Client/Server Programming for Internet Applications

TCSS460

Summer 2020

Lecture 10 - Node.js (cont'd)

TACOMA

node package manager (npm)

- **npm (Node Package Manager)** https://www.npmjs.com/
  - is a package manager for **Node.js**
  - allows developers to **create**, **share**, and **reuse modules** in Node.js applications
  - can also be used to share complete Node.js applications
  - **modules** are libraries of code that can be reused in different projects
- typical **examples** of modules include
  - a library for interacting with a **database**
  - a library for validating **input** data
  - a library for parsing YAML Ain't Markup Language (YAML) files
    - human-readable data-serialization language

2

Modules

- after you install npm, you can begin installing modules from the terminal:

```
npm install [module_name]
```

- to use modules in your Node.js applications, you must require them

```
var module = require('module');
```

- where to find modules
  - npmjs.com

```
npm search irc
```


Express

- **Express** is a web framework for Node.js. Web applications
  - you can develop faster and write applications on top of stable, tested code
  - lightweight, can build JSON APIs
  - some other web frameworks that you may be familiar with are
    - Ruby on Rails (Ruby)
    - Sinatra (Ruby)
    - Django (Python)
    - Zend (PHP)
    - CodeIgniter (PHP)

```
npm install -g express
```

Exploring Express

- *recommended* folder structure
  - **app.js**: application file used to start an application (i.e. configuration)
  - **node_modules**: any node modules that have been installed
  - **package.json**: gives information on the application, including dependencies
  - **public**: folder serves the application to the Web
  - **routes**: defines the pages an application should respond to
  - **views**: defines the layouts for the application

HTML Template Engines

- **template engines** are used in most frameworks to generate HTML
  - also known as *template processors* or *filters*
  - output data from an application to HTML
  - examples:
    - Smarty (PHP)
    - Mustache (supports many languages including JavaScript)

json

```
{
    "Name" : "Eyhab",
    "Email": "ealmasri@uw.edu"
}
```

Mustache

```
<p>My name is {{ name }}<br/>
   and my email is {{ email_address }}
</p>
```

https://github.com/janl/mustache.js

Routing in Web Applications

- **routing** describes where and how an application should respond to certain HTTP requests
  - defines the end point for HTTP requests
- How routing works in Express?
  - Express uses HTTP **verbs** to define routes
    - these verbs define the type of request made to the server
    - **GET** → retrieves data from server (retrieve **existing** resources)
    - **POST** → sends data to server (creates **new** resources)
    - others: **PUT**, **DELETE**, **HEAD**, **OPTIONS** and **TRACE**

TCSS 460 - Summer 2020 7

REpresentational State Transfer (REST)

- **Representational State Transfer (REST)** is an architectural style for building web services that access or manipulate web **resources**
  - web services that adhere to REST are called **RESTful web services**
  - a **resource** may be a
    - text file
    - media file (e.g. images, videos, audio)
    - specific row in a database table
    - collection of related data (e.g. products)
    - logical transaction
    - queue
    - downloadable program
    - business process (i.e. procedure)
    - almost anything!

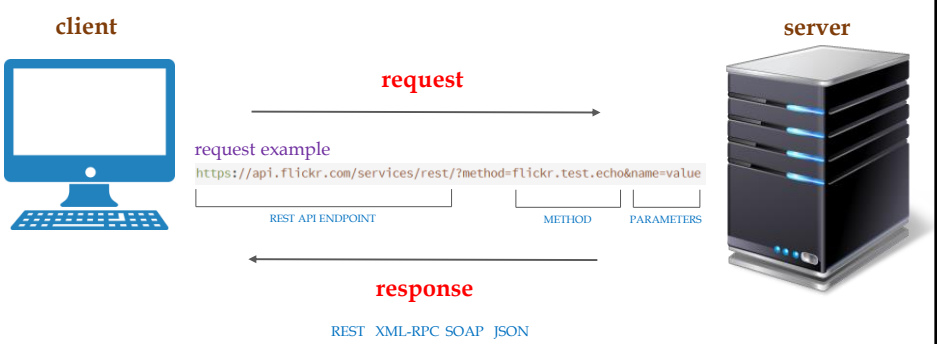
TCSS 460 - Summer 2020 8

REpresentational State Transfer (REST) (cont'd)

- clients manipulate the state of these resources through **representations**
  - example:
    - database table row may be represented as XHTML, XML or JSON
  - **representations** typically capture the current or "intended" **state of** a resource
- a client **receives a representation** from a service is usually **acquiring the most recent state** of the **resource**
- when clients **send representations** to services, their intent is usually to **alter the state** of a **resource**

