в БД Assessment Manager

система электронного обучения IBM Learning Space

Обучающая программа «Введение в разработку учебных курсов в LS» обучает составителей учебных курсов и преподавателей созданию эффективных и динамичных программ обучения в режиме on-lin

Она включает презентации и практические занятия по:

- разработке и модификации учебного расписания,
- структурированию баз данных профилей студентов,
- созданию мультимедийной библиотеки задач для студентов и учебных материалов,
- организации электронных дискуссий и руководству ими, а также
- организации проверки студенческих работ

система электронного обучения IBM Learning Space

- В Learning Space поддерживаются следующие механизмы оценки успеваемости:
- Установки статуса учебного материала;
- Начисление баллов за занятие.

Каждый учебный элемент по ходу учебного занятия получает текущее значение параметра «статус»:

- – учащийся не приступил к элементу;
- – элемент провален;
- – элемент частично завершен;
- – элемент полностью завершен;
- элемент пройден успешно

система электронного обучения IBM Learning Space

Поддерживаются следующие механизмы накопления информации о прохождении курса:

- – дата последнего доступа к элементу;
- – дата завершения элемента;
- – количество доступов к элементу;
- – затраченное время на прохождение элемента.

Также поддерживаются механизмы управления прохождением курса:

- – управление выполнением элемента;
- – управление доступом к элементу;
- – управление повторным доступом

система электронного обучения IBM Learning Space

- Learning Space поддерживает следующие виды занятий:
- Generic – общий тип занятий,
- Assessment – компьютерное тестирование,
- Lecture – «живой » урок в виртуальной классной комнате,
- Evaluction – ручное тестирование,
- Reading – хрестоматия,
- Tutorial – электронное учебное пособие,
- Field work – практические занятия,
- Simulation – моделирование (тренинг),
- Laboratory – лабораторные работы,
- Audio – аудиоматериалы,
- Vidio – видеоматериалы.

система электронного обучения IBM Learning Space

- В Learning Space предусмотрено участие в учебном процессе следующих пользователей: студентов,
- инструкторов (тьюторов),
- автров – разработчиков учебных курсов и
- администратора.

С каждым из участников связан определенный профиль – набор прав доступа к ресурсам системы. Одному и тому же участнику может одновременно соответствовать несколько профилей.

Корпоративные сетевые технологии

Раздел 3.

OLTP в сети: проектирование многоуровневых архитектур для транзакционных приложений

Корпоративные сетевые технологии

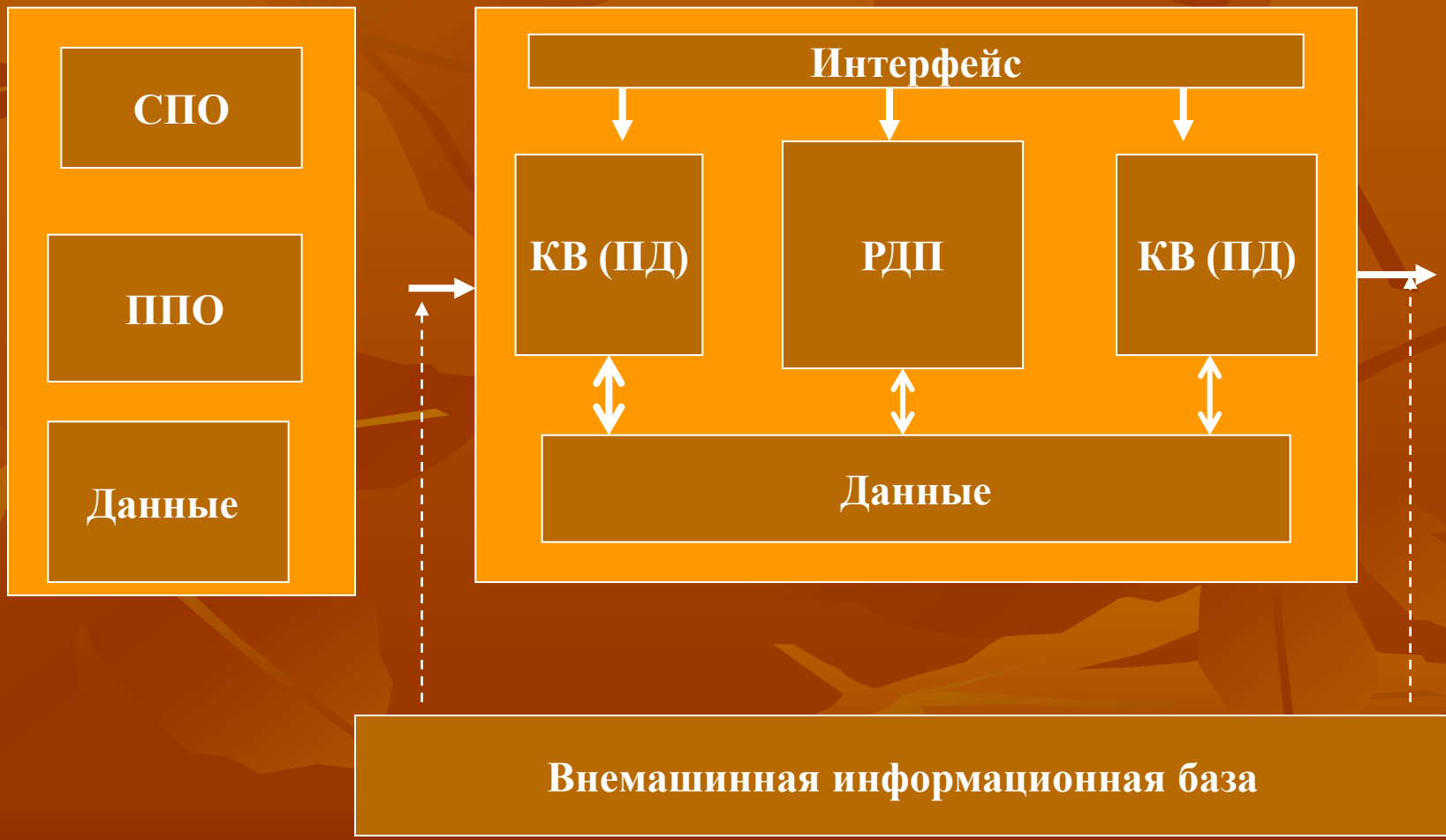
- Широкому использованию Интернет для организации взаимодействия между системами категории B2B и e-коммерции мешали две проблемы:
- Невозможность выполнения полноценных бизнес-транзакций;
- Разнородность интерфейсов RPC и необходимость переформатирования передаваемых данных

Корпоративные сетевые технологии

- Решение первой проблемы связано с развитием WEBOLTP –технологии, интегрирующей удобный доступ к данным через интернет и выполнение бизнес - транзакций в 3-х слойной (звенной) (многозвенной) архитектуре «клиент-сервер» Изложение возможных решений начнем с введения в эту архитектуру.

Архитектура клиент - сервер

- В любой ИС можно выделить 3 слоя:

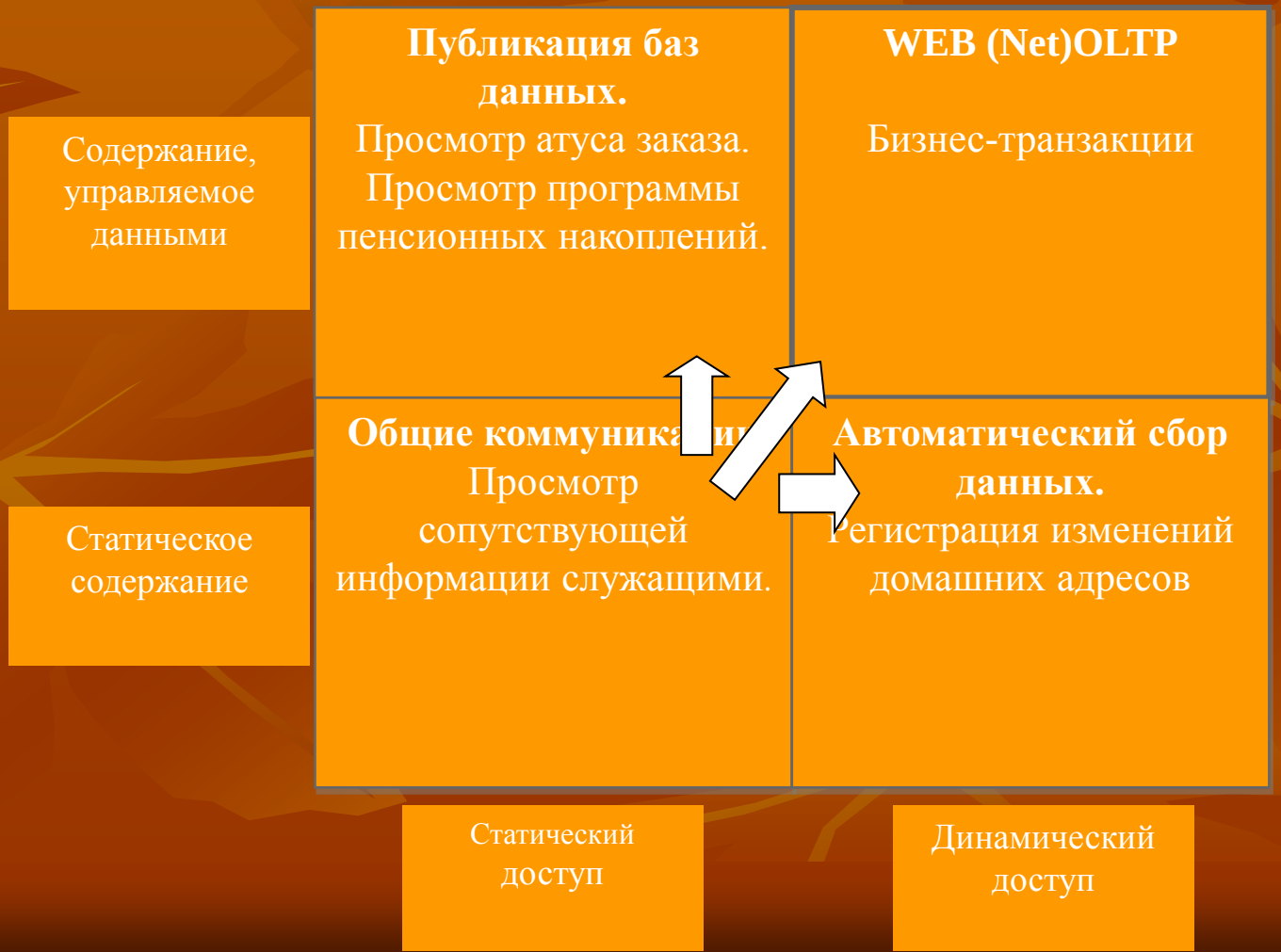


Архитектура клиент - сервер

- Двухуровневая (двухслойная) архитектура

Тенденции и направления развития Интернет

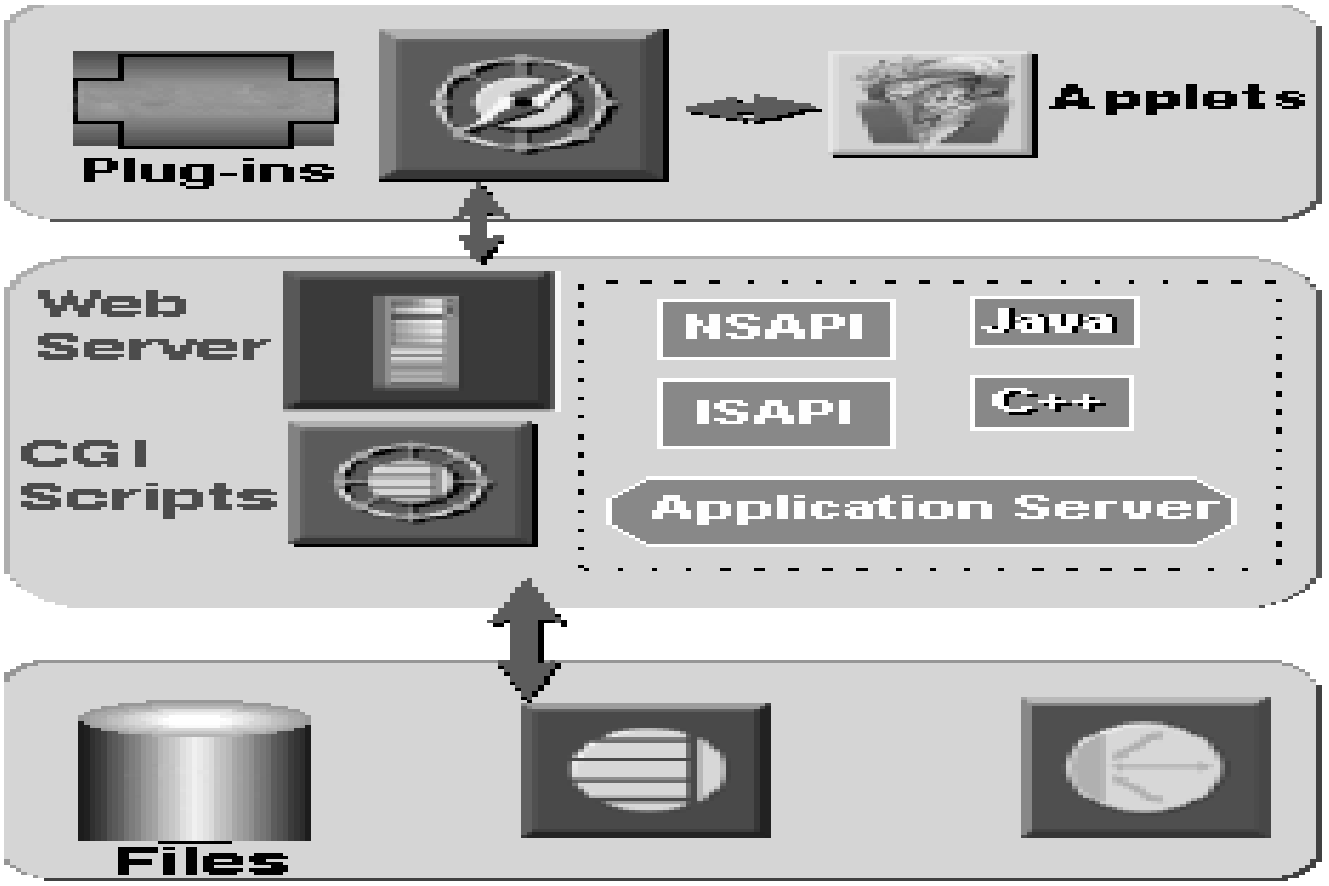
- Рассмотрим рисунок



Базовая архитектура Вэб

- Включает:
- WEB – браузер, служащий для отображения страниц в формате HTML;
- WEB – сервер, который хранит и управляет страницами HTML ;
- Стандартные средства связи между браузером и сервером на базе протокола HTTP.

Расширенная архитектура Вэб



Расширенная архитектура Вэб

- Базовая архитектура была расширена для большей динамичности приложений за счет:
- Простых форм запросов и форматирования данных на основе Java Script для браузера;
- API для Web-сервера, таких как NSAPI и ISAPI, позволяющих браузерам вызывать приложения на стороне сервера;
- Серверов динамической обработки, которые преобразуют данные из БД в страницы в формате HTML.

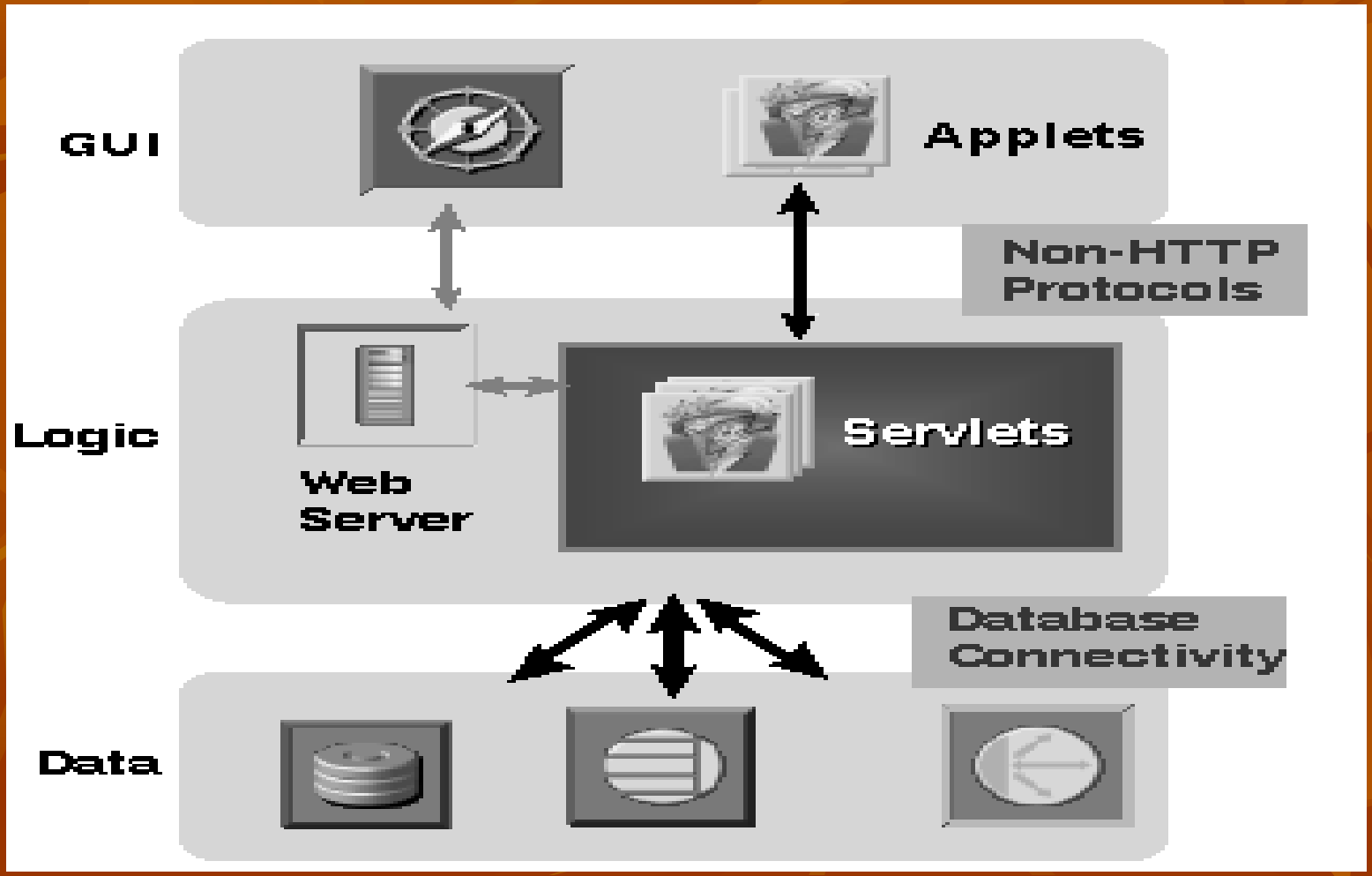
Расширенная архитектура Вэб

- ISAPI - Стандарт **Internet Server API** изначально был создан как Microsoft Information Server API, но в дальнейшем был предложен в качестве открытого стандарта. NSAPI - Стандарт **Netscape Server API** используется для взаимодействия с серверами компании Netscape

OLTP в сети

- WebOLTP — имя, которое было предложено компанией Sybase, для описания приложений, выполняющих транзакции в интернете, интранете или традиционных корпоративных сетях. Отличительные черты WebOLTP при сравнении с OLTP-технологией на мэйнфреймах или в системах клиент/сервер:
- *Тонкие клиенты* ;
- *Большие объемы при большом количестве соединений ;*
- *Непредсказуемые нагрузки* ;
- *Короткий жизненный цикл приложения*

Инфраструктура WebOltp



Инфраструктура WebOltp

- Апплеты являются динамически загружаемыми программами, которые управляют логикой представления данных. В архитектуре WebOLTP апплеты также содержат часть кода, отвечающего за коммуникации, что позволяет им напрямую соединяться с сервлетами, работающими на промежуточном уровне.
- Существует два метода для построения апплетов. Первый основан на использовании Java и JavaBeans, второй — на основе технологии ActiveX

Инфраструктура WebOltp

- **JavaBeans** — классы в языке Java, написанные по определённым правилам. Они используются для объединения нескольких объектов в один (англ. *bean* — фасоль, кофейное зерно) для удобной передачи данных.
- Спецификация Sun Microsystems определяет JavaBeans, как повторно используемые программные компоненты, которыми можно управлять с помощью графического интерфейса

Инфраструктура WebOltp

- ActiveX — это технология активных программных расширений Это новый подход Майкрософт к реализации технологии OLE.
- Технология связывания и внедрения объектов (OLE — Object Linking and Embedding) позволяет использовать документы (которые могут создаваться и редактироваться какой-либо специально разработанной для этой цели программой. Документ этот может быть включен в состав другого документа, создаваемого с помощью иного приложения и тогда он сохраняется в файле нового документа

Инфраструктура WebOltp

Апплеты реализуют следующие преимущества:

- небольшой объем, что обеспечивает быструю загрузку,
- большую интерактивность и более дружественный интерфейс по сравнению с обычными HTML-страницами,
- легкость в разработке и сопровождении

Протоколы, отличные от HTTP

- Ключевые требования для этих новых протоколов включают способность:
- поддерживать информацию о пользователях и транзакциях,
- эффективно формировать результирующую выборку и управлять ею,
- поддерживать транзакции,
- предоставлять надежные механизмы шифрования данных

Сервлеты

- Программы, относящиеся к ПО промежуточного слоя
- Сервлет - это самостоятельный компонент программы, который функционирует в web контейнере, динамически генерируя HTML страницу, XML документ или другой материал в ответ на полученный от клиента запрос. В общем случае, сервлет - это определенным образом построенный Java класс, не имеющий привязки к какой-либо конкретной платформе или web серверу.

Плюсы Сервлетов

- технологию сервлетов можно использовать на любой платформе, где удалось запустить виртуальную Java-машину, на любом web-сервере, имеющем соответствующую поддержку.
- сервлеты можно переносить с одной платформы на другую без какой-либо перекомпиляции.

Плюсы Сервлетов

- сервлетам всегда будет предоставлен доступ к определенному прикладному интрефейсу на любом сервере приложений
- использование объектно-ориентированного языка Java дает свободную расширяемость сервлетов

Cookie

- В большинстве бизнес-приложений необходима идентификация клиентов. Технология сервлетов предусматривает стандартное решение этой задачи: механизм **cookies** и сессии.
- **Cookie -** это некие данные, хранящиеся в браузере клиента. С программной точки зрения, **cookie -** это просто несколько пар «название - значение», которые используются при обращении браузера к определенному серверу.

