Информатика

Summary

В этом семестре…

- Web-приложения
  - HTTP
- Сетевые приложения
  - TCP/UDP

Интернет и Web

- **Интернет** – ***глобальная сетевая инфраструктура***, позволяющая компьютерам общаться друг с другом используя ***различные протоколы***
- **Web** – система связанных ссылками документов на различных компьютерах сети Интернет
  - а как же динамические Web-приложения?

Модель OSI

	Уровень (layer)	Тип данных (PDU)	Функции	Примеры
Host layers	7. Прикладной (application)		Доступ к сетевым службам	HTTP, FTP, SMTP, RDP, SNMP
	6. Представления (presentation)	Данные	Представление и шифрование данных	ASCII, EBCDIC, JPEG
	5. Сеансовый (session)		Управление сеансом связи	RPC, PAP
	4. Транспортный (transport)	Сегменты (segment)/ Дейтаграммы (datagram)	Прямая связь между конечными пунктами и надёжность	TCP, UDP, SCTP, PORTS
	3. Сетевой (network)	Пакеты (packet)	Определение маршрута и логическая адресация	IPv4, IPv6, IPsec, AppleTalk
Media layers	2. Канальный (data link)	Биты (bit)/ Кадры (frame)	Физическая адресация	PPP, IEEE 802.22, Ethernet, DSL, ARP, L2TP, Network Cards
	1. Физический (physical)	Биты (bit)	Работа со средой передачи, сигналами и двоичными данными	USB, витая пара, коаксиальный кабель, оптический кабель

Web-приложения

- Программные системы
  - программы VS приложения
- Работают с данными по протоколу HTTP
  - какие данные?
- Работают на основе событийной модели
  - что является событиями? примеры?

Клиент-серверная архитектура

- **Клиент** – узел сети, потребляющий данные
  - Запрашивает и получает данные
- **Сервер** – узел сети, обрабатывающий данные, обслуживающий клиентов
  - Обрабатывает запросы, генерирует ответы
  - Принимает и отдает данные

Общение клиент-сервер в web

- Работа с URL, URI
- Определение адреса с помощью DNS
- Обмениваются данными по HTTP
  - В основе - TCP/IP
    Маршрутизация + гарантия доставки
  - Работа с потоками данных

Структура http-сообщений

- **Стартовая строка**
  метод, URI, версия
- **Заголовки**
  общие, запроса, ответа, сущности
- **Тело**
  данные сообщения
  отделяются пустой строкой

```
GET /wiki/java HTTP/1.1

Host: ru.wikipedia.org

User-Agent: Mozilla/5.0 (X11; U;
Linux i686; ru; rv:1.9b5)
Gecko/2008050509 Firefox/3.0b5

Accept: text/html

Connection: close
```

Методы

- **OPTIONS** – определение возможностей сервера и параметров соединения. В ответе Allow со списком методов
- **GET** – получение ресурса, начало процесса. В ответе ресурс или данные о ходе процесса. Параметры запроса в URI. Идемпотентный
- **POST** – передача данных. В ответе код результата, заголовок Location. Параметры передаются в теле запроса. Не идемпотентный
- **PUT, PATCH** – загрузка ресурса (или его части) по соответствующему URI. В ответе код результата.
- **DELETE** – удаление ресурса.

Структура ответа

- **Стартовая строка**
  версия, код ответа
- **Заголовки**
- **Тело**

```
HTTP/1.1 200 OK
Date: Wed, 11 Feb 2009 11:20:59 GMT
Server: Apache
X-Powered-By: PHP/5.2.4-2ubuntu5wm1
Last-Modified: Wed, 11 Feb 2009 11:20:59
GMT
Content-Language: ru
Content-Type: text/html; charset=utf-8
Content-Length: 1234
Connection: close

(далее следует запрошенная страница в
HTML)
```

Коды ответа

- **1xx** – информационный (информация к сведению, отвечать не нужно)
- **2xx** – Успех (Success)
- **3xx** – Перенаправление (Redirection)
- **4xx** – Ошибка клиента
- **5xx** – Ошибка сервера

Формы

- Основной источник http-запросов
- Объединяют несколько параметров запроса
  - Input нужного типа
  - У всех input’ов есть имя (name)
  - Управляют передачей данных в запрос
    - В строку запроса в случае GET
    - В тело запроса в случае POST
- Связаны с действием по URI
- Определяют метод запроса

Пример multipart

POST /send-message.html HTTP/1.1
Host: mail.example.com
Referer: http://mail.example.com/send-message.html
User-Agent: BrowserForDummies/4.67b
Content-Type: multipart/form-data; boundary="Asrf456BGe4h"
Content-Length: (суммарный объём)
Connection: keep-alive
Keep-Alive: 300
(пустая строка)
(отсутствующая преамбула)

--Asrf456BGe4h
Content-Disposition: form-data; name="DestAddress"
(пустая строка)
brutal-vasya@example.com
--Asrf456BGe4h
Content-Disposition: form-data; name="AttachedFile1"; filename="horror-photo-1.jpg"
Content-Type: image/jpeg
(пустая строка)
(двоичное содержимое первой фотографии)