TCSS 460 - Summer 2020 9

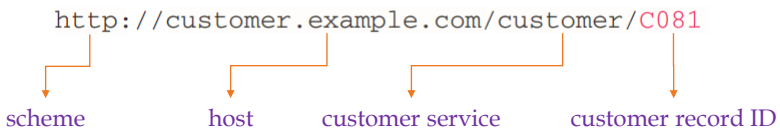
REpresentational State Transfer (REST) (cont'd)



10

Resource Identifiers and REST Services

- a **resource identifier** can be service-specific
  - one REST service can provide many resource identifiers for use by its **service consumers**
- assume we have an entity REST service called **Customer**
  - **service** would be responsible for encapsulating customer-related **resources**
  - a **resource identifier** that enables retrieval of customer data might look something like this



11

REST Methods

- a **method** is a type of function provided by a uniform contract to process **resource identifiers** and **data**
- typical uniform contract will provide a modest set of **high-level** methods
  - these methods are capable of performing basic processing functions
- anything more **specific** that a service consumer requires of a REST service is expressed in the resource identifier
- HTTP message expresses the **GET** statement in **header**

```
GET /customer/C081 HTTP/1.1
```

12

Cloud Computing for Deploying Web Applications

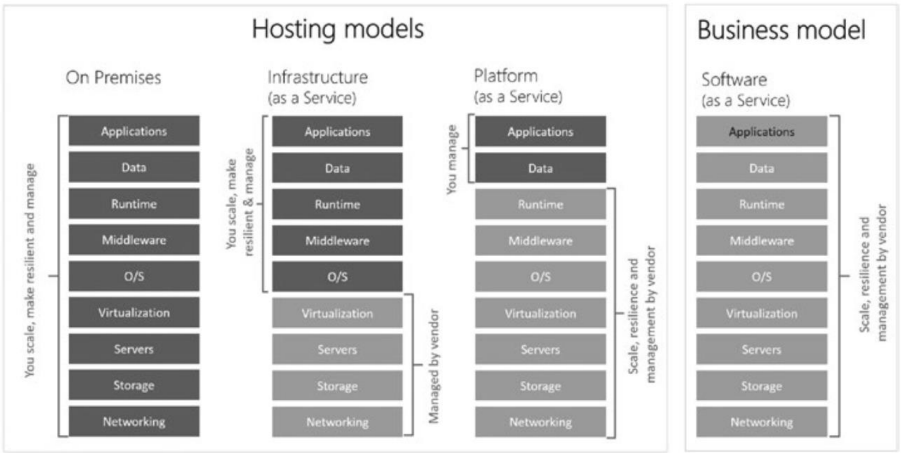
Cloud Computing

- **cloud computing** introduced the concept of a managed virtual environment
  - on-demand provisioning of cloud resources

- terms
  - **Infrastructure-as-a-Service (IaaS)**
  - **Platform-as-a Service (PaaS)**
  - **Software-as-a-Service (SaaS)**

  were introduced to define these choices

Cloud Hosting Models



Bob Familiar, Microservices, IoT, and Azure

15

IaaS, PaaS, SaaS

- **IaaS**: a cloud provider offers the network, virtual machines, and storage on demand
  - users of this model are responsible for all the layers above the VM level including OS configuration and patching
- **PaaS**: a cloud provider offers to maintain the operating system, provide middleware such as databases, enterprise messaging, and runtime containers for application code
  - users of this model can focus on the capabilities of their application and automating deployment
- **SaaS**: a business model where the entire software and hardware stack of a solution is managed by the cloud provider and often offered through a subscription model

16

Cloud Computing and REST APIs

- at a high level, cloud platforms provide infrastructure, storage, compute and applications services on-demand
- this is achieved via **RESTful web services** and a pay-as-you-go model
  - only pay for what you use
- support high-velocity does not require making huge investments in on-premises infrastructure
- the underlying architecture of cloud platforms is called

microservice architecture

17

What is a microservice?

- a **microservice** is a software building block that does a specific granular functionality and does it well
  - can be provisioned on-demand
  - elastically scaled
  - provides fault tolerance and fail over
  - when it is no longer needed, it can be de-provisioned
- all the capabilities provided by commercial cloud platforms are themselves microservices
  - can be automated through scripting languages
- when designing your own software as a service, the
  - recommended platform is the **cloud**
  - recommended architecture is **microservices**

18

Module Topics

- Server Side Development
- Node.js