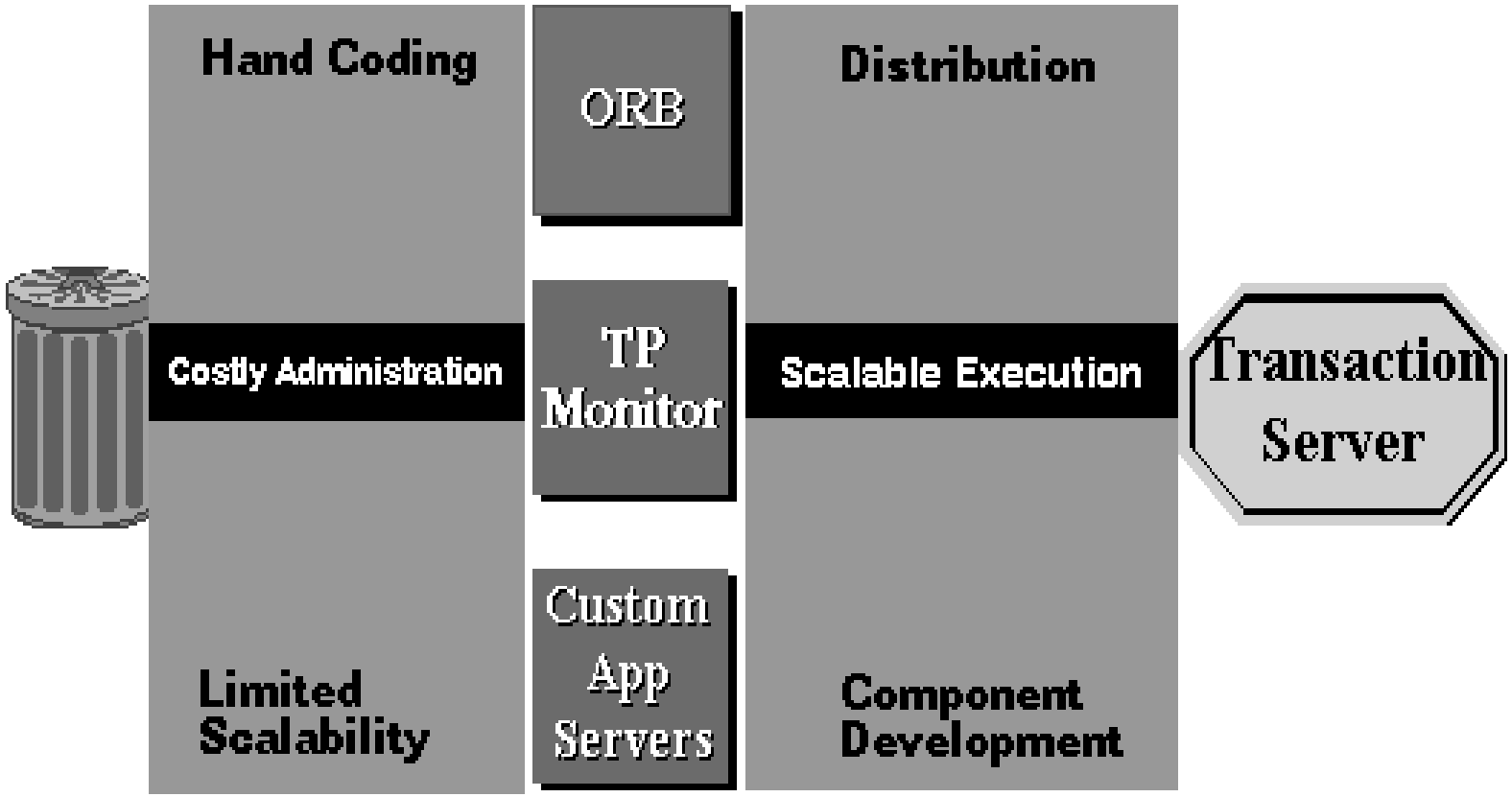
Среда промежуточного уровня

- С появлением многоуровневой архитектуры, на промежуточный уровень (или уровни) была перенесена основная тяжесть прикладной обработки. Этот факт делает промежуточный уровень одним из наиболее критических и проблематичных компонентов WebOLTP архитектуры

Варианты ПО промежуточного уровня

- На сегодняшний день существует следующие наиболее распространенных варианта технологий для построения ПО промежуточного уровня: **CORBA** на основе брокеров объектных запросов (Object Request Brokers — ORBs); **мониторы обработки транзакций** (Transaction Processing Monitors — TP-Monitors) и **серверы Web-приложений**.Каждая из этих технологий имеет свои сильные стороны, но ни одна из них не подходит для требований WebOLTP на промежуточном уровне.
- **серверы транзакций**

Новая архитектура для WebOltp



Новая архитектура для WebOltp

- основные требования к ПО промежуточного уровня:
- масштабируемость и производительность при работе с большим количеством пользователей, сессий, транзакций и соединений с БД,
- высокопроизводительное соединение браузера и back-end хранилища данных,
- поддержка быстрой разработки и развертывания WebOLTP-приложений на промежуточном уровне,
- поддержка как синхронного, так и асинхронного управления транзакциями

Новая архитектура для WebOltp

- Достоинства и недостатки CORBA

- Объекты CORBA имеют превосходные возможности построения многоуровневой архитектуры с вызовом сильно распределенных объектов и прочими сервисами.
- но сложность общего решения и недостаток надежных средств поддержки ограничивает их применение только квалифицированными разработчиками.
- К тому же, большинство ORB имеют примитивные механизмы исполнения на стороне сервера, что также ограничивает эффективность и масштабируемость

Новая архитектура для WebOltp

- TP мониторы имеют устойчивые и отработанные механизмы выполнения, которые предоставляют превосходную эффективность и масштабируемость.
- Однако, подобно объектам ORB, их общая сложность и собственный интерфейс API зачастую делает TP мониторы трудными в использовании и дорогими с точки зрения установки, управления и поддержки

Новая архитектура для WebOltp

- Серверы Web-приложений вообще являются специализированными (заказными) разработками на основе одного из инструментальных средств создания Web-узла Технология сервера Web-приложений появилась в результате попытки трансформировать Netscape и Web-серверы Microsoft в серверы приложений; рычагами к этому послужило последнее поколение соответствующих API — (NSAPI и ISAPI)..

Новая архитектура для WebOltp

- Чтобы преодолеть слабые стороны существующих систем и удовлетворить требованиям WebOLTP на промежуточном уровне, обеспечивающим масштабируемость и простоту использования, был предложен новый класс системного ПО: серверы транзакций.

Серверы Транзакций

- Серверы Транзакции объединяют самые лучшие возможности CORBA и мониторов TP с компонентнной разработкой: это дает возможность быстрому созданию масштабируемых WebOLTP приложений. Первыми доступными серверами транзакций стали Sybase Jaguar CTS (Компонентный сервер транзакций) от Sybase, Inc. и Microsoft Transaction Server (прежде известные как Viper).

Сервер транзакций

- Серверы транзакций характеризуются следующими отличительными свойствами:
- предлагают встроенные возможности управления транзакциями,
- обеспечивают механизм запуска и управления сервлетами (servlets),
- поддерживают вызовы распределенных объектов для обеспечения связи в многоуровневых приложениях,
- поддерживают средства быстрой разработки ПО для промежуточного уровня, включая компонентную разработку

Торгово-информационные системы. Платформа dot.Net

- Торгово информационные системы или системы е --коммерции явились первыми потребителями Интернет.

Торгово-информационные системы

- ТИС –это совокупность программно-технических средств и персонала, предназначенных для представления в сети Интернет информации о товарах и услугах с возможностью осуществления их заказа и оплаты

Торгово-информационные системы

- Задачи, решаемые ТИС:
- Информирование потребителей;
- Формированние положительного имиджа фирмы;
- Снижение издержек;
- Обеспечение канала прямых продаж;
- Обеспечение конкурентных преимуществ.

Торгово-информационные системы

- Классификация ТИС:
- В зависимости от доступности информации:
- строго для внутреннего использования - для отдельных корпоративных клиентов, дилеров, ограниченного числа пользователей;
- Общедоступные;
- Смешанные.

Торгово-информационные системы

- В зависимости от способа представления информации:
- Статические;
- Динамические (интерактивные);
- смешанные

Торгово-информационные системы

- В зависимости от наличия связи с остальным информационным полем компании:
- Автономные;
- интегрированные

Торгово-информационные системы

- В зависимости от целевого сегмента рынка:
- Массовые (розничная торговля);
- Бизнес-ориентированные (оптовые продажи);

Торгово-информационные системы

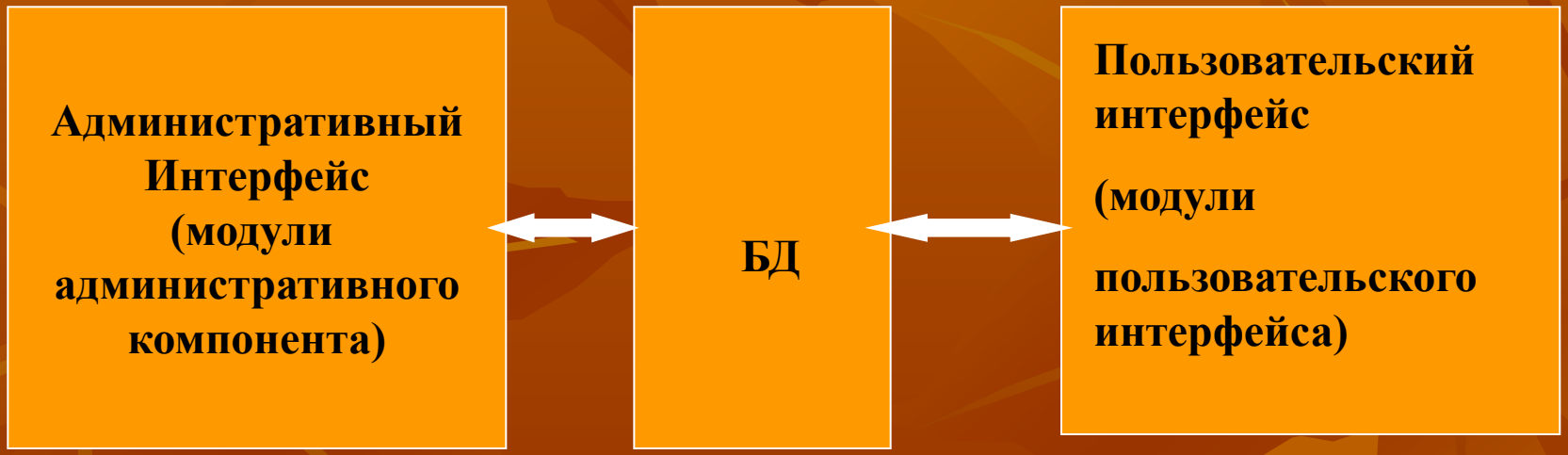
- Основные модули ТИС:
- Интерактивные каталоги;
- Модуль специальных предложений;
- Модуль поиска товара или услуги по заданному критерию, названию, артикулу и пр.;
- Модуль бланка заказа для представления пользователю заказанных товаров;

Торгово-информационные системы

- Модуль расчета стоимости товара и его доставки;
- Модуль аутентификации пользователей;
- модуль постоянных заказов;
- Модуль статистики.

Торгово-информационные системы

- Упрощенно схема ТИС может быть представлена следующим образом:



Платформа dot.Net

- Майкрософт .Net – открытая для языков программирования платформа, основным назначением которой является построение Web—служб.
  - Отличительной особенностью данной платформы является независимость от языков программирования. В рамках платформы предложен собственный язык C#. Наряду с ним платформа открыта для зыков Cobol, Eiffel, Fortran, Perl и др.

Платформа dot.Net

- Цель ее разработки состоит в желании фирмы представить профессиональным разработчикам абстрактную машину, пригодную как для обычных (клиент—серверных n—звенных приложений) так и для Web—ориентированных приложений

Архитектура dot.Net

- Включает 6 уровней:
- 1. Уровень Web—служб– предоставляет пользователяминдивидуальные и корпоративные службы для электронной коммерции и приложений категорий B2B;
- 2 уровень– оболочки и библиотеки ASP.Net, ADO.Net, Windows Forms. Оболочки и библиотеки ASP.Net (активные серверные страницы) обеспечивают разработку интеллектуальных сайтов;

Архитектура dot.Net

- ADO.Net –дополнение на базе языка XML к

ActivX Data Objects для объектно – реляционной обоаботки и доступа к БД.

Последний инструмент данного уровня обеспечивает работу с графикой.

- 3 уровень:
- 3.1 Стандарты обмена SOAP; WSDL;
- 3.2 Общие средства разработки Vstudio.Net

Архитектура dot.Net

- Стандарты обмена на базе XML служат в качестве платформенно независимых средств передачи объектов. SOAP –Simple Object Access Protocol –простой протокол доступа к объектам; WSDL --язык описания Web –служб.
- Vstudio.Net – единая среда разработки служит для создания, компиляции, просмотра и отладки программ, написанных на различных языках сторонними разработчиками

Архитектура dot.Net

- 4 уровень -- компонентная модель. Еще одна модель, претендующая на роль лидера в компонентной разработке. С помощью .Net можно создавать «сборочные» модули, каждый из которых состоит из нескольких классов с четко определенными интерфейсами. Достоинством модели является отсутствие языка описания интерфейсов IDL

Архитектура dot.Net

- 5 уровень представлен объектной моделью и спецификацией единого языка. Объектная модель предлагает еще одну концептуальную основу, на которой зиждется объектно—ориентированная система типов .Net и многое другое.
- Единая языковая среда представляет набор механизмов для исполнения .Net – программ вне завивисимости от языка, на котором она была написана.

Архитектура dot.Net

- 6 уровень представлен единой средой исполнения CLR _Common Language Runtime. Эта среда использует общую систему типов (Common Type System). Библиотека классов .Net Framework содержит огромное число классов, доступных в любом языке программирования, поддерживаемом платформой .Net

Архитектура dot.Net

- Вернемся к уровню 2. ASP.Net – новая разработка активных серверных страниц, которая дает инструментарий для создания интеллектуальных Web—сайтов с широкими возможностями программирования.
- Элементы управления в ASP.Net , ориентированные на Web элементы, реализуют пользовательский интерфейс, аналогичный GUI средам, не рассчитанным на Web и намного превосходящие все, что предлагают HTML.

Архитектура dot.Net

- ASP.Net берет на себя реализацию одного из самых сложных моментов обработки в .Net –поддержание состояния клиента. ASP.Net решает эту задачу не храня клиентской информации на сервере и освобождая разработчиков от выполнения этой трудоемкой задачи вручную, пользуясь громоздкими технологиями.

Архитектура dot.Net

- В паре с оболочкой ADO.Net ASP позволяет настроить часть страницы Web таким образом, чтобы напрямую отображать содержимое таблицы БД.
- ASP.Net обеспечивает решение главной проблему – устраняет различия между традиционной для ИТ разработкой программ и Web разработкой.

Платформа MS.Net

- Последний тезис позволяет утверждать, что платформа осуществляет поддержки нового промышленного стандарта – технологии Web служб.
- Следует также отметить, что особую роль в технологии MS.Net играет XML, например, WSDL -- язык описания Web служб основан на XML.

Роль XML в MS.Net

Практически все компоненты платформы dot.Net в той или иной мере используют XML: для кодирования запросов к Web службам и ответов, возвращаемых клиентам;для моделирования данных, используемых в технологиях ADO.net для доступа к данным; при создании кофигурационных файлов; является общепринятым языком для корпоративных серверов, построенных на платформе dot.Net.

Корпеоративные сетевые технологии

- Раздел 4

- Интерфейсы и XML

Корпоративные сетевые технологии

- Широкому использованию Интернет для организации взаимодействия между системами категории B2B и e-коммерции мешали две проблемы:
- Невозможность выполнения полноценных бизнес-транзакций;
- Разнородность интерфейсов RPC и необходимость переформатирования передаваемых данных

Интерфейсы и XML

- B2B-стиль интегрирует бизнес-процессы компаний, он требует передачи коммерческой информации между различными элементами бизнеса. Но способ ведения бизнеса, идентификация и использование данных существенно отличаются от предприятия к предприятию.
- Попытки использовать для передачи данных между предприятиями страницы HTML натолкнулись на непреодолимые трудности

Интерфейсы и XML

- Для решения этой проблемы был разработан новый стандарт языка разметки, называемый extensible Markup Language, или XML
- Язык XML разработан для облегчения обмена структурированными документами, например, заказами или счетами по Интернету и представляет собой метаязык, предназначенный для представления и манипулировании элементами данных.

Интерфейсы и XML

- Возможны несколько уровней или аспектов определения XML:
- На первом уровне XML является протоколом хранения и управления информацией.
- На следующем уровне – это семейство технологий, с помощью которых можно осуществлять все – от оформления документов до фильтрации данных.
- И, наконец, на самом высоком уровне – это философия обработки информации, которая призвана обеспечить максимальную полезность и гибкость данных путем придания им наиболее чистой и структурированной формы

Интерфейсы и XML

- Технология XML предназначена для транспортировки и хранения структурированных данных взамен существующих файлов баз данных, для обмена информацией между программами, а также для создания на его основе более специализированных языков разметки, (например, XHTML).
- XML-данные хранятся в **текстовом формате**. Это обеспечивает программно- и аппаратно-независимый способ хранения данных.

Интерфейсы и XML

- XML - это метаязык.
- Он не является языком разметки, а определяет набор правил для создания языков разметки.
- Язык XML не имеет предопределенных элементов разметки (тегов), а указывает, как создавать собственные .
- XML позволяет автору определить свои собственные теги и структуру документа.

Интерфейсы и XML

- [Определение:
- Объект данных становится **XML документом** если, в соответствии с определениями спецификации, он является корректным. Корректный XML документ также может стать действительным, если отвечает некоторым дополнительным ограничениям.]

Интерфейсы и XML

- Каждый XML документ имеет логическую и физическую структуру. Физически документ состоит из элементов, называемых сущностями.
- Любая сущность может ссылаться на другие сущности, обеспечивая их включение в данный документ. Документ начинается с "корня" или сущности документа. С логической точки зрения, документ строится из деклараций, элементов, комментариев, ссылок на символ и инструкций обработки. Все они размечаются в документе явным образом. Логические и физические структуры должны иметь корректную вложенность

Интерфейсы и XML

- Определение: Текстовый объект становится **корректным** (well-formed) XML документом, если:
- как единое целое, он соответствует сценарию document.
- отвечает всем ограничениям корректности, представленным в спецификации XML.
- Все разобранные сущности, на которые в данном документе прямо или косвенно делается ссылка, являются корректными (well-formed).

Интерфейсы и XML

- Соответствие сценарию document подразумевает следующее:
- В данном объекте содержится один или несколько элементов.
- [Определение: В объекте имеется в точности один элемент, называемый **корневым** или элементом документа, ни одна из частей которого не попадает в содержимое какого-либо еще элемента.] Для всех остальных элементов действует правило, что если начальный тэг находится в содержимом некого элемента, то и конечный тэг должен находиться среди содержимого того же элемента. Проще говоря, элементы, маркируемые начальными и конечными тэгами, должны быть вложены друг в друга правильным образом.

Интерфейсы и XML

- [Определение: Из вышесказанного следует что в документе для любого некорневого элемента **C** имеется другой элемент **P** из этого же документа, такой что **C** находится в содержимом **P**, но при этом не попадает в содержимое какого-либо третьего элемента, также находящегося в содержимом элемента **P**. В таком случае об элементе **P** говорят как о **родителе** элемента **C**, а элемент **C** называют **непосредственным потомком** элемента **P**.]

Интерфейсы и XML

- **Символы**
- [Определение: Разобранная сущность (parsed entity) содержит **текст** - последовательности символов, образующие разметку и символьные данные.]
- [Определение:**символ** - это элементарная единица текста, описанная в ISO/IEC 10646 [ISO/IEC 10646] (см. также [ISO/IEC 10646-2000]). Допустимы символы табуляции, возврата каретки, конца строки, а также разрешенные символы из наборов Unicode и ISO/IEC 10646.. Перечисленные стандарты могут быть дополнены новыми символами в ходе обновления или при написании для них новых редакций. Соответственно, XML процессоры должны принимать любой символ из диапазона, указанного для Char..]

Интерфейсы и XML

- **Символьные данные и разметка**
- Текст документа образуется сочетанием символьных данных и разметки.
- [Определение: **Разметка** принимает форму начальных тэгов, конечных тэгов, тэгов пустых элементов, ссылок на сущности, ссылок на символы, комментариев, разделителей секций CDATA, объявлений типов документов, инструкций обработки, деклараций XML,деклараций текста и любых пробельных символов, которые располагаются на верхнем уровне сущности документа (то есть, вне элемента document и за пределами иных элементов разметки).]
- [Определение: Текст, который не относится к разметке, формирует **символьные данные** документа (character data).]

Интерфейсы и XML

- Определение: **Комментарий** может размещаться в любом месте документа при условии, что он не попадает в границы какого-либо элемента разметки. Комментаций может также появляться в тех местах декларации типа документа, где это разрешено грамматикой. Комментарии не относятся к символьным данным документа, однако XML процессоры могут (но не обязаны) передавать приложению текст полученных комментариев. Пример комментария:
- <!-- declarations for <head> & <body> -->

Интерфейсы и XML

- Определение: **Инструкции обработки**
- (processing instruction, PI) позволяют размещать в документе инструкции для приложений.
- Инструкция PI начинается с указания адреса (PITarget), используемого для идентификации приложения, которому предназначается эта инструкция
- Для формального декларирования адресата инструкции PI может использоваться механизм нотацийXML

Декларирование нотаций

- [Определение: **Нотация** идентифицирует по имени формат неразобранных сущностей, формат элементов, обеспечивающих атрибут нотации, или же приложение, которому адресуется инструкция обработки.]
- [Определение: **Декларация нотации** дает этой нотации название, используемое при декларировании сущности, списка атрибутов или в спецификациях атрибутов, а также внешний идентификатор этой нотации, который может позволить XML процессору или его клиентскому приложению найти вспомогательную программу, способную обработать данные, представленные в этой нотации.]
- **Декларации нотации**
- [82] **NotationDecl ::= '<!NOTATION' S Name S (ExternalID | PublicID) S? '>'**[VC: Уникальность имени нотации]
  [83] **PublicID ::= 'PUBLIC' S PubidLiteral**

Декларирование нотаций

- **Ограничение действительности: Уникальность имени нотации**
- Любое Name может быть использовано только в одной декларации.
- XML процессор должен передать приложению название и внешний идентификатор(ы) всех нотаций, которые были декларированы и на которые имеется ссылка в значениях атрибутов, определениях атрибутов, либо декларациях сущностей. Кроме того, процессор может преобразовывать внешний идентификатор в системный идентификатор, имя файла или иную информацию, необходимую приложению чтобы вызвать процессор для обработки данных в описываемой нотации. (Впрочем, ситуация, когда XML документ декларирует и ссылается на нотацию, для которой не имеется соответствующего процессора обработки в системе, где работают XML процессор или приложение ошибочной не будет.)