Content-Type, Accept, MIME

- Accept – что принимает клиент
- Content-Type – что передаётся в теле сообщения
  - text/html
  - text/css
  - text/plain
  - text/xml
  - application/json
  - application/javascript
  - image/jpeg
  - …

Multipurpose Internet Mail Extensions
application
audio
example
image
message
model
multipart
text
video

Форматы данных

- XML (eXtensible Markup Language)
  - Произвольные теги, строгий
- HTML (HyperText Markup Language)
  - Фиксированные теги, нестрогий
- JSON (JavaScript Object Notation)
  - Объект, массив, строка, значение

Static vs Dynamic Web

- **Статические сайты**
  - Связь URI с файловой системой
  - Одни файлы для всех
  - Только GET запросы
- **Динамические сайты**
  - Содержимое ответа генерируется при обработке запроса
  - Ответ зависит от клиента

Http-сервер

- Основа динамических веб-приложений
- Постоянно работающее приложение
- Получает и обрабатывает запросы
- Исполняет код для генерации ответа
- Отправляет содержимое клиенту

Web-сервер, сервер приложений

- За ними «прячутся» web-приложения
- Берут на себя обработку http запросов/ответов, перенаправление их приложениям
- Сложные задачи присущи серверам приложений
  - Бесперебойность, распределение нагрузки, кластеризация

Типовые схемы работы

- Приложение разворачивается на web-сервере
  - IIS, Apache, Nginx
- Приложение разворачивается самостоятельно
  - Приложение самостоятельно
    - слушает порт,
    - обрабатывает запросы,
    - генерирует ответы
    - отправляет ответы клиентам

HttpListener

- Отвечает за прослушку подключений по HTTP
- Асинхронно ожидаем подключения (HttpListenerContext)
- Получаем доступ к запросу и ответу (HttpListenerRequest, HttpListenerResponse)

Request, Response

- Содержат всё необходимое для формирования http сообщений
- Url, Headers, Queries, Input/Output Streams

Маршрутизация запросов

- Одна из основных задач при разработке web-приложений
- Нужно сопоставить URI с обработчиком (программным модулем) *контроллером*
- Подходы к маршрутизации? Middleware, рефлексия

Данные приложения

- Данные web-приложения обычно хранятся в базах данных
  - Почему? Что такое БД?
- Наиболее популярны реляционные БД (табличные)
- Самый популярный подход к работе с БД – объектно-реляционные отображения

ORM

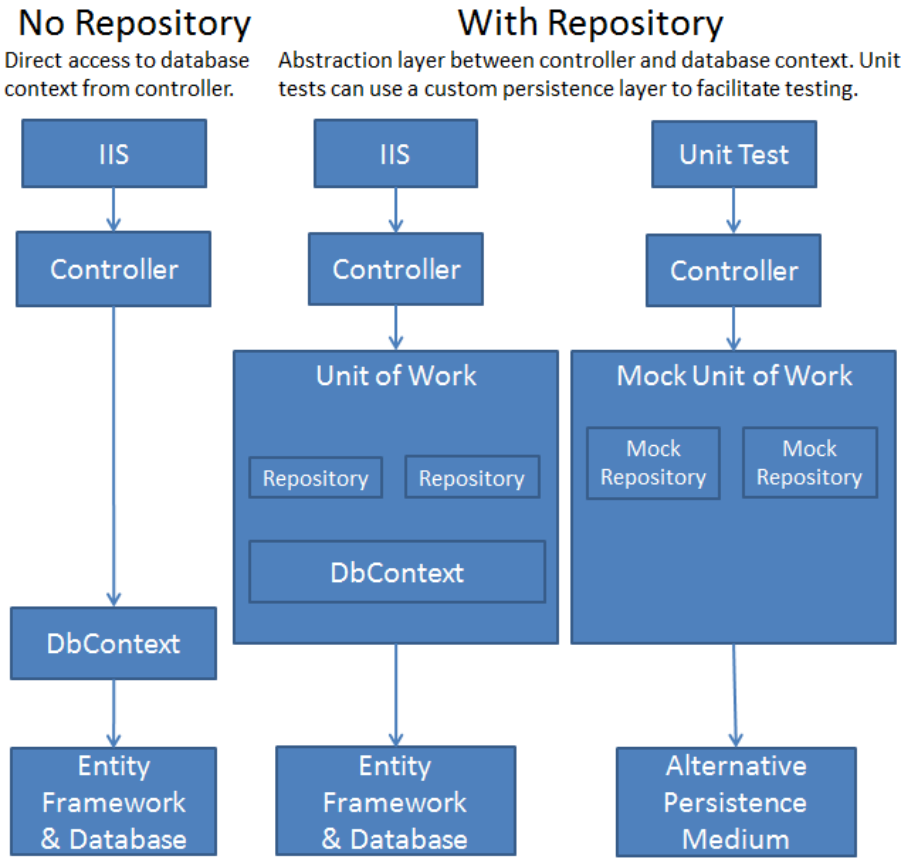
- Сопоставление
  - Объявлений классов и объявлений таблицы (схемы)
  - свойств и столбцов
  - методов и хранимых процедур
  - коллекций объектов и таблиц
  - объектов и строк таблицы