Интерфейсы и XML

- Определение: **Секция CDATA** может находиться повсюду, где могут размещаться символьные данные. Использование секции CDATA позволяет избежать обработки блока текста, содержащего символы, которые в других случаях распознавались бы как разметка. Секция CDATA начинается со строки "<![CDATA[" и заканчивается строкой "]]>":]

Интерфейсы и XML

- В секции CDATA распознается только один элемент разметки - строка CDEnd. Поэтому все символы левой угловой скобки и амперсанта могут предстать здесь в своем обычном текстовом виде. Эти символы не нужно (да и невозможно) маскировать с помощью комбинаций "<" и "&". Секции CDATA не могут быть вложенными.
- Пример секции CDATA, в которой строки "<greeting>" и "</greeting>" будут распознаваться не как разметка, а как обычные символьные данные:
- <![CDATA[<greeting>Hello, world!</greeting>]]>
-

Интерфейсы и XML

- Определение: Документ XML должен начинаться с **декларации XML**, указывающей версию используемого языка XML
- Задачей разметки XML документа должно быть описание схемы его размещения и логической структуры, а также связывание пар атрибут-значение с их логической структурой.
- XML предоставляет механизм для определения логических ограничений для логической структуры и формирования предопределенных единиц размещения -декларацию типа документа.
- [Определение: XML документ является **действительным**, если с ним связана декларация типа документа и если этот документ отвечает представленным в ней ограничениям.]
- Декларация типа должна располагаться в документе до первого элемента.

XML:особенности синтаксиса

- Все XML элементы наряду с открывающим должны иметь закрывающий тег:
- Параграф
- Следующий параграф
- В тегах нужно учитывать регистр:
- XML-теги чувствительны к регистру. Так <Letter> отличается от тега <letter>.
- Открывающий и закрывающий теги должно быть написаны с тем же регистром:
- <Message>Неверно</message>
- <message>Верно</message>

XML:особенности синтаксиса

- XML элементы должны быть правильно вложенными друг в друга:
- This text is bold and italic
- Значения атрибута должно быть заключено в кавычки:
- <note date="12/11/2007">
-

XML:особенности синтаксиса

- для добавления комментариев в XML-документ используются символы < -- к —>;
- префиксы «XML» и «xml» зарезервированы только для тегов XML .
- В XML определены два метода записи специальных символов: ссылка на сущность и ссылка по номеру символа.
- Ссылка на сущность (англ. entity references) указывается в том месте, где должна быть сущность и состоит из амперсанда (&), имени сущности и точки с запятой (;).

Особенности синтаксиса XML

- В заголовке документа помещается объявление XML, в котором указывается язык разметки документа, номер его версии и дополнительная информация;
- Все значения атрибутов, используемых в определении тэгов, должны быть заключены в кавычки;
- Вложенность тэгов в XML строго контролируется, поэтому необходимо следить за порядком следования открывающих и закрывающих тэгов;

Особенности синтаксиса XML

- В XML есть несколько предопределённых сущностей, но возможно также определять собственные.
- Полный список предопределённых сущностей состоит из &amp (&), &lt (<), &gt (>), &apos (') и &quot (") — последние две полезны для записи разделителей внутри значений атрибутов. Определить свои сущности можно в DTD-документе.

Особенности синтаксиса XML

- Вся информация, располагающаяся между начальным и конечными тэгами, рассматривается в XML как данные, и поэтому учитываются все символы форматирования (т.е. пробелы, переводы строк, табуляции не игнорируются, как в HTML).
- Если XML-документ не нарушает приведенные правила, то он называется ***формально-правильным*** и все анализаторы, предназначенные для разбора XML- документов, смогут работать с ним корректно

Особенности синтаксиса XML

- Угловые скобки (< >) и заключенные в них имена называются тегами (tags). Теги разграничивают и помечают части документа, а также добавляют другую информацию, которая помогает определить структуру. Текст между тегами является содержимым документа, необработанной информацией, которая может быть телом сообщения, заголовком или полем данных. Разметка и содержимое дополняют друг друга, создавая информационный объект, в котором поделенные на части и помеченные данные заключены в удобный пакет.

Средства проверки правильности XML документа

- Кроме проверки на формальное соответствие грамматике языка, в документе могут присутствовать средства контроля над содержанием документа,
- Существует два способа контроля правильности XML-документа: **DTD - определения(Document Type Definition)** и **схемы данных(Semantic Schema)**.
- Они содержат правила, определяющие необходимые соотношений между элементами и формирующие **структуру документа**.

Проверяющие и непроверяющие процессоры

- XML процессоры, отвечающие требованиям спецификации, делятся на два класса: проверяющие и непроверяющие.
- И проверяющие, и непроверяющие процессоры должны докладывать о нарушениях правил корректности данной спецификации, выявленных в содержимом сущности документа и содержимом других читаемых разобранных сущностей.

Проверяющие процессоры

- [Определение: Проверяющие процессоры должны сообщить (по выбору пользователя) о нарушении ограничений, сформулированных в декларациях DTD, а также невозможности соответствовать критериям действительности, представленным в данной спецификации.] Чтобы выполнить это требование, проверяющий XML процессор должен прочесть и обработать весь DTD и все внешние разобранные сущности, на которые в данном документе делается ссылка.

Непроверяющий процессор

- Для проверки корректности от непроверяющего процессора требется проанализировать лишь сущность документа, включая полный внутренний набор DTD. [Определение: Хотя непроверяющий процессор и не обязан проверять действительность документа, он должен **обработать** все декларации, найденные во внутреннем наборе DTD, а также во всех прочитанных им сущностях параметров, но только до первой ссылки на сущность параметра, которую он уже *не* должен читать. Иными словами, он должен использовать сведения из этих деклараций для *нормализации* значений атрибутов, *подстановки* текста замены для внутренних сущностей и предоставления *значений по умолчанию для атрибутов*.] За исключением случая **standalone="yes"**, процессорам запрещается обрабатыватьдекларации сущностей и списков атрибутов, расположенные после ссылки на сущность параметра, последняя не читается, поскольку может содержать переопределяющие декларации.
-

Определение типа документа (DTD) и XML-схемы

- Решения B2B-стиля требуют высокой степени бизнес-интеграции компаний. Компании, использующие транзакции B2B-стиля, должны иметь способ понимать и проверять любые другие теги. Один из методов выполнения этой задачи состоит в использовании определения типа документа (Document Type Definition, DTD).

Определение типа документа (DTD)

- DTD представляет собой файл с расширением dtd, который описывает XML-элементы, DTD-файл обеспечивает построение логической модели базы данных и определяет синтаксические правила или теги проверки для каждого типа XM L-документов.
- Компании класса B2B и e-коммерции должны разработать DTD и предоставить его в совместное использование

Определение типа документа (DTD)

- В XML-документах DTD определяет набор действительных элементов, идентифицирует элементы, которые могут находиться в других элементах, и определяет действительные атрибуты для каждого из них .
- В DTD для XML используются следующие типы правил: правила для элементов и их атрибутов, описания категорий(макроопределений), описание форматов бинарных данных

Определение типа документа (DTD)

- Для того, чтобы использовать DTD в документе, можно или описать его во внешнем файле и при описании DTD просто указать ссылку на этот файл или же непосредственно внутри самого документа выделить область, в которой определить нужные правила.

Определение типа документа (DTD)

- В первом случае в документе указывается имя файла, содержащего DTD-описания:
- **<?xml version="1.0" standalone=“no" ?>**
- **<! DOCTYPE team SYSTEM "team.dtd">**
- **...**
- Внутри же документа DTD- декларации включаются следующим образом:
- **<?xml version="1.0" ?>**
- **<! DOCTYPE team [**
- **<! ELEMENT team (coach, player, assistant)>**
- **...**
- **]>**

Определение типа документа (DTD)

- В том случае, если используются одновременно внутренние и внешние описания, то программой-анализатором будут сначала рассматриваться внутренние, т.е. их приоритет выше.
- При проверке документа XML-процессор в первую очередь ищет DTD внутри документа.
- Если правила внутри документа не определены и не задан атрибут ***standalone ="yes"*** , то программа загрузит указанный внешний файл и правила, находящиеся в нем, будут считаны оттуда.
- Если же атрибут ***standalone*** имеет значение *"yes*", то использование внешних DTD описаний будет запрещено

Определение типа документа (DTD)

Определение элемента

- Элемент в DTD определяется с помощью дескриптора **!ELEMENT**, в котором указывается название элемента и структура его содержимого. Например, для элемента **<coach>** можно определить следующее правило:
- **<!ELEMENT coach (#PCDATA)>**

PCDATA - parseable character data - любая информация, с которой может работать программа-анализатор) В тексте как правило символ P опускается. В данной секции размещается «чистый» текст.

Определение типа документа (DTD)

- Существует еще две инструкции, определяющие тип содержимого:

EMPTY,ANY.

- **EMPTY** указывает на то, что элемент должен быть пустым(например, <red/>),
- **ANY** - на то, что содержимое элемента специально не описывается.

Определение типа документа (DTD)

- Для текущего элемента может быть задано несколько дочерних объектов в виде списка разделенных запятыми названий элементов.
- Для того, чтобы указать количество повторений включений этих элементов могут использоваться символы или индикаторы повторяемости +,*, ? :

<!ELEMENT team(coach+, player*, assistant?)>

индикаторы повторяемости

Индикатор	Смысл
?	Содержимое встречается не более одного раза
*	Содержимое встречается любое количество раз
+	Содержимое встречается не менее 1 раза
[ничего не указывается]	Содержимое встечается ровно 1 раз

Пример DTD для XML документа

- <?xml version="1.0"?>
- <! DOCTYPE team [
- <!ELEMENT team (title,coach+, player*, assistant?)>
- <!ELEMENT coach (name|PCDATA)>
- <!ELEMENT name (#CDATA)>
- <!ELEMENT player (name, nationality)>
- <!ELEMENT nationality (#CDATA)>
- <!ELEMENT l_name (#CDATA)>
- <!ELEMENT assistant (#CDATA)>]>

Задание списков атрибутов элемента

- Списки атрибутов элемента определяются с помощью ключевого слова !**ATTLIST**. Внутри него задаются названия атрибутов, типы их значений и дополнительные параметры. Например, для элемента <player> могут быть определены следующие атрибуты:
- **<!ATTLIST playernumber ID #REQUIRED**

  type (goalkeeper | back | halfback | forward) #IMPLIED

Типы значений атрибутов (1)

- CDATA - содержимым документа могут быть любые символьные данные
- ID - определяет уникальный идентификатор элемента в документе
- IDREF(IDREFS) - указывает, что значением атрибута должно выступать название (или несколько таких названий, разделенных пробелами во втором случае) уникального идентификатора, определенного в этом документе элемента

Типы значений атрибутов (2)

- ENTITY(ENTITIES - значение атрибута должно быть названием(или списком названий, если используется ENTITIES) компонента (макроопределения), определенного в документе
- NMTOKEN (NMTOKENS) - содержимым элемента может быть только одно отдельное слово(т.е. этот параметр является ограниченным вариантом CDATA)
- Список допустимых значений - определяется список значений, которые может иметь данный атрибут

Типы значений атрибутов (3)

- Так же в определении атрибута можно использовать следующие параметры:
- #REQUIRED - определяет обязательный атрибут, который должен быть задан во всех элементах данного типа
- #IMPLIED - атрибут не является обязательным
- #FIXED "значение" - указывает, что атрибут должен иметь только указанное значение, однако само определение атрибута не является обязательным, но в процессе разбора его значение в любом случае будет передано программе-анализатору
- Значение - задает значение атрибута по умолчанию

Определение компонентов (макроопределений)

- Компонент (entity) представляет собой определения, содержимое которых может быть повторно использовано в документе. В других языках программирования подобные элементы называются макроопределениями. Создаются DTD-компоненты при помощи инструкции !ENTITY:
- **<!ENTITY hello ' Мы рады приветствовать Вас!' >**
- Программа-анализатор, просматривая в первую очередь содержимое области DTD- определений, обработает эту инструкцию и при дальнейшем разборе документа будет использовать содержимое DTD-компонента в том месте, где будет встречаться его название. Теперь в документе можно использовать выражение &hello; , которое будет заменено на строчку *"Мы рады приветствовать Вас"*

Определение компонентов (1)

- Внутри DTD можно задать три типа макроопределений:
- **внутренние макроопределения** - предназначены для определения строковой константы, с их помощью можно организовывать ссылки на часто изменяемую информацию, делая документ более читабельным. Внутренние компоненты включаются в документ при помощи амперсанта &

Определение компонентов (2)

- В XML существует пять предустановленных внутренних символьных констант:
- < - символ "<"
- > - символ ">"
- & - символ "&"
- ' - символ апострофа "'"
- " - символ двойной кавычки """

Определение компонентов (3)

- **Внешние макроопределения** - указывают на содержимое внешнего файла, причем этим содержимым могут быть как текстовые, так и двоичные данные. В первом случае в месте использования макроса будут вставлены текстовые строки, во втором - бинарные данные, которые анализатором не рассматриваются и используются внешними программами
- <!ENTITY logotype SYSTEM "/image.gif" NDATA GIF87A>

Определение компонентов (4)

- **Макроопределения правил -** макроопределения параметров могут использоваться только внутри области DTD и обозначаются специальным символом %, вставляемым перед названием макроса. При этом содержимое компонента будет помещено непосредственно в текст DTD-правила
- Рассмотрим пример:

Определение компонентов (5)

- для следующего фрагмента документа:
- **<!ELEMENT title (PCDATA)>**
  <!ELEMENT name (PCDATA)>
  <!ELEMENT nationality (PCDATA)>
  <!ELEMENT coach (PCDATA | name)>
  <!ELEMENT player ((PCDATA | name), nationality)>
- **<!ELEMENT team (title,coach, player*)>**

Определение компонентов (6)

- можно использовать более короткую форму записи:
- **<!ELEMENT name (PCDATA)>**
- **<! ENTITY %names 'PCDATA | name'>**
  <!ELEMENT coach (%names;)>
  <!ELEMENT player (%names, nationality)>
- **<!ENTITY %content 'coach | (player*)'>**
- **<!ELEMENT team (title,%content;)>**

Типизация данных

- Довольно часто при создании XML-элемента разработчику требуется определить, данные какого типа могут использоваться в качестве его содержимого ;
- Используя типизацию данных, можно создавать элементы, значения которых могут использоваться, например, в качестве параметров SQL-запросов.
- Если в качестве программы на стороне клиента используется верифицирующий XML-процессор, то информацию о типе можно передавать при помощи специально созданного для этого атрибута элемента, имеющего соответствующее DTD-определение.

DTD-определение типа данных

- **Пример**: чтобы указать, что содержимое элемента должно быть длинным целым, можно использовать следующее DTD- определение:
- **<!ELEMENT counter (PCDATA)>**
- **<!ATTLIST counter data_long CDATA #FIXED "LONG">**
- Задав атрибуту значение по умолчанию LONG и определив его как FIXED, программа-клиент получит необходимую информацию о типе содержимого данного элемента, и сможет самостоятельно определить соответствие типа этого содержимого указанному в DTD-определении

DTD-определение типа данных

- <!ELEMENT price (PCDATA)>
- <!ATTLIST price data_currency CDATA #FIXED "CURRENCY">
- <!ELEMENT rooms_num (PCDATA)>
- <!ATTLIST rooms_num data_byte CDATA #FIXED "BYTE">
- <!ELEMENT floor (PCDATA)>
- <!ATTLIST floor data_byte CDATA #FIXED "INTEGER">
- <!ELEMENT living_space (PCDATA)>
- <!ATTLIST living_space data_float CDATA #FIXED "FLOAT">
- <!ELEMENT counter (PCDATA)>
- <!ATTLIST counter data_long CDATA #FIXED "LONG">
- <!ELEMENT is_tel (PCDATA)>
- <!ATTLIST is_tel data_bool CDATA #FIXED "BOOL">
- <!ELEMENT house (rooms_num, floor,living_space, is_tel, counter, price)>
- <!ATTLIST house id ID #REQUIED>
- ……

DTD-определение типа данных

- Продолжение: фрагмент локумента XML
- …….
- <house id="0">
- <rooms_num>5</rooms_num>
- <floor>2</floor>
- <living_space>32.5</living_space>
- <is_tel>true</is_tel>
- <counter>18346</counter>
- <price>100 р. 00 к.</price>
- </house>

Схемы данных

- Схемы данных (Schemas) являются альтернативным способом создания правил построения XML-документов. По сравнению с DTD, схемы обладают более мощными средствами для определения сложных структур данных, обеспечивают более понятный способ описания грамматики языка, способны легко модернизироваться и расширяться. Безусловным достоинством схем является также то, что они позволяют описывать правила для XML-документа средствами самого же XML

Схемы данных

- схемы не могут полностью заменить DTD-описания - этот способ определения грамматики языка используется сейчас практическими всеми верифицирующими анализаторами XML и, более того, сами схемы, как обычные XML-элементы, тоже описываются DTD.
- Но серьезные возможности нового языка и его относительная простота, безусловно, дают основания утверждать, что данный стандарт имеет широкое применение в качестве удобного и эффективного средства проверки корректности составления документов

Схемы данных

- Документ размечается при помощи специальных элементов, выполняющих в схемах роль инструкций. Эти инструкции составляют набор правил, используя которые, программа-клиент будет делать вывод о том, корректен документ или нет. Схема данных, например, может выглядеть следующем образом:

Схемы данных. Пример (1)

- <schema id="TeamSchema">
- <elementType id="#namee">
- <string/>
- </elementType>
- <elementType id="player">
- <element type="#name"/>
- <attribute name="number"/>
- <attribute name="type"/>
- </elementType>

Схемы данных. Пример (2)

- <elementType id="team">
- <element type="#player"/>
- <attribute name=”title”/>
- </elementType>
- </schema

XML документ, соответствующий данной схеме данных

- **<team title=”Celtics”>**
- **<player number="1" type=”goalkeeper”> <name>John Ree</ name></ player> <player number="2" type=”back”> <name>Peter Loyd</ name></ player> <player number="2" type=”forward”> <name>Emil McGeer</ name></ player> </team>**
- Все конструкции языка схем описываются правилами "XML DTD for XML-Data-Schema".

Схемы данных

- Если использовать отдельное пространство имен, то полный XML-документ, содержащий в себе схему данных, будет выглядеть следующим образом:
- **<?XML version='1.0' ?>**
- **<?xml:namespace href= "http://www.mrcpk.nstu.ru/schemas/" as="s"/?> <s:schema id="OurSchema">**
- **<!-- последовательность инструкций --> </s:schema>**

Описание элементов

- Для определения класса элемента, к которому в дальнейшем будут применяться инструкции, описывающие его содержимое и структуру, предназначен специальный элемент схемы **elementType.**
- Название элемента задается атрибутом id . Все дальнейшие инструкции, которые относятся к описываемому классу, определяют его внутреннюю структуру и набор допустимых данных, содержатся внутри блока, заданного тэгами <elementType> и </elementType>.
- При определении класса элемента, можно также использовать комментарии к нему, которые заключаются в тэги <**descript**></**descript**>

Атрибуты элемента

- Для того, чтобы в описании элемента определить его атрибуты и описать свойства этих атрибутов нужно использовать элемент **attribute**:
- **<elementType id="player">**
- **<attribute name="number"/>…**
- **</elementType>**
- В данном примере элементу <*player*> определяется атрибут *number*, значением которого может быть любая последовательность разрешенных символов:
- **<player number="0"/>**
- **<player number="some text"/>**

Атрибуты элемента

- Подобно DTD, схемы данных позволяют устанавливать ограничения на значения и способ использования атрибутов. Для этого в дескрипторе <attribute> необходимо использовать параметр **atttype**. Например, если мы хотим указать, что значение атрибута должно использоваться программой-анализатором как уникальный идентификатор, то нам необходимо создать следующее правило:
- **<elementType id="player">**
- **<attribute name="number" atttype="ID"/>**
- **</elementType>**

Атрибуты элемента

Если же требуется задать список возможных значений атрибута, то пример будет выглядеть следующим образом:

- **<attribute name="type" atttype="ENUMERATION«**

values="goalkeeper, back, halfback, forward">

Модель содержимого элемента

- Под моделью содержимого в схеме данных понимают описание всех допустимых объектов XML-документа, использование которых внутри данного элемента является корректным.
- Модель содержимого определяется инструкциями, расположенными внутри блока **<elementType>.** Вложенные элементы описываются при помощи инструкции **element**, в которой параметром type указывается класс объекта - ссылка на его определение:
- **<elementType id="player">**
- **<element Type="#name"/>**
- **<element Type="#nationality"/>**
- **</elementType>**

Атрибуты элемента

- Если требуется указать режим использования вложенного элемента, то надо определить параметр **occurs**:
- **<elementType id="player">**
- **<elementType="#name" occurs="REQUIRED"/>**
- **<elementType="#nationality" occurs="OPTIONAL"/>**
- **<elementType="#clubs" occurs="ONEORMORE"/>**
- **</elementType>**

Атрибуты элемента

- Возможные значения этого параметра таковы:
- REQUIRED - элемент должен быть обязательно определен
- OPTIONAL - использование элемента не является обязательным
- ZEROORMORE - вложенный элемент может встречаться несколько раз или ни разу
- ONEORMORE - элемент должен встречаться хотя бы один раз

Примеры правильных XML-документов

- **<player><name>John Ree</name>**
  <nationality>English</ nationality>
  <clubs>Celtics</clubs>
  <clubs>Portsmut</clubs>

</article>

или

- **<player><name>John Ree</name>**
  <clubs>Celtics</clubs>
  <clubs>Portsmut</clubs>
- **</article>**

Модель содержимого XML документа

- Кроме элементов, содержимым XML-документа могут также являться обычный текст и области CDATA. Для обозначения типов содержимого текущего элемента в схемах используются следующие инструкции:
- **<string/>** - указывает на то, что содержимым элемента является только свободная текстовая информация(секция PCDATA) :

```
<elementType id="name">
<string/>
</elementType>
```

Модель содержимого XML документа

- **<any/>** - указывает на то, что содержимым элемента должны являться только элементы, без текста, не заключенного ни в один элемент:

  <elementType id="coach">

  <any/>

  </elementType>
- **<mixed>** - любое сочетание элементов и текста

  <elementType id="player">

  <mixed/>

  </elementType>
- **<empty>** - пустой элемент.

Группировка элементов

- Элемент **group** используется для того, чтобы задать некоторую последовательность вложенных объектов:

```
<elementType id="team">
    <element type="#title" occurs="REQUIRED"/>
    <group occurs="OPTIONAL">
        <element type="#player">
        <element type="#assistant">
    </group>
</elementType>
```

Преобразование XML документов. XSLT и XPATH

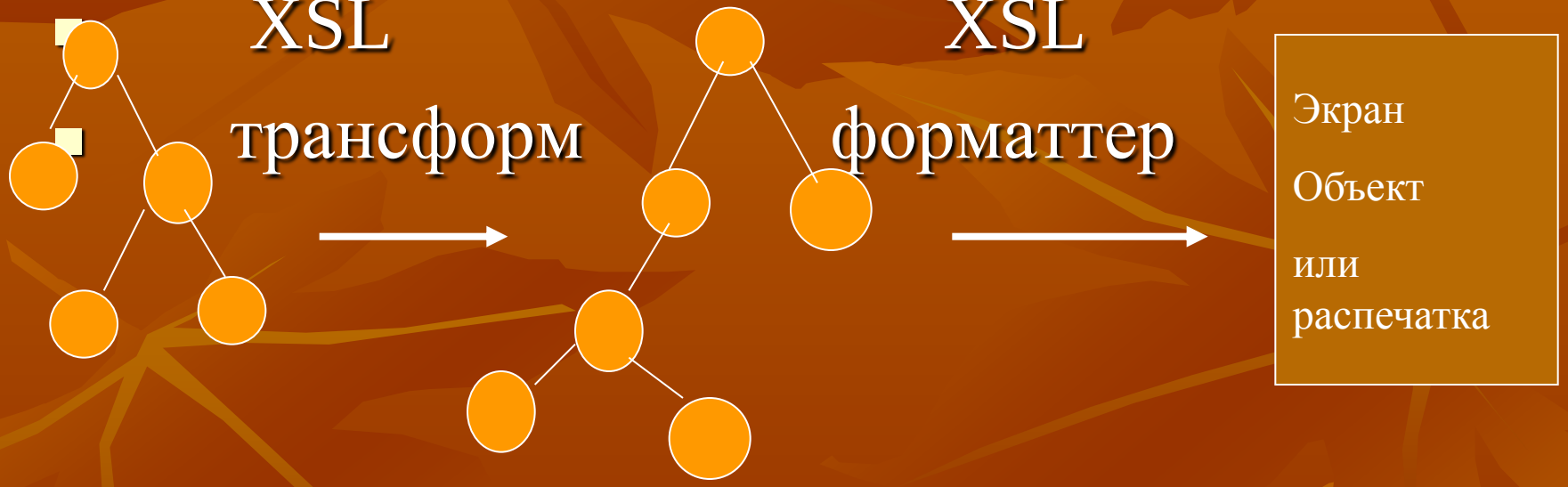
- Процесс преобразования и форматирования информации в готовый результат называется **стилизацией**. Чтобы стилизацию сделать возможной, существуют две рекомендации от W3C: **XSL преобразования** (XSLT), которые позволяют реорганизовывать информацию, а также сам язык **XSL**, который определяет как форматировать реорганизуемую информацию.

Этапы преобразования XML документов

- XSL stylesheet процессор производит представление исходного содержания XML к виду, определенному проектировщиком в stylesheet.
- Представление производится в 2 этапа:
- Первое – XML дерево (source tree) трансформируется в дерево преобразования (result tree);
- Второе – форматирование в представление пользователя. Этот процесс выполняется форматтером.

Этапы преобразования XML документов

- Данный процесс иллюстрируется следующей схемой



- Результат обработки XML дерева

Преобразование XML документов.

- Когда помещают XML и XSL данные в XSL-процессор, получают не просто улучшенный вариант XML кода. Получают результат в виде дерева, которое может быть расширено, изменено, либо реорганизовано.
- XSL процессор использует так называемый **стиль**, состоящий из набора XSL команд, который затем он преобразует, используя XML документ,

XSLT. Назначение и общее описание

- XSLT – язык преобразования XML документов в другие документы.
- XSLT является составной частью XSL, но может использоваться самостоятельно.
- XSL определяет стиль XML документа, используя XSLT.
- Помимо XSLT XSL содержит словарь XML для описания форматирования.

XSLT. Назначение и общее описание

- XSLT представляет собой правильно составленный документ XML и обладает определенной структурой (набором тэгов и атрибутов).
- Преобразование в языке XSLT предстает в виде корректного (правильного) XML документа, соответствующего требованиям для пространства имен XML, которое в спецификации XSLT называется пространством имен XSLT.

Описание стилей

- Преобразование, выраженное через XSLT , называется стилем (ctylesheet).
- Стили определяются набором команд XSL. Они создают корректные XML документы.
- Стили используют механизм совпадения паттернов (образцов) для нахождения элементов и атрибутов.

Описание стилей

- Описание стиля состоит из версии и пространства имен. Пространство имен объявляет префикс для тэгов, которые будут использованы в стиле и местонахождение описания тэгов:
- <xsl:stylesheet

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

- . . .
- </xsl:stylesheet>
- Если присутствуют ссылки на какие-либо расширения, то должно быть указано и пространство имен. Например, если нужно использовать язык Java, то необходимо указать это пространсво имен:
- xmlns:java=http://xml.apache.org/xslt/java

Структура таблицы стилей

- *В этом примере показана структура таблицы стилей. Многоточие (...) обозначает место, где значения атрибутов или содержимое опущены. Хотя в примере показан каждый допустимый элемент, таблицы стилей могут содержать ноль или более каждого из этих элементов.*
- *<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" > <xsl:import href="..."/> <xsl:include href="..."/> <xsl:strip-space elements="..."/> <xsl:preserve-space elements="..."/> <xsl:output method="..."/> <xsl:key name="..." match="..." use="..."/> <xsl:decimal-format name="..."/> <xsl:namespace-alias stylesheet-prefix="..." result-prefix="..."/>*

Структура таблицы стилей (продолжение)

- *<xsl:attribute-set name="..."> ... </xsl:attribute-set>*
  <xsl:variable name="...">...</xsl:variable>
- *<xsl:param name="...">...</xsl:param>*
- *<xsl:template match="..."> ... </xsl:template>*
- *<xsl:template name="..."> ... </xsl:template>*
  </xsl:stylesheet>
- *Порядок, в котором появляются потомки элемента xsl:stylesheet, не имеет значения, за исключением элементов xsl:import и обработки ошибок. Пользователи могут произвольно упорядочивать элементы, и утилиты создания таблиц стилей не обязаны предоставлять контроль над порядком появления элементов*

Преобразование XML документов.

Обратимся к примеру с использованием стиля stylesheet:

- <xsl:stylesheet xmlns:xsl="">
- <xsl:template match="/">
- <html> <body>
- <xsl:value-of select="employee/name"/> <xsl:text>: </xsl:text>
- <xsl:value-of select="employee/title"/> </body>
- </html>
- </xsl:template>
- </stylesheet>

Стиль может находиться либо в файле, либо в базе данных:

Описание стилей

- Стиль содержит набор правил шаблона.
- Правило шаблона состоит из двух частей: образца (pattern) и шаблона (template).
- Образец сопоставляется с узлами в исходном дереве (resource tree).
- Шаблон обрабатывается для формирования фрагментов в конечном дереве (result tree).

Такая схема позволяет использовать единый стиль для большого числа документов с одинаковой структурой исходного дерева

Применение шаблонов

- Примеры образцов выбора (math pattern):
- “/” –выбирает корневой узел
- “*” –выбирает элементы узлов (но не всех узлов);
- “library” –выбирает элементы library;
- “library/book”- выбирает все элементы book, дочерние для элемента library;
- “//library” выбирает все элементы library, производные от корневого узла;
- “.”- выбирает текущий узел но это не образец выбора, а выражение XPath/

Создание шаблонов

- Когда процессор XSLT находит узел, удовлетворяющий образцу, этот узел становится контекстным узлом шаблона и все операции производятся над этим узлом.
- На текущий узел можно ссылаться при помощи выражения XPath “.”
- В соответствии с правилами шаблона по умолчанию для обработки дочерних узлов необходимо использовать элемент
- “xsl:apple-templates”.

Синтаксис шаблона

- <xsl:template math= “nodename”>
- ……
- </xsl:template>
- Шаблон основан на имени ветви, поэтому все команды стилей применимы в этом шаблоне.
- Шаблон вызывается из используемого стиля командой apply-templates
- <xsl:apply-templates select=“nodename”/>

Значения типов элементов

- Команда <xsl:template match=“/”> означает выбор корневой ветви;
- Команда <xsl:apply-template> позволяет произвести новый поиск шаблонов для «текущего» узла;
- Команда <xsl:value of select..> дает возможность выбрать определенный элемент для анализируемого конечного XML документа.

Значения типов элементов

- Элемент <xsl:value of> записывает в результирующий документ строковое значение выражения. В частности, с его помощью можно возвратить значение узла, которым для элемента будет заключенный в него текст. Атрибуту выбора элемента <xsl:value of> можно присвоить выражение XPath, задающее узел или набор узлов

Значения типов элементов

- Паттерны позволяют выбирать определенные элементы из XML документа. Команда <xsl:value-of select... дает возможность выбрать требуемый элемент для конечного XML документа, как показано в таблице:

команда	результат
<xsl:value-of select= "employee/name"/> <xsl:value-of select= "employee/@id"/>	Joe Shmo 03432

Преобразование XML документов. пример

- Фрагмент XML документа имеет вид
- <employee id="03432">
- <name>Joe Shmo</name> <title>Manager</title>
- </employee>
- Если мы хотим чтобы наш HTML выглядел следующим образом:
- <html> <body> Joe Shmo: Manager </body> </html>

Преобразование XML документов.

Нужно использовать стиль stylesheet:

- <xsl:stylesheet xmlns:xsl="">
- <xsl:template match="/">
- <html> <body>
- <xsl:value-of select="employee/name"/> <xsl:text>: </xsl:text>
- <xsl:value-of select="employee/title"/> </body>
- </html>
- </xsl:template>
- </stylesheet>

Работа с XSLT (1)

- Версия planets.xml для Intertnet Explorer.6.0:
- <? Xml version=“1.0”?>
- <? Xsl: stylesheet type=“text/xsl” href =
  =“planet.xsl”?>
- <planets>
  - <planet>
    - <NAME>Venus </NAME>
    - <MASS UNITS=“(Earth=1)”>.815</MASS UNITS>
    - <DAY UNITS=“days”>58.65</DAY>
    - <RADIUS UNITS=“miles”>3716</RADIUS>
    - ……

  </planet> и т.д.

Работа с XSLT (2)

- <planet>
  - <NAME>Mercury </NAME>
  - <MASS UNITS="(Earth=1)">.0553</MASS UNITS>
  - ……
  - </planet>
- </planets>
- Версия **planets.xsl** для того же IE6.0 будет выглядеть следующим образом:

Работа с XSLT (3)

- <? xml version=“1.0”?>
- <? xsl: stylesheet xmlns: xsl=http://www.w3.org/TR/WD-xsl

  - <xsl: template math=“/”>
    - <HTML>
      - <HEAD>
        - <TITLE>The Planets Table </TITLE>

        </HEAD>

        <BODY>

        Прододжение на следующем слайде

Работа с XSLT (4)

```
<H11>The Planets Table</H1>
  <TABLE BORDER="2">
    <TR>
      <TD> Name</TD>
      <TD> Mass</TD>
      <TD> Radius</TD>
      <TD> Day</TD>
    </TR>
  <xsl: apply-templates/>
 </Table>
</Body>
</HTML>
</xsl: template>
```

Работа с XSLT (5)

- <xsl: template math="PLANETS">
- <xsl: apply-templates/>
- </xsl: template>
- <xsl: template math="PLANET">

<TR>
<TD>xsl: value of select="NAME"/></TD>
<TD> xsl: value of templates select ="Mass"/></TD>
<TD> xsl: value of templates select =" Radius"/></TD>
<TD> xsl: value of templates select =" Day"/></TD>
</TR>
</xsl: template>

Работа с XSLT

- Чтобы получить доступ к значениям атрибута при помощи выражений XPath, нужно добавить к имени атрибута префикс@. например “@MASS, “@Day” и т.д.Для выбора любого атрибута можно применить “@*”. Для выбора конкретного атрибута UNITS (ед. измерения) нужно в каждом элементе “MASS”,:RADIUS:, “DAY” использовать выражение “@UNITS”.

Работа с XSLT (6)

- <xsl: template math=“MASS”>
- <xsl: value of select=“.”/>

< xsl: value of select =“@UNITS”/>

</xsl: template>

- <xsl: template math=“RADIUS”>
- <xsl: value of select=“.”/>

< xsl: value of select =“@UNITS”/>

</xsl: template>

<xsl: template math=“DAY”>

<xsl: value of select=“.”/>

< xsl: value of select =“@UNITS”/>

</xsl: template>

</xsl: stylesheet>

Вид XMLдокумента в IE

NAME	MASS	RADIUS	DAY
VENUS	.815 (Earth=1)	3716	116
MERCURY	.0553 (Earth=1)	1516	58.65
EARTH	1 (Earth=1)	2107	1

Элемент “xsl:text”

- Текстовые узлы создаются при помощи элемента “xsl:text”, позволяющего по ходу дела замещать элементы целиком на текст. Одной из целей применения элемента “xsl:text”является сохраненбие символов – разделителей; вставить единственный пробел можно с помощью атрибута «disable-output-escaping». Устанавливается в yes для того, чтобы такие символы как”<“ и “>”выводились буквально, а не как &lt и соответственно &gt/ По умолчанию устанавливается в no.

Элемент “xsl:text”

- Пример:
- <?xml vtrsion=“1.0”?>
- <?xsl: stylesheet version=“1.0” <xmlns: xsl=“http:// www.w3.org/1999/XSL/Transformxsl”?>
- <xsl:template match”/PLANETS”>
- <HTML>
  - <HEAD>
  - <TITLE>
    thse Planets Table
  - </TITLE>

</HEAD>

Элемент “xsl:text”

- <body>
  - <H1>
    Thse Planets Table
    </H1>
    <Table>
    <TD>NAME</TD>
    <TD>Mass </TD>
    <TD>Radius </TD>
    <TD>Day</TD>
    <xsl:apple tamplates/>
    </Table>
    </BODY>
    </HTML>
    <xsl:template>

Элемент “xsl:text”

- <xsl:template match=“PLANET”>
  - <TR>
    - <TD><xsl:value-of select=“NAME/>”><TD>
    - <TD><xsl:value-of select=“MASS/>”><TD>
    - <TD><xsl:value-of select=“RADIUS/>”><TD>

    </TR>

    </xsl:template>
- <xsl:template match=“MASS”>

  <xsl:value of select=“.”/>

  <xsl:text> </xsl:text>

  <xsl: value of select=“@UNITS”/>

</xsl:template>

Элемент “xsl:text”

- <xsl:template match=“RADIUS”>
  <xsl:value of select=“.”/>
  <xsl:text> <xsl:text>
  <xsl: value of select=“@UNITS”/>
  </xsl:template>
- <xsl:template match=“Day”>
  <xsl:value of select=“.”/>
  <xsl:text> <xsl:text>
  <xsl: value of select=“@UNITS”/>
  </xsl:template>

</xsl: stylesheet>

Элемент “xsl:text”

NAME	MASS	RADIUS	DAY
MERCURY	0533 (Eath=1)	1.516 miles	58.65 days
VENUS	615 (Eath=1)	3716 miles	116.75 days
Eath	1 (Eath=1)	2107 miles	1 days

XSLT и XPath

- С точки зрения XSLT документы представляют собой образованные из узлов деревья.
- XSLT распознает 7 типов узлов:
- Корневой узел. Представляет для процессора XSLT весь документ. Следует различать корневой узел и корневой элемент;
- Узел атрибута;

XSLT и XPath (2)

- Узел комментариев;
- Узел элемента;
- Узел пространства имен;
- Узел инструкции обработки;
- Текстовый узел

XSLT и XPAth

- XSLT использует язык выражений XPATH для:
- Выбора узлов для обработки;
- Формулирования условий для разных вариантов обработки узла;
- Генерации текста для подстановки в конечное дерево.
- Выражение должно соответствовать сценарию Expr из Xpath.

Язык выражений XPATH

- Выражения используются:
- В значении определенных атрибутов у элементов, описанных в XSLT
- В фигурных скобках – в шаблоне для значений атрибута
- Внешние выражения – не являются частью других выражений; они получают свой контекст следующим образом:
- Узел контекста получается из текущего узла;

Язык выражений XPath

- Положение в контексте определяется положением текущего узла в текущем наборе узлов, первая позиция имеет индекс 1;
- Размер контекста определяется размером текущего набора узлов;
- Выражение XPath возвращает единственный удовлетворяющий выражению узел или множество узлов, если таких узлов несколько.

XSLT и XPath

- Основная задача XPath – адресовать части документа XML. Для реализации этой цели он предоставляет основные средства оперирования: строками, числами и логическими значениями.
- Синтаксис XPath отличен от синтаксиса XML, что облегчает его применение в идентификаторах URI и значениях атрибутов XML. Для навигации по иерархической структуре документа XML в нем используется нотация пути (patch) , как в идентификаторах URI.

Язык выражений XPath

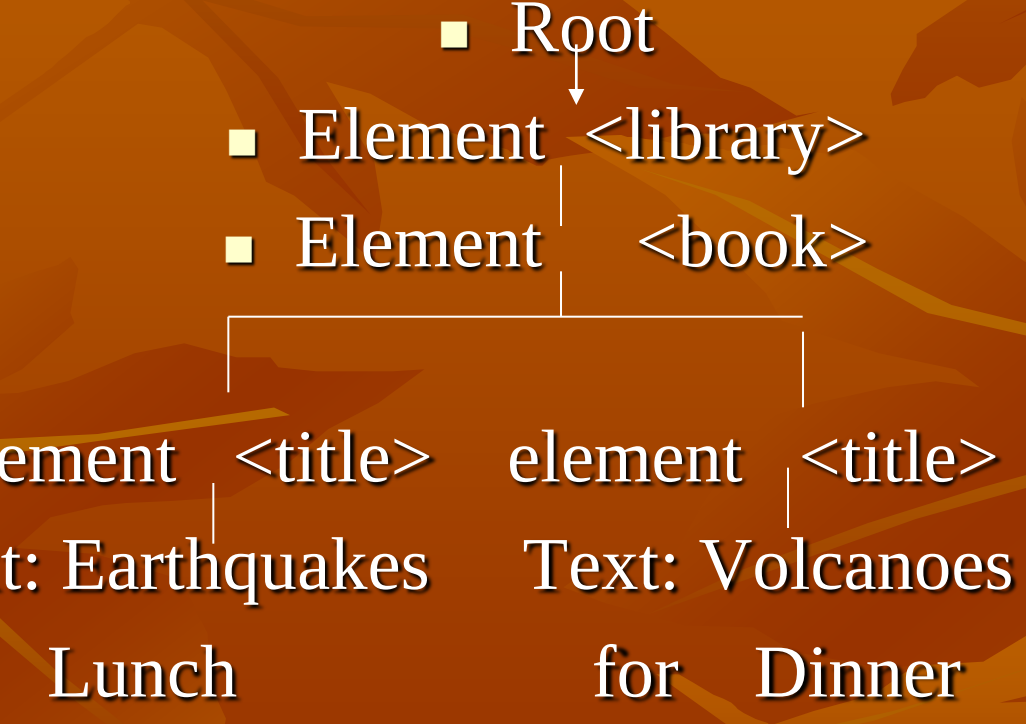
- XPath предоставляет средства для выбора набора узлов (node set), а также чисел, логических значений и строки.
- Полный синтаксис выражений XPatch приведен в спецификации XPatch

Язык выражений XPath

- Рассмотрим пример:
- <?xml version=“1.0”?>
- <library>
  - <book>
    - <title> Earthquakes for Lunch </title>
    - <title>Volcanoes for Dinner </title>
  - </book>
  - </library>

Язык выражений XPath

- Данный XML будет выглядеть в процессоре XSLT в виде дерева, состоящего из узлов:



Язык выражений XPath

- В этом дереве
- Root --Корневой узел;
- Library --Узел корневого элемента;
- Узел book имеет два дочерних узла title;
- Они же внуки для узла library;
- Родители и их родители – предки, а производные от них узлы - потомки .

Язык выражений XPath

- Выражения XPath применимы в XSLT не только в образцах выбора, но и в других приложениях: в атрибуте select элементов «xsl: apple_template», «xsl: value_of», «xsl:for-each», «xsl: param».«xsl: variable», в шаблонах значений атрибутов, в атрибуте test элементов «xsl: if», «xsl: when», атрибуте value элемента «xsl: number» и в предикатах образцов выбора.

Платформа MS.Net. WSDL

- WSDL (англ. Web Services Description Language)- язык описания Web служб или по другому Web Services.
- приложение "должно знать" интерфейс, чтобы взаимодействовать с Web-сервисом. Большинство прикладных сред способно генерировать файлы WSDL автоматически.

Платформа MS.Net. WSDL

- Для получения WSDL-файла соответствующего Web Services достаточно, например, сделать запрос следующего вида
- http://www.mywebservice.com/someServiceURL/wsdl?
- В ответ получим документ XML в формате, речь о котором пойдет дальше.

Логическая структура WSDL документа

- Логическая структура включает 3 уровня:
- Уровень 1 содержит два главных элемента: поддерживаемую версию языка XML и оператор определения верхнего уровня Definitions (определения) - это элемент, в котором определяется сервис и поддерживаемые им типы данных. В него включаются все элементы Уровня 2.

Логическая структура WSDL документа

- *Уровень 2* содержит определение сервиса (The Service Definition).
- Types (Типы). С помощью этого элемента определяются сложные типы данных, используемые в остальных элементах файла WSDL.
- Message Name (имя сообщения). Этот элемент помогает определить те сообщения, которые умеет обрабатывать программа. Здесь происходит отображение внешних имен сообщений в их внутренние имена.

Логическая структура WSDL документа

- Port Type (тип порта) – теперь interfase -это общедоступная точка входа для вызывающих приложений, где определяются поддерживаемые операции и сообщения, используемые каждой из этих операций.
- Binding Name (привязка имени). Этот элемент привязывает операцию к транспортному механизму.

Логическая структура WSDL документа

- Service Name (имя сервиса) - это публичная информация о сервисе. С помощью этого элемента приводятся имя сервиса и адрес, по которому его можно найти, а, возможно, и документация на него в читабельном формате.
- *Уровень 3* детализирует элементы Уровня 2

Структура WSDL

- В связи со значительностью изменений, внесенные в версию языка 1.1, WSDL 1.1 был переименован в WSDL 2.0. Ниже перечислены основные изменения:
- В язык WSDL добавлена дополнительная семантика, что явилось одной из причин, почему атрибут **targetNamespace** элемента **definitions** стал обязательным.
- Удалены конструкции сообщений. Теперь они задаются в элементе **types** при помощи системы типов XML-схемы.

Структура WSDL

- Отсутствует поддержка перегрузки операторов.
- Элемент **portType** переименован как **interface**. Поддержка наследования элемента **interface** достигается благодаря использованию атрибута **extends** в элементе **interface**.
- Элемент **port** переименован в **endpoint**.