Entity Framework

- Entity Data Model (Классы сущностей, таблицы, отношения и типы данных, отображение)
- Классы сущностей (TEntity)
- DbContext
- DbModelBuilder
- DbSet<TEntity>
- Связи 1-1, 1-Many, Many-Many
- DB First, Code First
- Миграции

Repository, UnitOfWork

- Уровень абстракции
- Независимость от ORM
- Модульность и независимость компонентов



Repository, UnitOfWork

- Репозитории для доступа к сущностям
- GenericRepository для универсальности
- UnitOfWork – работает с репозиториями, управляет транзакциями в БД, обеспечивает целостность
  - dbContext в EF – это UnitOfWork + GenericRepository
- Подключение UnitOfWork через внедрение зависимостей
(IoC-контейнеры Autofac, Ninject, Unity и т.д.)

Модель

- Данные и способы обработки
- Модель предметной области
  - сколь угодно детализированная и сложная
- Модель для использования в web-приложении
  - фасад для модели предметной области

HTTP-Stateless

- Каждый запрос должен обрабатываться отдельно от других, не завися от предыдущих
- Можно с этим бороться, используя
  - ViewState
  - Скрытые поля
  - Сессии + Cookie

Сессии + Cookies

- Сессии (сеансы) – данные, связанные с клиентом, сохраняемые пока пользователь работает с web-приложением
  - Локально, в отдельной службе, в базе данных
- Идентификатор сессии записывается в Cookie
  пара строк ключ-значение + expires + path
- Cookie передаётся с каждым запросом (в заголовках, можно в URI)
- Контролируем подключения клиентов с помощью сессий (выход со всех устройств – закрытие сессий)

View

- Классика web-приложений – генерация HTML
- HTML – GUI
- Javascript – браузерный язык, логика на клиенте, обработка событий клиента, изменение содержимого HTML

Шаблонизаторы

- Часто приходится генерировать HTML по заданному шаблону
- Некоторые части общие для всех страниц приложения (шапка, подвал)
- Есть специальные инструменты-шаблонизаторы
  - Пример - Razor

AJAX

- Asynchronous Javascript and XML
- Фоновый обмен информацией с сервером + динамическое изменение содержимого страницы
- Плюсы, минусы?
- Можно не только XML. Смотреть jQuery и JSON AJAX запросы

Modern web-dev

- Javascript делает возможным генерацию представлений на клиенте по данным, полученным с сервера в результате AJAX запросов
- См. JS фреймворки (Angular, Vue.JS, React,…)

Архитектура MVC

- Разделение модели от представления
- Модель независима от других частей системы активная и пассивная
- Представление отображает модель активное и пассивное
- Контроллер – знает о модели, изменяет её, управляет представлениями

MVC

- Пользовательский интерфейс – представление и контроллер
- Модель MVC != модель предметной области фасад для доменной модели
- Адаптеры и фасад для клиент-ориентированной архитектуры (Interface Segregation)

Пассивный сервер

- Классический подход
- События генерируются клиентом
- Сервер только обрабатывает запросы
- Сервер не может выступать инициатором действий

Активный сервер

- Polling, Long Polling
  - Опрашиваем с заданной периодичностью или держим открытое соединение
- Server-sent events (HTML5, нет в Edge)
  - EventSourse + onmessage, Accept: text/event-stream, Flush
- WebSockets (двунаправленная связь)
  - HTTP handshake, onopen,onmessage,onerror,onclose, send,close

Что нужно сделать в семестровке

- Веб-приложение по заданной теме
- Работающее с базой данных (желательно ORM + Repository&UnitOfWork)
- Работающее с сессиями и Cookies
- Содержащее публичные и частные страницы разграничение ролей (автор/гость, админ/пользователь)
- Содержащее страницы с поиском/просмотром по категориям
- 4-5 обработчиков (хендлеров/контроллеров)

Не только HTTP

- **TCP**
  - Гарантированная доставка (как в HTTP)
  - Клиент-серверная модель
  - Сокеты или Listener с Client'ом
    - TcpListener, TcpClient
- **UDP**
  - Быстрая передача
  - Широковещательная рассылка
  - Сокеты или UdpClient
- **Http клиенты** (для работы с web-сервисами/API)

Требования к сетевому приложению

- Работа с одним из протоколов TCP или UDP (интереснее – UDP)
- Явно выраженное сетевое взаимодействие
- Через сервер или между клиентами напрямую
- Авторизация пользователей через один из популярных сервисов (API соц. сетей)

Вопросы?

e-mail: marchenko@it.kfu.ru