Логическая структура WSDL документа

- Рассмотренные выше описания 1 и 2 уровней выглядят следующим образом:
- <?xml version= “1.0”?>
- <wsdl:definitions> ……..1 уровень
- <wsdl:types>...</wsdl:types> .
- <wsdl:message>...<wsdl:message/> .
- <wsdl:portType>...</wsdl:portType> …. 2 уровень
- <wsdl:binding>...</wsdl:binding> .
- <wsdl:service>...<wsdl:service> .
- </wsdl:definitions> .

Структура WSDL

- Описание Web-сервиса можно разделить на две части:абстрактное определение и реализацию. В абстрактной части описания Web-сервис описывается в языке WSDL с помощью системы типов, обычно W3C XML-схемы, в терминах сообщений, которые этот сервис отправляет и получает. Шаблоны обмена сообщениями определяют последовательность и количество сообщений.

Структура WSDL

- Концептуальная схема WSDL представлена следующим рисунком:

Структура WSDL

- Элемент **operation** связывает шаблоны обмена сообщениями с одним или несколькими сообщениями.
- Элемент **interface** группирует операции (элементы **operation**) независимо от транспорта и способа доставки.

Структура WSDL

- В конкретной части описания элементы **binding** задают транспорт и формат доставки для интерфейсов (элементов **interface**).
- Элемент сервиса (элемента **service**) **endpoint** связывает сетевой адрес в соответствие со связыванием (элементом **binding**).
- элемент **service** группирует конечные точки (элементы **endpoint**), которые реализуют общий интерфейс (элемент **interface**).

Компоненты WSDL

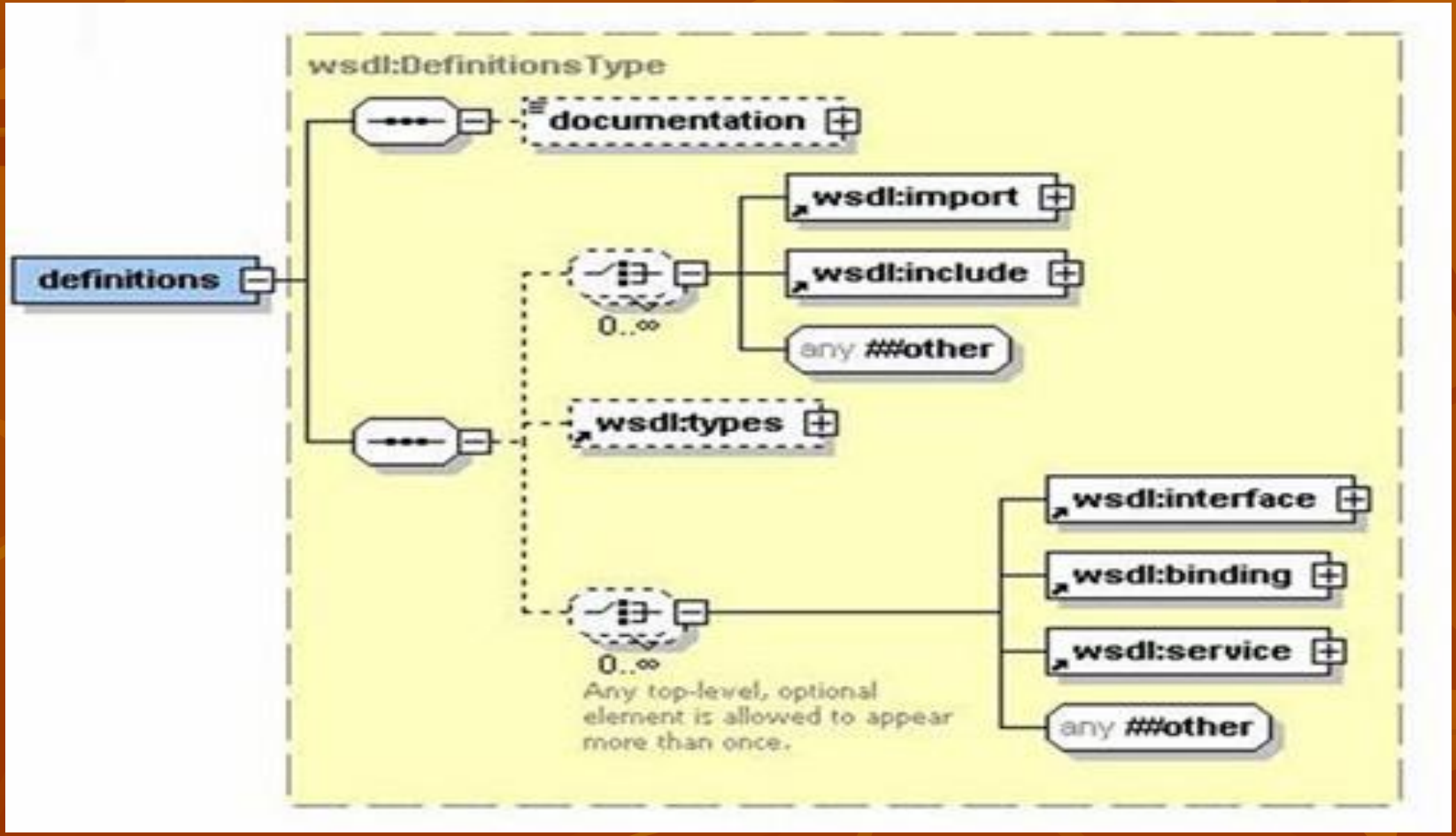
- Язык WSDL содержит ряд компонентов и их ассоциированных свойств, предназначенных для описания Web-сервисов.
- <definitions targetNamespace = “anyURI”>
- [<import /> | <include />] *
- <types /> ?
- [<interface /> | <binding /> | <service />] *
- </definitions>
-

Компоненты WSDL

- **definitions**
- Элемент **definitions** является корневым элементов любого документа WSDL. Он используется в качестве контейнера, в котором содержится вся необходимая информация о данной услуге и ее атрибутах. На рисунке 2 приведена схема элемента **definitions**.

Компоненты WSDL

- Рис.2 Схема элемента definitions



Компоненты WSDL

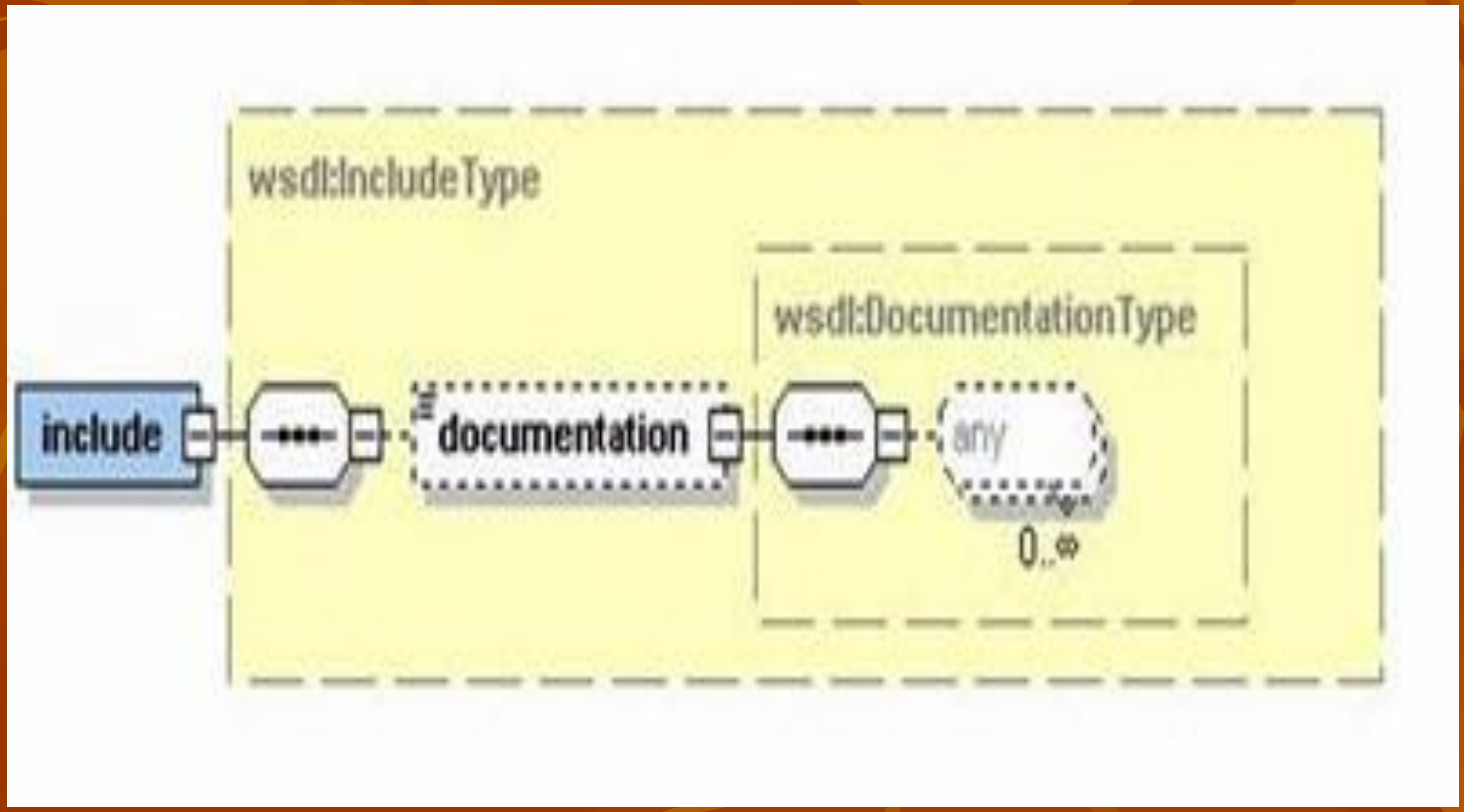
- Атрибут **targetNamespace** этого элемента является обязательным атрибутом типа **anyURI**. Это пространство имен может напрямую или косвенно определять семантику WSDL.
- элемент **definitions** может иметь другие необязательные атрибуты, соответствующие различным пространствам имен, которые могут использоваться в документе WSDL.

Компоненты WSDL

- **include**
- Элемент **include** предназначен для разделения описаний Web-сервиса на модули - различные компоненты описаний сервисов из одного и того же пространства имен могут находиться в другом документе WSDL, который можно использовать в описаниях Web-сервисов. Атрибут location является обязательным, он задает нахождение этих документов WSDL. Фактическое значение пространства имен добавляемого документа WSDL должно соответствовать целевому пространству имен элемента **definitions** в документе WSDL, в который добавляется первый документ

Компоненты WSDL

- На рисунке 3 приведена схема элемента **include**.

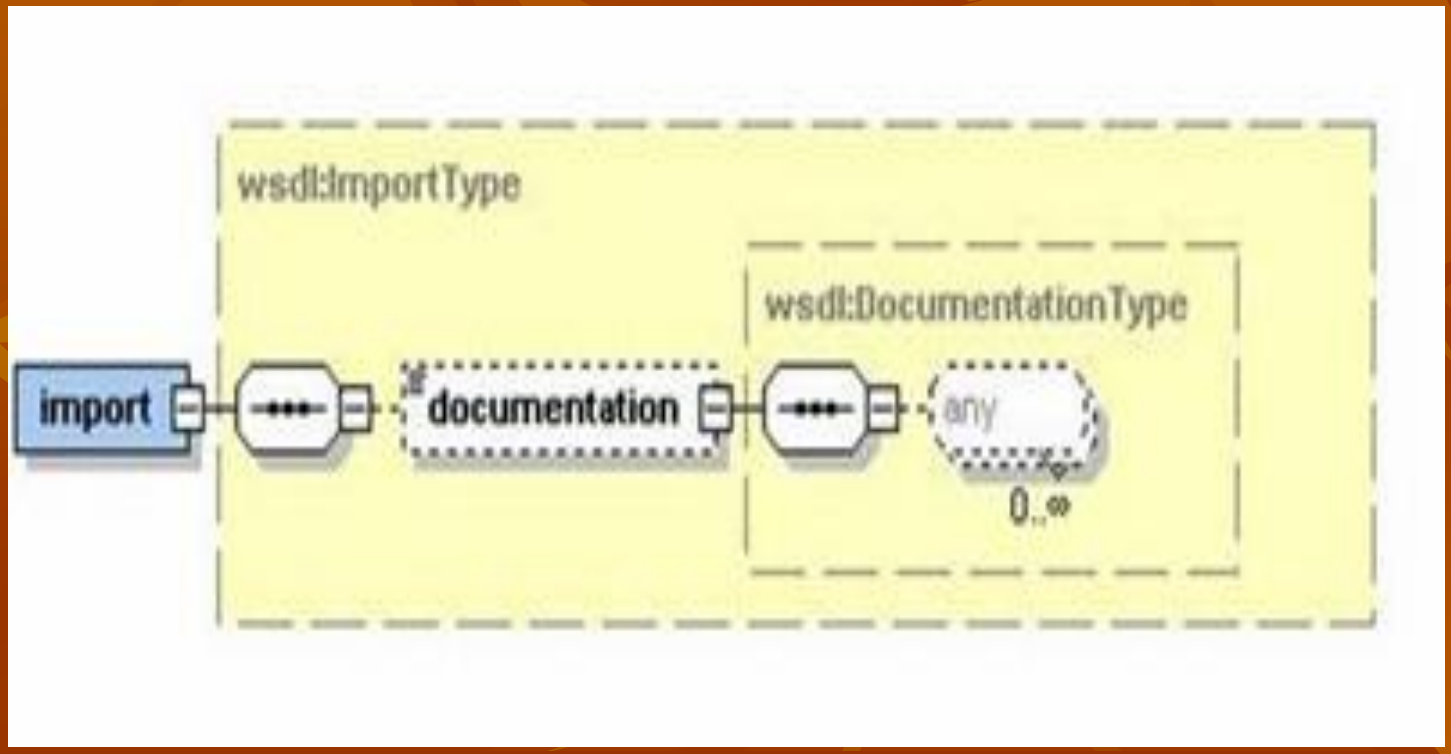


Компоненты WSDL

- **import**
- По своему назначению элемент **import** очень похож на элемент **include**, с тем исключением, что импортированный документ WSDL может относиться к другому целевому пространству имен. Атрибут **namespace** этого элемента является обязательным, а атрибут **location** - необязательным.

Компоненты WSDL

- *На рисунке 4 приведена схема элемента* ***import****.*

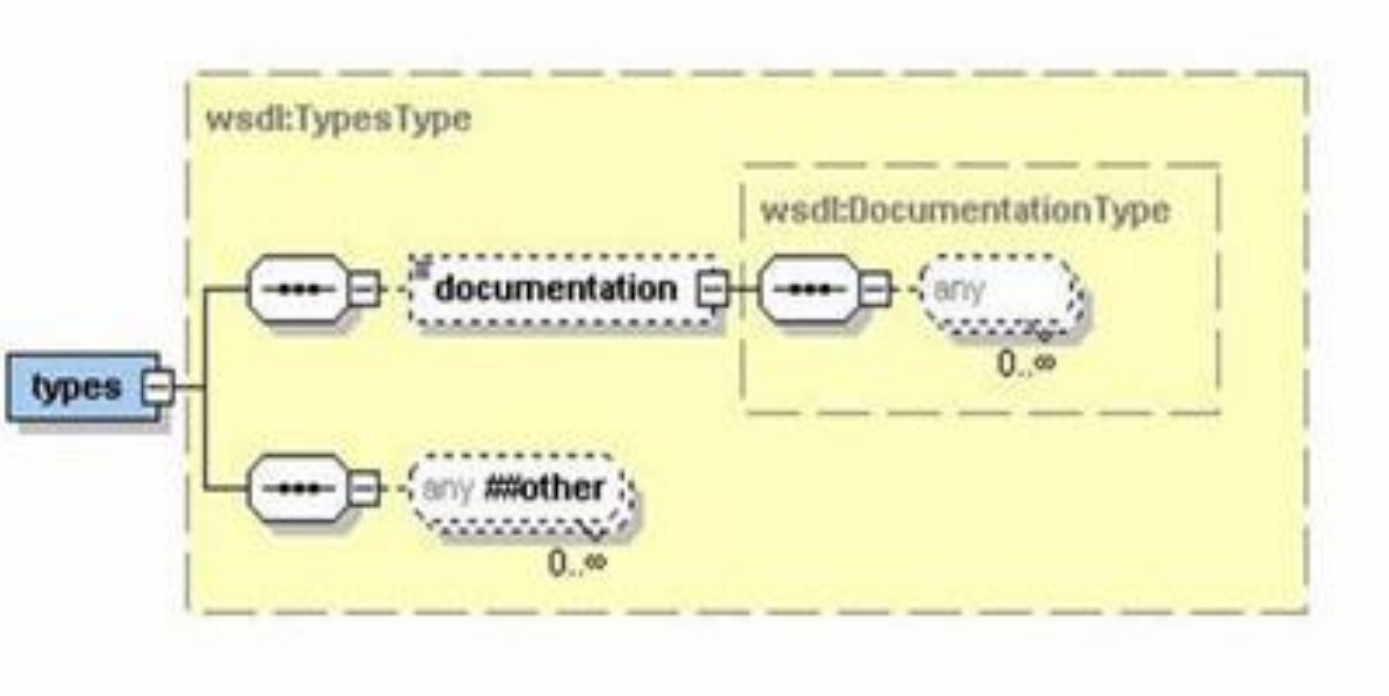


Компоненты WSDL

- **types**
- Элемент **types** определяет типы данных, которые используются при обмене сообщениями. Язык WSDL использует W3C XML-схему в качестве предпочтительного языка схемы. WSDL может использовать и другие системы, например, DTD. Чтобы воспользоваться схемами, их необходимо импортировать или внедрять в элементе **types** документа WSDL. Для импортирования используется конструкция **xs:import**, а для внедрения - **xs:schema**.

Компоненты WSDL

- Импортированные или внедренные компоненты схемы в документе WSDL указываются по **QName**. На рисунке 5 приведена схема элемента **types**.



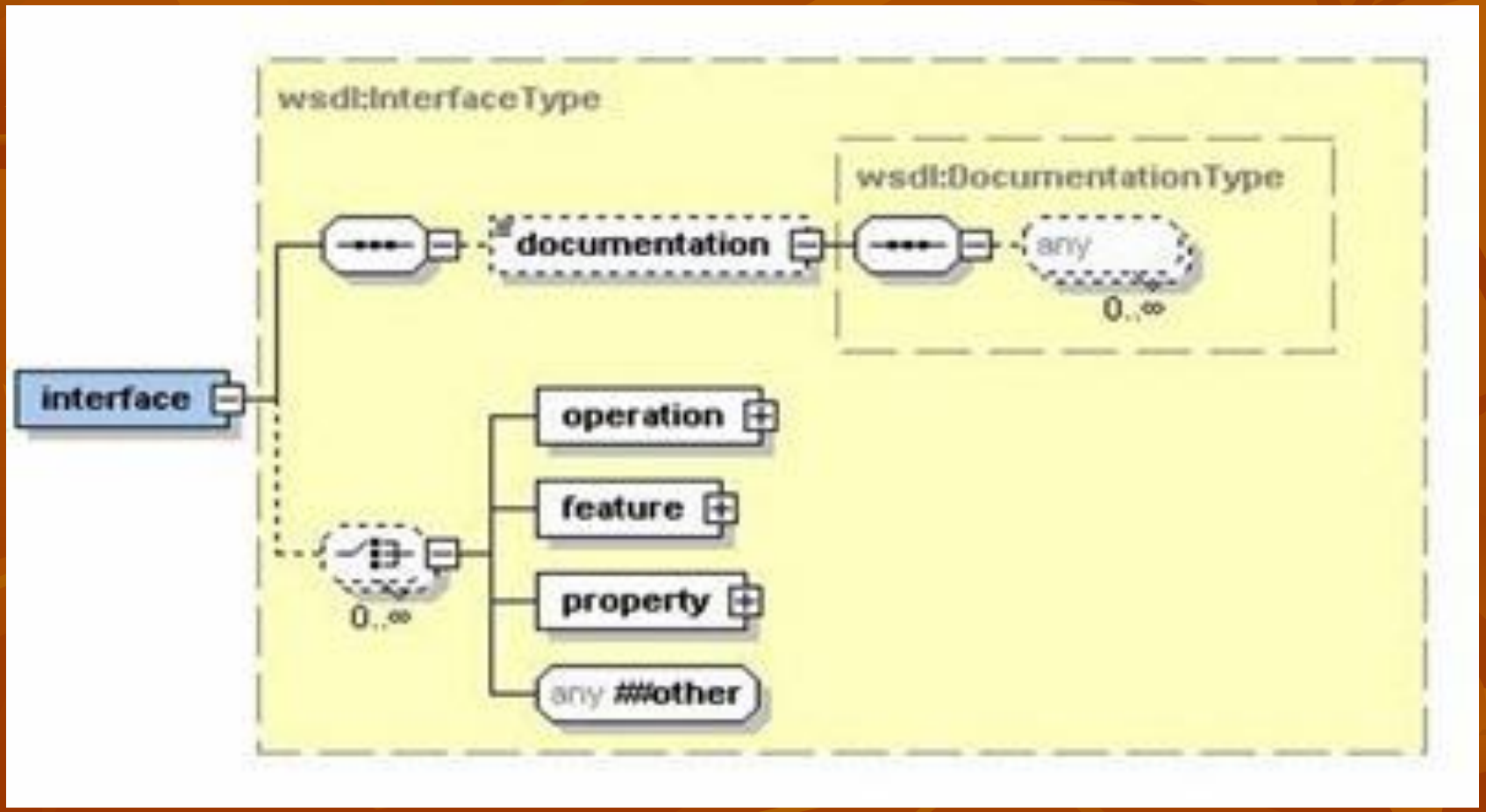
Компоненты WSDL

- **interface**
- Элемент **interface** содержит поименованный набор абстрактных операций и сообщений. При необходимости он может расширять один или несколько других элементов **interface**. Элементы **interface** в других компонентах, как, например, **binding**, указываются по **QName**. Элемент **interface** содержит элемент **operation**, атрибуты **name** и **pattern** которого является обязательными, а **style -** необязательным.

Компоненты WSDL

- На рисунке 6 приведена схема элемента **interface**. Элементы **feature** определяют функциональные возможности, связанные с обменом сообщениями между общающимися сторонами, что включает сведения о надежности, безопасности, зависимостях и маршрутизации. Элемент **property** используется для управления поведением элемента **feature**. Ряд возможных и допустимых значений этого элемента задается указаниями на описание схемы. Эти значения могут использоваться в нескольких элементах **feature**.

Компоненты WSDL



Компоненты WSDL

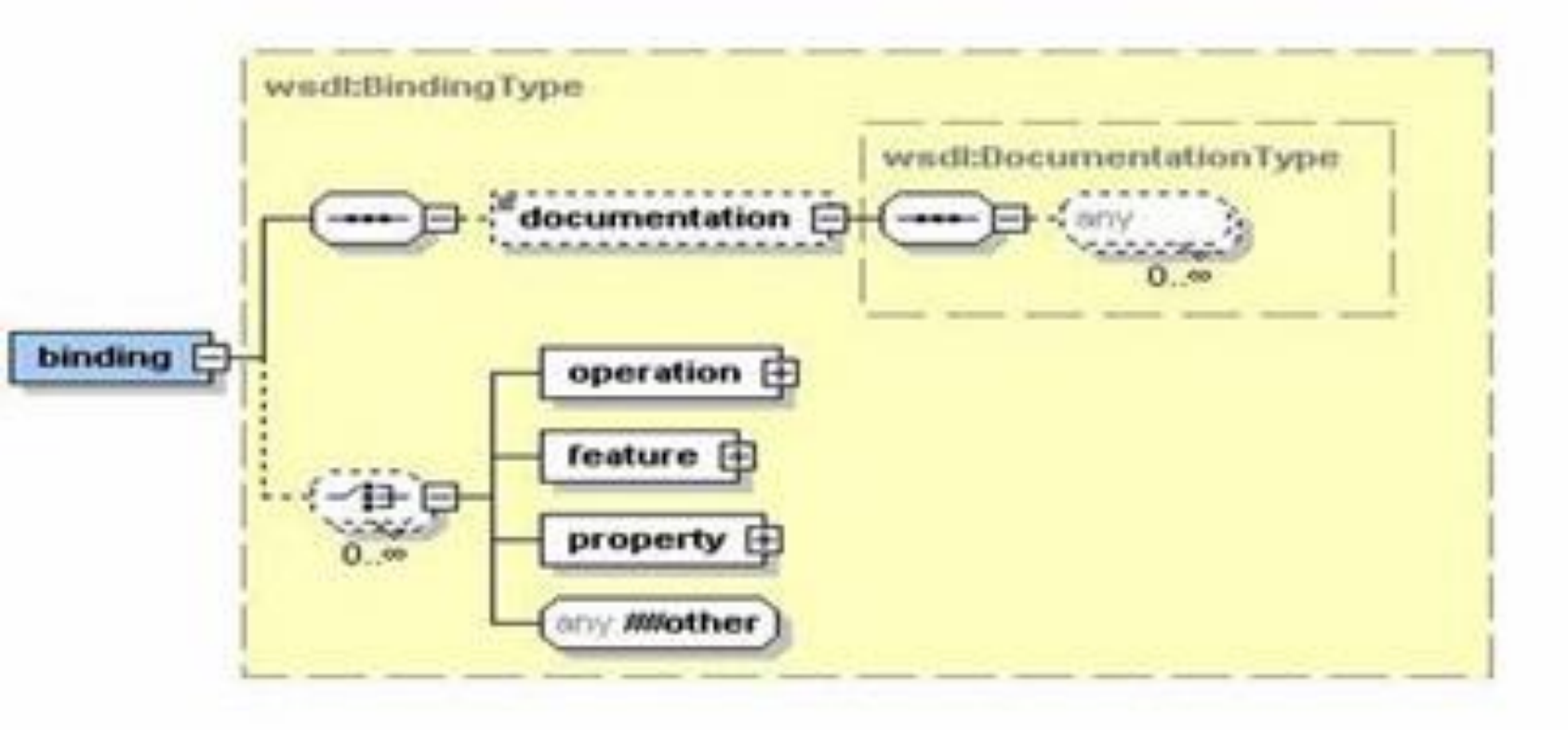
- **binding**
- Элемент **binding** определяет базовый транспорт и формат передачи для сообщений. Каждый элемент **binding** в документе WSDL указывает на элемент **interface**. Все элементы **operation**, определенные в элементе **interface**, должны быть связаны с этим **binding**. Элемент **endpoint** в компоненте **service** указывает элемент **binding**.

Компоненты WSDL

- И элементы **endpoint**, и элементы **binding** созданы для обеспечения гибкости и прозрачности местоположения. Многочисленные элементы **endpoint** с различным сетевым адресом могут использовать один и тот же протокол, определенный в **binding**.

Компоненты WSDL

- На рисунке 7 приведена схема элемента **binding**.

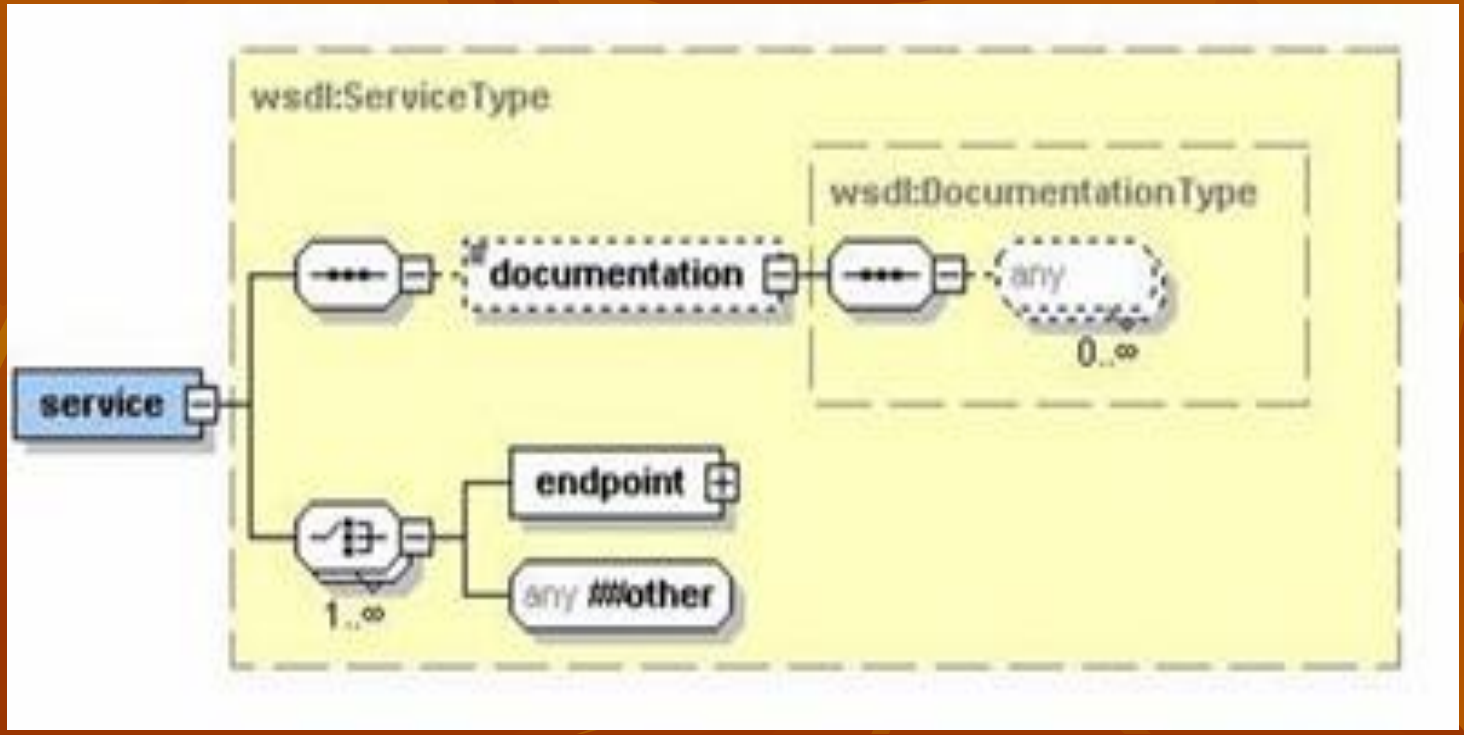


Компоненты WSDL

- **service**
- Элемент **service** описывает набор элементов **endpoint**, которые указывают на одиночный сетевой адрес для элемента **binding**. Вся другая информация о протоколе сдержится в элементе **binding**. К элементу **service** можно обратиться по **Qname**. У этого элемента есть обязательные атрибуты **name** и **interface**.

Компоненты WSDL

- На рисунке 8 приведена схема элемента **service**.



Содержимое WSDL-документа

- Экземпляр WSDL-документа является XML-документом с корневым элементом <*definitions*>, указывающим пространство имен языка WSDL (*http://schemas.xmlsoap.org/wsdl*) и определяющим набор веб-служб в качестве коллекции сетевых конечных точек (портов).

Пример WSDL документа (1)

- В следующем примере представлен упрощенный вариант документа WSDL 1.1 Specification Document комитета SOAP.
- Пример : SOAP 1.1 Request/Response с использованием HTTP:
- <?xml version="1.0"?>
- <definitions name="StockQuote"
- targetNamespace="http://example.com/stockquote.wsdl"

Пример WSDL документа (2)

- xmlns:tns="http://example.com/stockquote.wsdl"
  xmlns:xsd1="http://example.com/stockquote.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
- <schema targetNamespace=
  "http://example.com/stockquote.xsd"
- xmlns="http://www.w3.org/2000/10/XMLSchema">

Пример WSDL документа (3)

- <element name="TradePriceRequest">
- <complexType>
- <all>
- <element name="tickerSymbol“ type=“string"/>
- </all>
- </complexType>
- </element>

Пример WSDL документа (4)

```
<element name="TradePrice">
        <complexType>
            <all>
                <element name="price" type="float"/>
            </all>
        </complexType>
    </element>
</schema>
</types>
```

Пример WSDL документа (5)

- <message name="GetLastTradePriceInput">
- <part name="body" element="xsd1:TradePriceRequest"/>
- </message>
- <message name="GetLastTradePriceOutput">
- <part name="body" element="xsd1:TradePrice"/>
- </message>

Пример WSDL документа (6)

- <portType name="StockQuotePortType">
- <operation name="GetLastTradePrice">
<input message="tns:GetLastTradePriceInput"/>
<output message=
"tns:GetLastTradePriceOutput"/>
- </operation>
</portType>

Пример WSDL документа (7)

- <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
- <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
- <operation name="GetLastTradePrice">
- <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
- <input>
- <soap:body use="literal"/>
- </input>

Пример WSDL документа (8)

```
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
</binding>
```

Пример WSDL документа (9)

- <service name="StockQuoteService">
- <documentation>My first service</documentation>
- <port name="StockQuotePort" binding="tns:StockQuoteBinding">

<soap:address location="http://example.com/stockquote"/>

- </port>
- </service>

</definitions>

Комментарий к примеру

- Просматривая этот документ, можно прежде всего определить, где размещается данный сервис. Об этом сообщается в поле "*soap: Address location*" раздела определений сервиса файла WSDL: <soap: address location="http://example.com/stockquote">
- Чтобы выяснить, какие сообщения данный Web-сервис распознает, нужно найти определение сообщения *GetLastTradePriceInput*, использующего в качестве входных данных код эмитента ценных бумаг.
- Обратите внимание, что раздел *Message Name* определен при этом как элемент Уровня 2.

Комментарий к примеру (2)

- В данном примере имеется всего один элемент Уровня 3 - *part name* (имя компонента).
- <message name="GetLastTradePriceInput">
- <part name="Ticker Symbol" element="String"></message>
- И наконец, узнать о том, какую информацию мы получили от Web-сервиса в качестве ответа и в каком формате, можно из определения сообщения с именем *GetLastTradePriceOutput*:
- <message name="GetLastTradePriceOutput">
- <part name="price" element="float"></message>
- в ответном сообщении информация о курсе акций будет иметь вид числа с плавающей запятой

Использование языка WSDL

- Язык описания веб-служб (WSDL) определяет веб-интерфейс для любых функций RPC, расположенных на конечной точке HTTP, а также описывает возможности пакетов SQL для конечной точки.
- Клиент может запрашивать WSDL-ответ с экземпляра SQL Server и создавать с его помощью пакетные запросы RPC и SQL на сервере с использованием конечных точек HTTP, настроенных для поддержки WSDL-типа

Использование языка WSDL

- Фактически, ответ WSDL является динамически формируемым XML-документом, основанным на функциях RPC, связанных с конечной точкой на момент запроса.
- На конечных точках HTTP с поддержкой языка WSDL может быть указана одна из приведенных ниже конфигураций языка WSDL для работы с клиентами, отправляющими запросы WSDL:
- язык WSDL по умолчанию;
- пользовательский язык WSDL.

Web--сервисы

- ***"Веб-сервисы - это выделенные в самостоятельный элемент, слабосвязанные сжатые функции, предлагаемые по стандартным протоколам",*** где:
- "Выделенные в самостоятельный элемент" означает, что реализация этой функции никогда не заметна извне.
- "Слабо связанные" означает, что изменение одной функции не требует изменения других, вызывающих ее функций.
- "Ограниченные" означает, что существуют открытые для общего доступа описания поведения функции, способов использования, а также ее входных и выходных параметров.

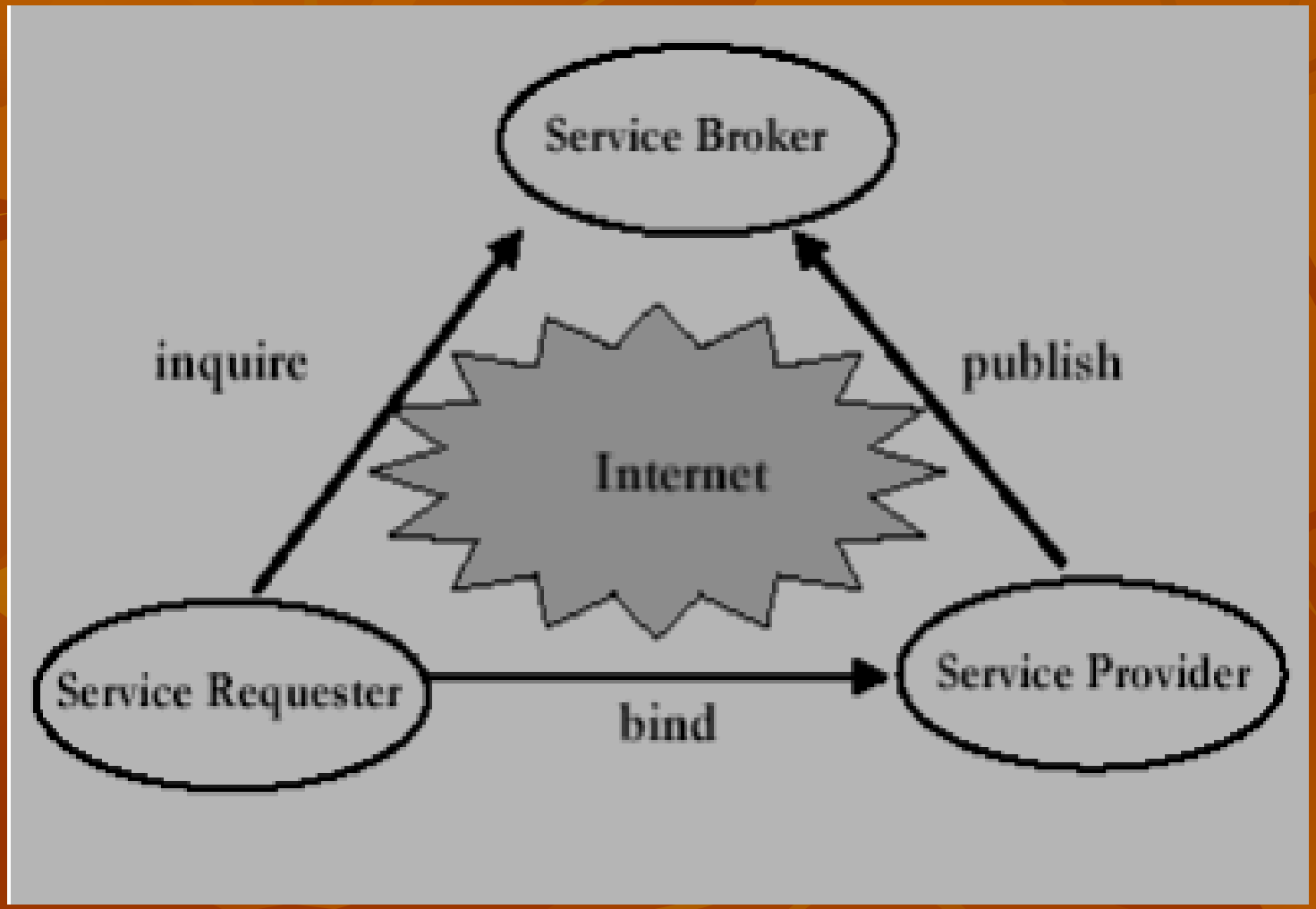
Web—сервисы. Архитектура

- Согласно традиционным представлениям клиент-серверного подхода существовал сервер, предлагающий какие-либо функциональные возможности, которые могли быть использованы или вызваны клиентом. Механизм, похожий на поисковую службу, исполнял роль агента между этими клиентом и сервером.
- Поскольку веб-сервисы представляют просто еще одну парадигму для распределенных приложений, они состоят их тех же самых трех компонентов:

Web—сервисы. Архитектура

- Сервисного агента, играющего роль поисковой службы между поставщиком и инициатором сервисного запроса.
- Поставщика сервисов, который публикует свои сервисы для сервисного агента.
- Инициатора сервисного запроса, который запрашивает у сервисного агента информацию о том, где найти подходящего поставщика сервисов, а затем связывается с этим поставщиком.

Web—сервисы. Архитектура



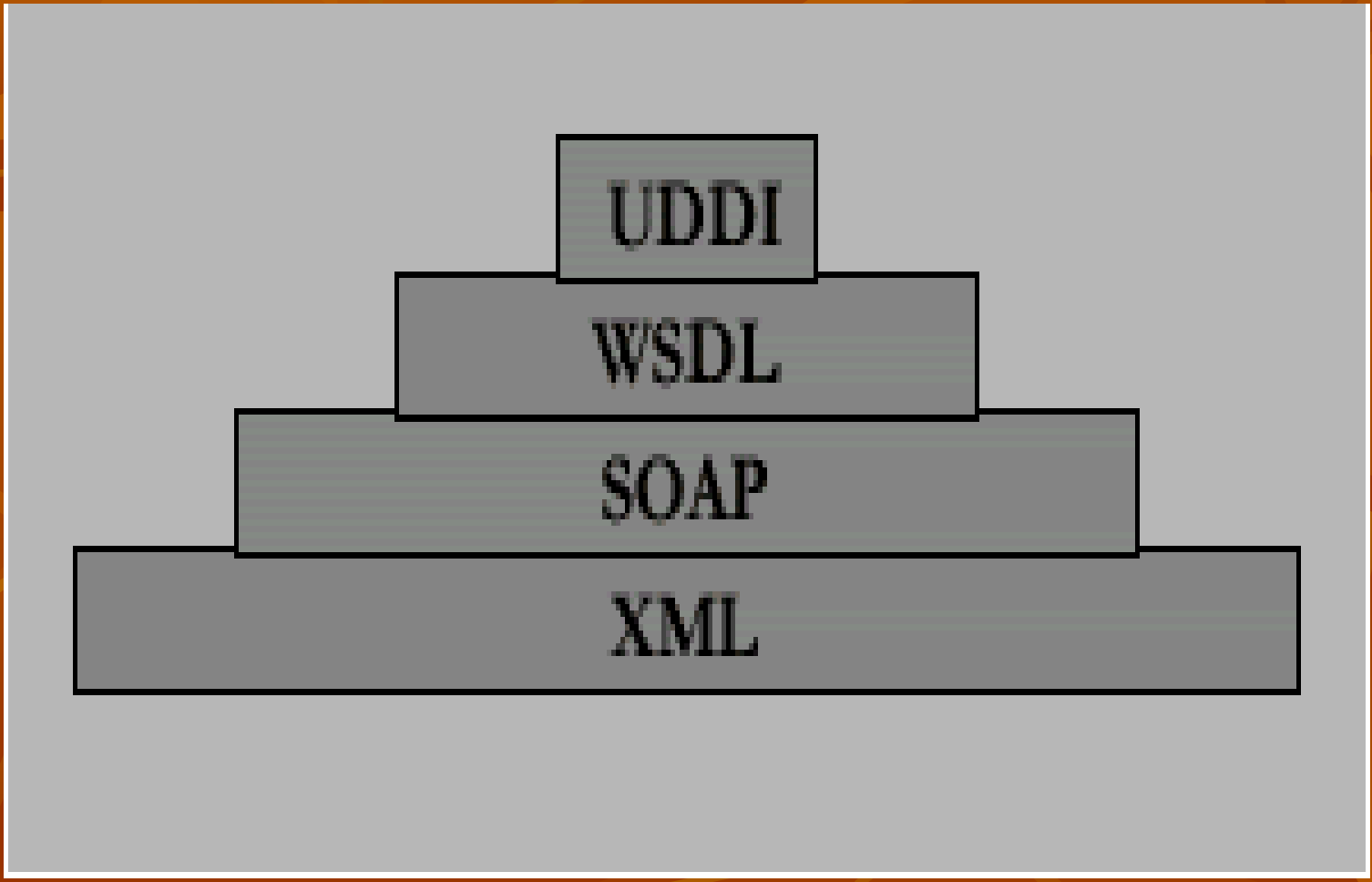
Web—сервисы. Архитектура

- **Подписи к рисунку**
- Service Broker - Сервисный агент
- Inquire - Запрос
- Publish - Публикация
- Internet - Интернет
- Service Requester -- Инициатор сервисного запроса
- Service Provider - Поставщик сервисов
- Bind - Соединение

Стек Web—сервисов.

- стек веб-сервисов, состоит из следующих частей:
- язык расширяемой разметки XML (Extensible Markup Language);
- протокол доступа к простым объектам SOAP (Simple Object Access Protocol);
- язык определения веб-сервисов WSDL (Web Services Definition Language);
- универсальная интеграция поиска описаний UDDI (Universal Discovery Description Integration).

Стек Web—сервисов



Стек Web—сервисов.Характеристика уровней

- **SOAP**
- Поскольку веб-сервисы будут выполняться в гетерогенных средах, протоколы, используемые для переноса данных между функциями, должны быть независимы от среды выполнения. SOAP - это протокол, обладающий такими свойствами.

Стек Web—сервисов.Характеристика уровней

- **WSDL**
- WSDL описывает сетевые сервисы с помощью грамматики XML. В рамках этой технологии обеспечивается документация для распределенных систем. Ее цель - дать приложениям возможность взаимодействовать друг с другом в автоматическом режиме.
- В то время как SOAP определяет взаимодействие между инициатором запроса и поставщиком, WSDL описывает сервисы, предлагаемые поставщиком (в "конечной точке"), которые могут использоваться как средство создания правильных сообщений SOAP для доступа к сервисам.

Стек Web—сервисов.Характеристика уровней

- **UDDI**
- UDDI (Universal Description, Discovery, and Integration) - это стандарт, разработанный для создания пригодного к поиску каталога предприятий и предоставляемых ими веб-сервисов. Таким образом, он является чем-то вроде сервисного агента, который помогает инициаторам сервисных запросов находить подходящих поставщиков сервисов. В отношении многих аспектов UDDI проектируется также как телефонная книга. В нем предусмотрена поддержка для следующих типов данных:
- .

Стек Web—сервисов.Характеристика уровней

- систематические каталоги желтых страниц
  С их помощью можно выполнять поиск предприятий, которые предлагают сервисы для определенной отрасли или категории продуктов, а также тех, которые расположены в конкретной географической области.
- белые страницы
  Информация о поставщиках сервисов, включая адрес, контактные данные и известные идентификаторы.
- зеленые страницы
  Техническая информация о веб-сервисах, которая опубликована поставщиком, например, сведения о том, как взаимодействовать с веб-сервисом

Стек Web—сервисов.Характеристика уровней

- Чтобы получить доступ к UDDI-сервисам, каталог UDDI предлагает набор программных интерфейсов API в виде созданных на основе SOAP веб-сервисов. Эти интерфейсы поделены на две логических части: интерфейс API для запросов и интерфейс API для публикаций. Интерфейс API для запросов состоит из двух следующих частей: одна из них используется для создания программ, позволяющих осуществлять поиск информации в реестре UDDI, а затем просматривать ее, а вторая часть используется в случае возникновения сбоев.

Сервис –ориентированная архитектура

- **Се́рвис-ориенти́рованная архитекту́ра** (англ. SOA, service-oriented architecture) — модульный подход к разработке программного обеспечения, основанный на использовании сервисов (служб) со стандартизированными интерфейсами.

Определение SOA

- OASIS (Организация по распространению открытых стандартов структурированной информации) определяет SOA следующим образом (OASIS Reference Model for Service Oriented Architecture V 1.0): Сервисно-ориентированная архитектура — это парадигма организации и использования распределенных информационных ресурсов таких как приложения и данные, находящихся в сфере ответственности разных владельцев, для достижения желаемых результатов потребителем, которым может быть: конечный пользователь или другое приложение

Сервис –ориентированная архитектура

- В основе SOA лежат принципы многократного использования функциональных элементов ИТ, ликвидации дублирования функциональности в ПО, унификации типовых операционных процессов, обеспечения перевода операционной модели компании на централизованные процессы и функциональную организацию на основе промышленной платформы интеграции.

Сервис –ориентированная архитектура

- Компоненты программы могут быть распределены по разным узлам сети, и предлагаются как независимые, слабо связанные, заменяемые сервисы-приложения.
- Программные комплексы, разработанные в соответствии с SOA, часто реализуются как набор веб-сервисов, интегрированных при помощи известных стандартных протоколов (SOAP, WSDL, и т. п.)

Сервис –ориентированная архитектура

- Интерфейс компонентов SOA-программы инкапсулирует детали реализации конкретного компонента (ОС, платформы, языка программирования, вендора, и т. п.) от остальных компонентов.
- Таким образом, SOA предоставляет гибкий и элегантный способ комбинирования и многократного использования компонентов для построения сложных распределённых программных комплексов.

Принципы SOA

- Архитектура, как таковая, не привязана к какой-то определённой технологии,
- Независимость организации системы от используемой вычислительной платформы (платформ),
- Независимость организации системы от применяемых языков программирования,
- Использование сервисов, независимых от конкретных приложений, с единообразными интерфейсами доступа к ним,
- Организация сервисов как слабо-связанных компонентов для построения систем

Другие SOA-Концепции

- SOA также может рассматриваться как стиль архитектуры информационных систем, который позволяет создавать приложения, построенные путём комбинации слабо-связанных и взаимодействующих сервисов. Эти сервисы взаимодействуют на основе какого-либо строго определённого платформенно-независимого и языково-независимого интерфейса (например, WSDL). Определение интерфейса скрывает языково-зависимую реализацию сервиса

Другие SOA-Концепции

- Таким образом, системы, основанные на SOA, могут быть независимы от технологий разработки и платформ (таких как Java, .NET и т. д.). К примеру, сервисы, написанные на C#, работающие на платформах .Net и сервисы на Java, работающие на платформах Java EE, могут быть с одинаковым успехом вызваны общим составным приложением. Приложения, работающие на одних платформах, могут вызывать сервисы, работающие на других платформах, что облегчает повторное использование компонентов.

Другие SOA-концепции

- SOA может поддерживать интеграцию и консолидацию операций в составе сложных систем, однако SOA не определяет и не предоставляет методологий или фреймворков для документирования сервисов

Другие SOA-концепции

- Языки высокого уровня, такие как BPEL, или спецификации, такие как WS-CDL и WS-Coordination, расширяют концепцию сервиса, предоставляя метод оркестрации, для объединения мелких сервисов в более обширные бизнес-сервисы, которые, в свою очередь, могут быть включены в состав технологических процессов и бизнес-процессов, реализованных в виде составных приложений или порталов

Сервисно-ориентированная архитектура баз данных

- Развитие архитектуры SOA. База данных является хранилищем сообщений, промежуточных состояний, метаинформации об очередях сообщений и сервисах. Отправка сообщений в очередь и прием сообщений из очереди производится в одной транзакции с изменением данных, что обеспечивает транзакционную целостность системы.

Сервисно-ориентированная архитектура баз данных

- Так как очереди сообщений и данные хранятся и обрабатываются в базе единообразно, это обеспечивает гарантированную доставку и обработку сообщений в случае сбоев оборудования или питания с таким же успехом, как и прочих данных, хранящихся в той же базе данных. Кроме этого, в базе данных хранится информация о самих сервисах и обрабатываемых ими очередях сообщений, что обеспечивает восстановление после сбоя состояний не только данных и сообщений, но и настроек сервисов и очередей сообщений

SOAP-Simple Objects Access Protocol

- SOAP-сообщение формально задается как информационное множество XML [XML Infoset]], дающее абстрактное описание его содержимого. Информационные множества могут иметь различные представления, один из общих примеров которых находится в документе XML 1.0 [XML 1.0].

SOAP-Simple Objects Access Protocol

- SOAP является парадигмой, в основе своей не имеющей состояний, характеризующейся односторонним обменом сообщениями.
- Однако приложения могут создавать более сложные шаблоны взаимодействия (например, запрос/отклик, запрос/множественные отклики и т. д.), сочетая односторонний обмен с функциональностью, предоставляемой нижележащим протоколом и/или специфичной для конкретного приложения информацией.

SOAP-Simple Objects Access Protocol

- SOAP умалчивает о семантике каких бы то ни было специфичных для приложения данных, которые он передает - это является прерогативой таких вопросов, как маршрутизация SOAP-сообщений, надежность передачи данных, обеспечение безопасности посредством брандмауэра и т. д.
- SOAP представляет собой структуру, опираясь на которую, можно весьма гибко передавать специфичную для приложений информацию. Помимо этого, SOAP дает полное описание необходимых действий SOAP-узла при получении SOAP-сообщения.

SOAP-Simple Objects Access Protocol

- В своей основе SOAP-сообщение является единицей односторонней передачи данных между SOAP-узлами, от SOAP-отправителя к SOAP-получателю, поэтому SOAP-сообщения должны компоноваться приложениями с целью реализации более сложных шаблонов взаимодействия, начиная шаблонами взаимодействия типа "запрос/отклик" и кончая множественными двунаправленными (двусторонними) "диалоговыми" вариантами обмена сообщениями.

SOAP-Simple Objects Access Protocol

- Пример 1 показывает данные, необходимые для работы приложения бронирования путешествий, в виде SOAP-сообщения.
- <?xml version='1.0' ?> //заголовок//
- <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
- <env:Header>
- <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
- env:role="http://www.w3.org/2003/05/soap-envelope/role/next"

SOAP-Simple Objects Access Protocol

- env:mustUnderstand="true">
- <n:name>Eke Jygvan Iyvind</n:name>
  </n:passenger>
- </env:Header>
- Префиксы пространств имен "env", "enc" и "rpc", используемые далее, обозначают пространства имен "http://www.w3.org/2003/05/soap-envelope" , "http://www.w3.org/2003/05/soap-encoding", и "http://www.w3.org/2003/05/soap-rpc" соответственно.

SOAP-Simple Objects Access Protocol

- <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
  <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
- </m:reservation>
- <n:passengerxmlns:n="http://mycompany.example.com/employees"
  env:role="http://www.w3.org/2003/05/soap-envelope/role/next"

SOAP-Simple Objects Access Protocol

- Префиксы пространств имен "xs" и "xsi", используемые в тексте, обозначают пространства имен "http://www.w3.org/2001/XMLSchema" и "http://www.w3.org/2001/XMLSchema-instance" соответственно, которые определены в спецификациях XML Schema [XML Schema Часть 1], [XML Schema Часть 2]

SOAP-Simple Objects Access Protocol

- <env:Body> //тело сообщения//
- <p:itinerary xmlns:p="http://travelcompany.example.org/reservation/travel">
- <p:departure>
- <p:departing>New York</p:departing>
  <p:arriving>Los Angeles</p:arriving>
  <p:departureDate>2001-12-14</p:departureDate>
  <p:departureTime>late afternoon</p:departureTime>
  <p:seatPreference>aisle</p:seatPreference>
  </p:departure>

SOAP-Simple Objects Access Protocol

- <p:return>
- <p:departing>Los Angeles</p:departing>
  <p:arriving>New York</p:arriving>
  <p:departureDate>2001-12-20</p:departureDate>
- <p:departureTime>mid-morning</p:departureTime>
  <p:seatPreference/>
- </p:return>
- </p:itinerary>

SOAP-Simple Objects Access Protocol

- <q:lodging xmlns:q="http://travelcompany.example.org/reservation/hotels"> <q:preference>none</q:preference> </q:lodging>
- </env:Body>
- </env:Envelope>
- Образец SOAP-сообщения бронирования путешествия, содержащий заголовочный блок и тело сообщения

SOAP-Simple Objects Access Protocol

- Заголовочный элемент SOAP не является обязательным, однако он был включен в пример для того, чтобы объяснить некоторые функции SOAP. Заголовок SOAP является расширением, предоставляющим способ передачи в SOAP-сообщениях информации, вообще говоря не являющейся полезной для приложения.
- Подобная "контрольная" информация включает, например, директивы прохождения сообщения или контекстную информацию, относящуюся к обработке сообщения. Это позволяет подстраивать SOAP-сообщения под каждое конкретное приложение.
- Следующие непосредственно за env:Header дочерние элементы называются заголовочными блоками. Они представляют логическую группировку данных, которые могут быть индивидуально адресованы SOAP-узлам, встречаемым сообщением на пути от отправителя к конечному получателю

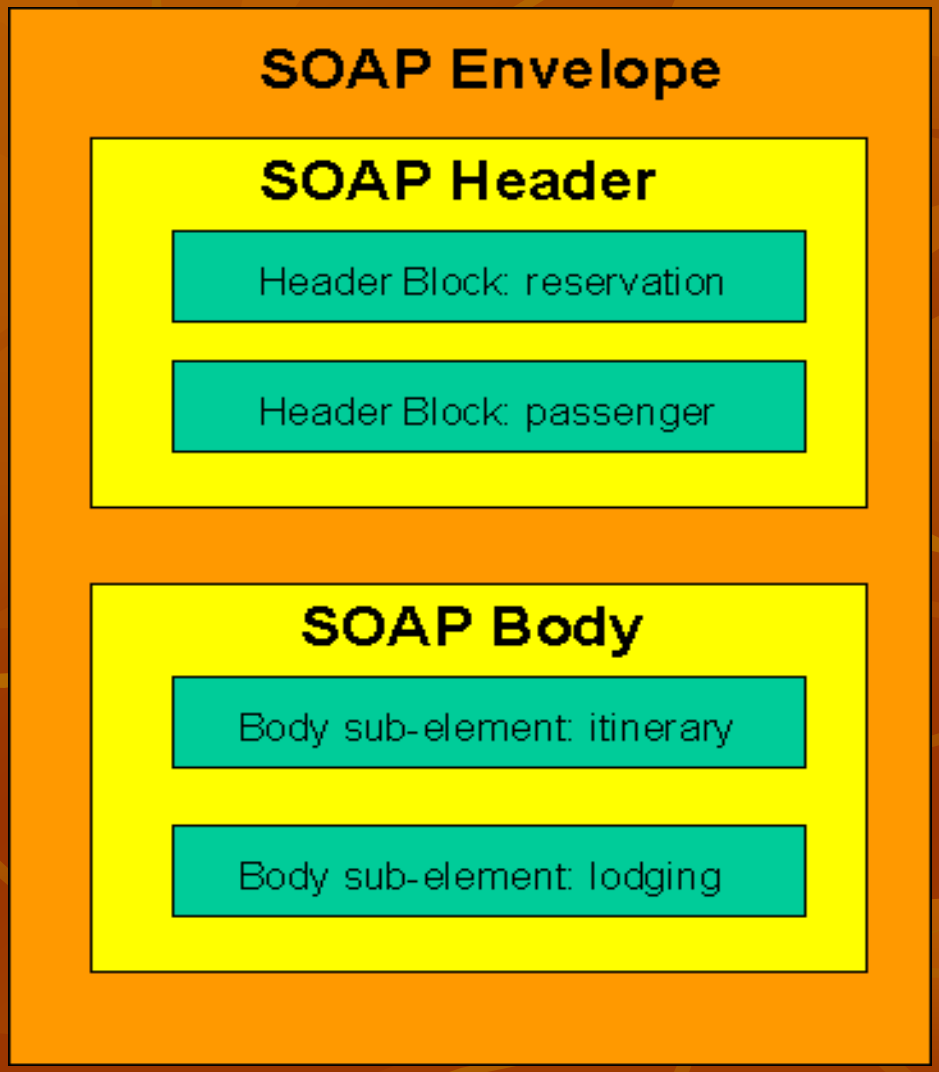
SOAP-Simple Objects Access Protocol

- Заголовки SOAP были созданы в предположении появления различных вариантов использования SOAP, многие из которых будут вовлекать во взаимодействие другие обрабатывающие SOAP-сообщения узлы, называемые SOAP-посредниками - на пути сообщения от начального отправителя SOAP-сообщения до конечного SOAP-получателя. Это позволяет SOAP-посредникам предоставлять дополнительные сервисы.
- Заголовки могут быть просмотрены, вставлены, удалены или пересланы SOAP-узлами, встреченными на пути SOAP-сообщения.

SOAP-Simple Objects Access Protocol

- Тело SOAP-сообщения является обязательным элементом внутри env:Envelope, содержащим основную информацию SOAP-сообщения, которая должна быть передана из начальной точки пути сообщения в конечную.
- Графическое представление структуры SOAP-сообщения, приведенного в примере 1 выглядит как показано на следующем слайде

SOAP-Simple Objects Access Protocol



SOAP-Simple Objects Access Protocol

- Заголовочные блоки reservation и passenger должны обрабатываться следующим SOAP-посредником, встреченным на пути сообщения, либо, в случае отсутствия узла-посредника, конечным получателем сообщения. На тот факт, что сообщение адресовано следующему SOAP-узлу, встреченному на пути сообщения, указывает присутствие атрибута env:role со значением "http://www.w3.org/2003/05/soap-envelope/role/next" (здесь и далее просто "next"). Этот атрибут реализует роль, исполнять которую обязаны все SOAP-узлы. Присутствие же атрибута env:mustUnderstand со значением "true" указывает на то, что узел (узлы), обрабатывающий заголовочные блоки, должен обрабатывать их в строгом соответствии с их спецификациями либо не обрабатывать SOAP-сообщение вовсе и выдать сообщение об ошибке.

SOAP-Simple Objects Access Protocol

- Решение по поводу того, какие данные поместить в заголовочный блок и тело SOAP-сообщения, должно быть принято на этапе проектирования приложения. Необходимо помнить, что заголовочные блоки могут быть адресованы различным узлам, которые могут встречаться на пути сообщения от отправителя к конечному получателю. Подобные промежуточные SOAP-узлы могут предоставлять дополнительные услуги, используя данные, содержащиеся в этих заголовочных блоках. В примере 1, данные о пассажире помещены в заголовочный блок для того, чтобы проиллюстрировать их использование SOAP-посредником для выполнения некоторой дополнительной обработки.

SOAP-Simple Objects Access Protocol

- Диалоговый обмен сообщениями
- Продолжая рассмотрение сценария бронирования путешествия, пример 2 демонстрирует SOAP-сообщение, полученное от сервиса продажи билетов в ответ на запрос о бронировании, реализованный сообщением примера 1. Этот отклик выделяет из запроса некоторую информацию, а именно выбор аэропорта в городе отправления.

SOAP-Simple Objects Access Protocol

- <?xml version='1.0' ?>
- <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
- <env:Header>
- <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
- env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
- env:mustUnderstand="true">
-

SOAP-Simple Objects Access Protocol

- <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference> <m:dateAndTime>2001-11-29T13:35:00.000-05:00</m:dateAndTime>
- </m:reservation>
- <n:passenger xmlns:n="http://mycompany.example.com/employees" env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
- env:mustUnderstand="true">

SOAP-Simple Objects Access Protocol

- <n:name>Eke Jygvan Iyvind</n:name> </n:passenger>
- </env:Header>
- <env:Body>
- <p:itineraryClarification xmlns:p="http://travelcompany.example.org/reservation/travel">
- <p:departure>
- <p:departing>

SOAP-Simple Objects Access Protocol

- <p:airportChoices> JFK LGA EWR </p:airportChoices>
- </p:departing>
- </p:departure>
- <p:return>
- <p:arriving>
- <p:airportChoices> JFK LGA EWR </p:airportChoices>

SOAP-Simple Objects Access Protocol

- </p:arriving>
- </p:return>
- </p:itineraryClarification>
- </env:Body>
- </env:Envelope>

SOAP-Simple Objects Access Protocol

- Заголовочные блоки reservation и passenger должны обрабатываться следующим SOAP-посредником, встреченным на пути сообщения, либо, в случае отсутствия узла-посредника, конечным получателем сообщения. На тот факт, что сообщение адресовано следующему SOAP-узлу, встреченному на пути сообщения, указывает присутствие атрибута env:role со значением "http://www.w3.org/2003/05/soap-envelope/role/next" (здесь и далее просто "next"). Этот атрибут реализует роль, исполнять которую обязаны все SOAP-узлы. Присутствие же атрибута env:mustUnderstand со значением "true" указывает на то, что узел (узлы), обрабатывающий заголовочные блоки, должен обрабатывать их в строгом соответствии с их спецификациями либо не обрабатывать SOAP-сообщение вовсе и выдать сообщение об ошибке.

SOAP-Simple Objects Access Protocol

- Решение по поводу того, какие данные поместить в заголовочный блок и тело SOAP-сообщения, должно быть принято на этапе проектирования приложения. Необходимо помнить, что заголовочные блоки могут быть адресованы различным узлам, которые могут встречаться на пути сообщения от отправителя к конечному получателю. Подобные промежуточные SOAP-узлы могут предоставлять дополнительные услуги, используя данные, содержащиеся в этих заголовочных блоках. В примере 1, данные о пассажире помещены в заголовочный блок для того, чтобы проиллюстрировать их использование SOAP-посредником для выполнения некоторой дополнительной обработки.

SOAP-Simple Objects Access Protocol

- Диалоговый обмен сообщениями
- Продолжая рассмотрение сценария бронирования путешествия, пример 2 демонстрирует SOAP-сообщение, полученное от сервиса продажи билетов в ответ на запрос о бронировании, реализованный сообщением примера 1. Этот отклик выделяет из запроса некоторую информацию, а именно выбор аэропорта в городе отправления.

SOAP-Simple Objects Access Protocol

- <?xml version='1.0' ?>
- <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
- <env:Header>
- <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
- env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
- env:mustUnderstand="true">
-

SOAP-Simple Objects Access Protocol

- <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference> <m:dateAndTime>2001-11-29T13:35:00.000-05:00</m:dateAndTime>
- </m:reservation>
- <n:passenger xmlns:n="http://mycompany.example.com/employees" env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
- env:mustUnderstand="true">

SOAP-Simple Objects Access Protocol

- <n:name>Eke Jygvan Iyvind</n:name> </n:passenger>
- </env:Header>
- <env:Body>
- <p:itineraryClarification xmlns:p="http://travelcompany.example.org/reservation/travel">
- <p:departure>
- <p:departing>

SOAP-Simple Objects Access Protocol

- <p:airportChoices> JFK LGA EWR </p:airportChoices>
- </p:departing>
- </p:departure>
- <p:return>
- <p:arriving>
- <p:airportChoices> JFK LGA EWR </p:airportChoices>

SOAP-Simple Objects Access Protocol

- </p:arriving>
- </p:return>
- </p:itineraryClarification>
- </env:Body>
- </env:Envelope>

SOAP-Simple Objects Access Protocol

- Как было описано ранее, элемент env:Body содержит основной контент сообщения, удовлетворяющий схеме описания в пространстве имен XML http://travelcompany.example.org/reservation/travel, который в этом примере включает в себя список различных вариантов аэропортов. В этом примере в качестве отклика возвращены заголовочные блоки примера 1 (с измененными значениями нескольких субэлементов)

SOAP-Simple Objects Access Protocol

- Более широкий набор сценариев использования, которые укладываются в определение шаблона обмена сообщениями типа "запрос-отклик", может быть реализован просто с помощью XML-контента в SOAP-сообщениях при реализации двунаправленного "диалога", где семантика определяется отправляющим и получающим приложениями

SOAP-Simple Objects Access Protocol

- **Вызовы удаленных процедур**
- Одна из целей SOAP Версия 1.2 - инкапсулировать функциональность, реализуемую посредством вызовов удаленных процедур, используя широкие возможности применения и гибкость XML.

SOAP-Simple Objects Access Protocol

- Для вызова SOAP RPC требуется следующая информация:
- 1. Адрес SOAP-узла места назначения;
- 2. Имя процедуры либо метода;
- 3. Наименования и значения всех аргументов, передаваемых процедуре или методу вместе с выходными параметрами и возвращаемым значением;

SOAP-Simple Objects Access Protocol

4. Четкое разделение аргументов, используемых для идентификации web-ресурса, являющегося действительным местом назначения RPC, от аргументов, содержащих данные и контрольную информацию, используемых для обработки вызова ресурсом места назначения RPC

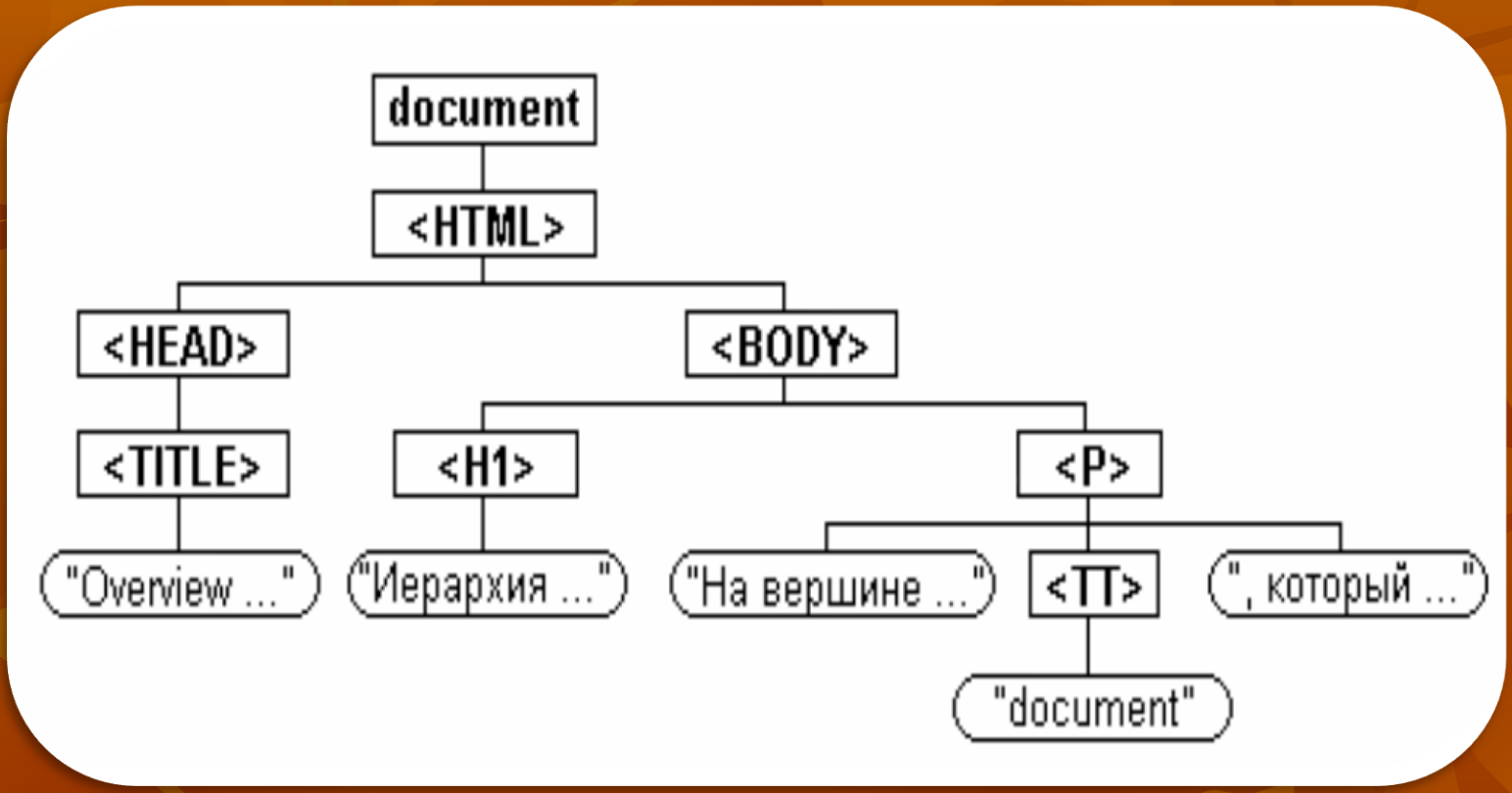
- 5. Определение шаблона обмена сообщениями, а также так называемого "Web-метода", которые будут использоваться для передачи RPC.
- 6. Данные, которые могут быть переданы как часть заголовочных блоков SOAP. Эти данные не являются обязательными

Пример HTML документа

Рассмотрим пример небольшого фрагмента HTML документа:

<HTML>
<HEAD>
<TITLE>
Overview of the DOM
</TITLE>
</HEAD>
<BODY>
<H1>
Иерархия узлов
</H1>
<P>
На вершине иерархии находится узел
<TT>
Document
</TT>
, который представляет в DOM сам документ.
</P>
</BODY>
</HTML>

Структура документа



Структура документа
Продолжение

Элементы, имеющие открывающий и закрывающий тэги (элементы-контейнеры), могут иметь потомков. Текст, атрибуты и элементы типа IMG, не имеющие закрывающего тэга, иметь потомков не могут. Также узлы типа атрибут могут быть потомками только узлов типа элемент. В HTML разные элементы, например, P или IMG, имеют разный набор допустимых для них атрибутов. В DOM узлы-элементы разного вида также имеют разный набор допустимых для них узлов-атрибутов. За редким исключением они соответствуют HTML4. Списки рекомендованных атрибутов для различных элементов можно найти в документации W3C.

Узлы разного типа (document, элементы, атрибуты и текст) имеют свой набор свойств и методов, которые позволяют через сценарии манипулировать ими. Ряд узлов-элементов (объектов) имеют свои собственные свойства и методы. Так, например, объект элемента TABLE имеет метод createTHead.

Пример документа

При рассмотрении свойств и методов узлов будем использовать следующий фрагмент документа:

```
<DIV>
<UL ID="components">
<LI>
    HTML
</LI>
<LI>
    CSS
</LI>
<LI>
    Javascript
</LI>
</UL>
</DIV>
```

В DOM этому фрагменту соответствует ветка дерева, растущая из узла <DIV> к узлу <UL>. Здесь она разветвляется на три веточки по числу узлов <LI> (узлы-атрибуты не принято включать в состав дерева). Каждый из этих узлов имеет
по одному побегу, который заканчивается текстовым узлом.

Навигация по дереву документа

Навигацию по дереву документа можно начинать с любого узла-элемента, для которого мы знаем идентификатор, присваиваемый ему в качестве значения атрибута ID. Ссылку на такой узел можно получить с помощью метода getElementById() объекта document. Параметром метода является идентификатор. Следующая строка кода присваивает переменной oList ссылку на наш список:

```
var oList =
    document.getElementById("components")
```

Навигация по дереву документа

Стартуя с некоторого узла, мы можем бродить по дереву в любом направлении, используя ряд свойств узлов. Так, узлы-элементы и текстовые узлы имеют свойство parentNode, которое возвращает ссылку на родительский узел.

Возьмем для примера узел (объект) oList. Ссылку на родительский элемент DIV этого узла можно получить следующий образом:

```
var oParent = oList.parentNode
```

Навигация по дереву документа

Узлы-элементы и текстовые узлы, являющиеся дочерними некоторого узла, входят в состав коллекции childNodes этого узла.

Узлы-атрибуты составляют отдельную коллекцию attributes. К каждому из них можно обращаться по индексу массива. Например, строка кода

var oItem1 = oList.childNodes[1]

присваивает переменной oItem1 ссылку на элемент <LI> CSS </LI> нашего списка. Именно этот элемент представлен в DOM как узел childNodes[1] узла oList.

Навигация по дереву документа

Первый (childNodes[0]) и последний элементы коллекции имеют специальные имена: firstChild и lastChild. Эти имена являются свойствами родительского узла.

Каждый из элементов коллекции имеет свойства previousSibling и

nextSibling. Эти свойства хранят ссылку на ближайших братков элемента -

предыдущий и последующий элементы коллекции (возвращают null, когда братков нет). Так, элемент childNodes[1] является свойством

nextSibling элемента childNodes[0] и свойством previousSibling элемента

childNodes[2].

Навигация по дереву документа

Используя эти свойства, мы можем получить ссылку на узел childNodes[1] любым из следующих способов:

oList

.firstChild.nextSiblingoList

.childNodes[2].previousSibling

Ссылка на более удаленные узлы как по горизонтали, так и по вертикали дерева формируется путем слияния ссылок на ближайших родственников по стандартным правилам объектно-ориентированного программирования.

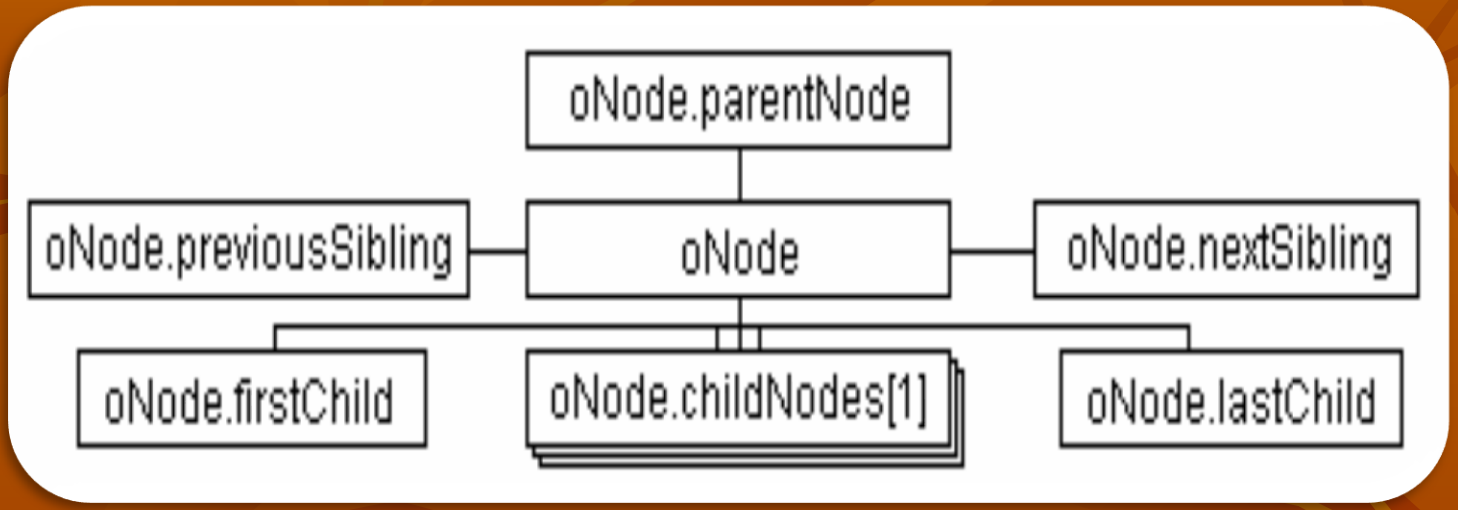
Т ак,

oList.childNodes[1].firstChild является ссылкой на текст"CSS"элемента

<LI> CSS </LI>

нашего списка.

Схема узлов документа



Все описанные выше свойства узлов (parentNode, firstChild, lastChild, nextSibling и previousSibling), необходимые для навигации по дереву документа, являются свойствами только для чтения.

Свойства-характеристики узлов

Свойство узла nodeType (только для чтения) возвращает 1, 2 или 3 для узлов, соответствующих тэгу, атрибуту или тексту, соответственно. Свойство nodeName (только для чтения) возвращает имя HTML тэга, которому соответствует данный узел, например, P для параграфа или UL для ненумерованного списка.

Для узлов-атрибутов nodeName возвращает название атрибута, а для тестовых узлов возвращает #text.

Текстовые узлы имеют еще одно очень важное свойство: nodeValue. Это свойство для чтения и записи хранит содержание текстового узла. Для элементов оно возвращает null, а для атрибутов - значение атрибута.

Свойства-характеристики узлов

Свойство узла nodeType (только для чтения) возвращает 1, 2 или 3 для узлов, соответствующих тэгу, атрибуту или тексту, соответственно. Свойство nodeName (только для чтения) возвращает имя HTML тэга, которому соответствует данный узел, например, P для параграфа или UL для ненумерованного списка.

Для узлов-атрибутов nodeName возвращает название атрибута, а для тестовых узлов возвращает #text.

Текстовые узлы имеют еще одно очень важное свойство: nodeValue. Это свойство для чтения и записи хранит содержание текстового узла. Для элементов оно возвращает null, а для атрибутов - значение атрибута.

Создание новых узлов

Метод createElement()

Метод createTextNode()

Создание новых узлов

Продолжение

Технику создания новых элементов обсудим на конкретном примере: мы хотим добавить к нашему списку элемент

```
<LI>
    XML
</LI>
```

Этому HTML элементу в DOM соответствуют два узла: узел-элемент <LI> и текстовый узел "XML". Следовательно, нужно создать два новых узла. Новые узлы создаются с помощью методов createElement() и createTextNode() объекта document.

Создание новых узлов

Метод createElement()

Метод createElement() принимает в качестве параметра строку с названием тэга элемента и создает новый HTML элемент указанного типа. Например, строка кода

```
var oItem = document.createElement("LI")
```

создает новый элемент списка <LI>, у которого нет содержания. Метод возвращает ссылку на созданный им объект. Созданный выше элемент <LI> находится в памяти, но не входит в состав текущего документа. Для того, чтобы он стал частью документа, его надо добавить к существующим узлам документа с помощью методов insertBefore() или appendChild( ).

Создание новых узлов

Метод createTextNode()

Метод createTextNode() используется для создания текстового узла. Он принимает в качестве параметра строку текста, которая задает значение свойства nodeValue текстового узла. Например, строка кода

```
var oText = document.createTextNode("XML")
```

создает новый текстовый узел "XML". Метод возвращает ссылку на созданный им объект. Созданный текст еще не входит в состав текущего документа. Для того, чтобы он стал частью документа, его надо присоединить к существующим узлам документа с помощью методов appendChild(), replaceNode() или insertBefore().

Создание новых узлов
Метод createTextNode()

Если имеется ссылка на ненужный текстовый узел (например, oText), то можно не создавать нового узла, а воспользоваться уже существующим. Для этого надо просто присвоить новый текст в качестве значения свойства nodeValue существующего узла:

Text.nodeValue= "XML"

Итак, мы создали два новых узла: узел-элемент <LI> и текстовый узел "XML".

Редактирование дерева документа

DOM, как редактор, позволяет копировать, вставлять, замещать и удалять как отдельные узлы, так и целые ветви дерева документа.

Вспомним, что у нас есть два узла: узел-элемент <LI> и текстовый узел "XML". Оба узла находятся в памяти и мы хотим встроить их в текущий документ. Прежде всего, надо задать текст "XML" в качестве содержания элемента <LI>. Сделать это можно с помощью метода appendChild().

Редактирование дерева документа

Методы

Вставка: методы appendChild() и insertBefore()

Копирование: метод cloneNode()

Замещение: методы replaceChild()и replaceNode()*

Удаление: методы removeChild()и removeNode()*

Перестановка: метод swapNode()*

Дополнительные методы: applyElement()* и hasChildNodes()

Примечание. Методы, помеченные звездочкой*, не входят в список рекомендуемых W3C.

Редактирование дерева документа

Вставка: метод appendChild()

Данный метод добавляет элемент в конец коллекции childNodes узла, который его активизировал. Сам этот узел становится родительским узлом для узла, ссылку на который метод принимает в качестве параметра. Например, строка кода

oItem.appendChild(oText)

добавляет узел oText к узлу oItem. Отметим, что метод возвращает ссылку на объект, который он добавляет. В нашем случае это объект oItem.firstChild. Теперь мы имеет в памяти элемент (веточку дерева из двух узлов)

<LI> XML </LI>

Вставим эту веточку в текущий документ. Если мы хотим вставить ее в самый конец нашего списка, то надо, как и выше, использовать методappendChild():

oList.appendChild(oItem)

Редактирование дерева документа

Вставка: метод appendChild()

Поскольку узел oList, к которому мы присоединили узел oItem, является частью текущего документа, созданный нами элемент списка также становится частью документа. Теперь наш список выглядит так:

<UL ID="components">

<LI> HTML </LI>

<LI> CSS </LI>

<LI> Javascript </LI>

<LI> XML </LI>

</UL>

Вставить созданный нами элемент списка не в конец, а, скажем, после элемента списка<LI> HTML </LI> можно сделать с помощью метода insertBefore().

Редактирование дерева документа

Вставка: метод insertBefore()

В отличие от метода appendChild(), метод insertBefore() позволяет указать, в какое место коллекции childNodes будущего родительского узла будет вставлен новый узел. Как следует из названия, метод требует ссылки на узел, перед которым будет вставлен его новый браток. Мы создадим эту ссылку в отдельной строке, хотя это и необязательно. Итак, код

Var oBrother = oList.firstChild.nextSiblingoList.insertBefore(oItem, oBrother)

добавляет в коллекцию childNodes узла oList узел oItem сразу после узла childNodes[0]. В качестве первого параметра метод insertBefore() принимает ссылку на узел, который мы хотим добавить, а в качестве второго параметра – ссылку на узел, перед которым будет вставлен новый браток. Второй параметр метода является необязательным.

Редактирование дерева документа
Вставка: метод insertBefore()

Если родительский узел не имеет деток, то задавать его не следует. Если родительский узел имеет деток, а второй параметр не задан, то добавляемый узел становится самым последним среди деток родительского объекта. Метод insertBefore()возвращает ссылку на вставленный в документ объект.

Теперь первоначальный список выглядит так:

```
<UL ID="components">
<LI>HTML</LI>
<LI>XML</LI>
<LI>CSS</LI>
<LI>Javascript</LI>
</UL>
```

Редактирование дерева документа

Копирование: метод cloneNode()

Если необходимо скопировать некоторый узел вместе со всеми его атрибутами, то надо воспользоваться методом cloneNode(). В качестве параметра метод принимает выражение типа Boolean. Если оно равно false, то копируется только тот узел, который активизирует метод. Если параметр метода равен true, то копируется узел вместе со всеми его потомками. Например, строка кода

```
var oClone = oList.cloneNode(true)
```

копирует в память всю ветвь дерева, начинающуюся на узле oList, то есть весь наш список целиком. Метод возвращает ссылку на копию узла. Используя эту ссылку, мы можем в дальнейшем работать с этой копией, например, отредактировать ее и вставить в документ.

Редактирование дерева документа

Замещение: метод replaceChild()

Метод replaceChild() позволяет у узла, который его активизирует, заменить одного из его деток на нового. Ссылку на новый и на заменяемый узлы метод принимает в качестве первого и второго параметров, соответственно. Так, следующий фрагмент сценария

```
var oItem=document.createElement("LI")oItem.appendChild(document.createTextNode("JScript"))
oList.replaceChild(oItem, oList.lastChild)
```

создает сначала элемент списка с текстом "JScript", а затем заменяет им последний элемент нашего списка. Отметим, что метод возвращает ссылку на вставленный в документ узел.

Теперь список выглядит так:

```
<UL ID="components">
<LI>HTML</LI>
<LI>CSS</LI>
<LI>JScript</LI>
</UL>
```

Описанный выше пример надо рассматривать только как иллюстративный, поскольку тот же результат можно получить гораздо проще:

```
oList.lastChild.firstChild.nodeValue= "JScript"
```

Редактирование дерева документа

Замещение: метод replaceNode()

Если нужно заменить некоторый узел в документе другим узлом, то можно воспользоваться методом replaceNode(). Метод replaceNode() удаляет из документа узел, который его активизирует, и вставляет в документ вместо него новый узел, ссылку на который он принимает в качестве параметра. Заметим, что узел удаляется вместе со всеми своими атрибутами и потомками. Так, следующий фрагмент сценария

```
var oParagraph = document.createElement("P")
var oText = document.createTextNode("Составные части DHTML")
oParagraph.appendChild(oText)
oList.replaceNode(oParagraph)
```

создает сначала параграф с текстом "Составные части DHTML", а затем заменяет наш список oList на этот параграф. Отметим, что метод возвращает ссылку на вставленный в документ узел.

Метод replaceNode() не входит в список рекомендуемых W3C, но поддерживается Internet Explorer 5.

Редактирование дерева документа

Удаление: метод removeChild()

Метод removeChild() позволяет у узла, который его активизирует, удалить одного из его дочерних узлов. Ссылку на удаляемый узел метод принимает в качестве параметра. Например, строка кода

```
var oRemovedItem = oList.removeChild(oList.lastChild)
```

удаляет из нашего списка последний элемент. Метод возвращает ссылку на удаляемый им узел. Поскольку удаленный из документа узел остается в памяти, мы можем в дальнейшем работать с ним, используя эту ссылку. Например, пристроить в какой-нибудь другой список.

Редактирование дерева документа

Удаление: метод removeNode()

Если нужно удалить некоторый узел из документа, то можно воспользоваться методом removeNode(). В качестве параметра метод принимает выражение типа Boolean. Если оно равно false, то удаляется только тот узел, который активизировал метод. При этом, идущая от него ветвь дерева присоединяется к его родительскому узлу. Если параметр метода равен true, то узел удаляется вместе со всеми своими потомками. Например, строка кода

```
var oRemovedList = oList.removeNode(true)
```

удаляет из документа весь наш список целиком. Метод removeNode() возвращает ссылку на удаляемый им узел. Поскольку удаленный из документа узел остается в памяти, мы можем в дальнейшем работать с ним, используя эту ссылку.

Использовать false в качестве параметра надо осмысленно. Так, если мы применим метод removeNode() с параметром false к нашему списку, то его детки должны будут перейти к элементу DIV. Но, согласно требованиям HTML 4, элементы списка LI не могут размещаться в этом контейнере!

Подчеркнем, что этот метод не входит в список рекомендуемых W3C, но поддерживается Internet Explorer 5.

Редактирование дерева документа

Перестановка: метод swapNode()

Если необходимо переставить два узла в документе (в общем случае, две ветви дерева), то используется метод swapNode(). Данный метод меняет местами узел, который его активизировал, и узел, ссылку на который он принимает в качестве параметра. Например, строка кода

```
var oSwappedNode = oFirst.swapNode(oSecond)
```

переставляет узлы oFirst и oSecond и возвращает ссылку на узел, который активизирует метод.

Замечание: метод swapNode() не входит в список рекомендуемых W3C, но поддерживается Internet Explorer 5.

Редактирование дерева документа

Метод applyElement()

Описанные выше методы appendChild() и insertBefore() позволяют узлам заводить деток. Internet Explorer 5 поддерживает также метод, который позволяет узлу-элементу обзаводиться родителем. Таким методом является метод applyElement(), который принимает в качестве параметра ссылку на будущего родителя. Например, строка кода

var oParent = oChildNode.applyElement(oParentNode)

задает узел oParentNode в качестве родительского для узла oChildNode. При этом узел oParentNode лишается (если они у него были) как своего родителя, так и своих деток перед тем, как "заполучить" нового наследника. Метод возвращает ссылку на нового родителя.

Замечание: метод applyElement() не применим к текстовым узлам.

Редактирование дерева документа

Метод hasChildNodes()

Метод hasChildNodes() позволяет узнать есть или нет у узла потомки. Например, строка кода

oTestedNode.hasChildNodes()

возвращает true, когда узел oTestedNode имеет потомков, и false в противном случае.

Методы работы с атрибутами элементов

Для работы с атрибутами элементов используются

Метод createAttribute()

Метод setAttribute()

Метод removeAttribute()

Метод getAttribute()

Работа с атрибутами элемента

Продолжение

Все атрибуты узла-элемента (за исключением атрибута STYLE) составляют коллекцию attributes. W3C определяет эту коллекцию как массив с доступом по именам элементов. Например,

oNode.attributes.align

или

oNode.attributes["align"]

возвращает значение атрибута ALIGN узла oNode. (В документации Microsoft по DHTML сказано, что к элементам коллекции нужно обращаться по индексу массива. Но на практике работает описанное выше обращение по именам. Вероятно, документация обновляется реже, чем появляются новые версии броузера.)

Коммуникационные технологии

Раздел 6

Управление знаниями