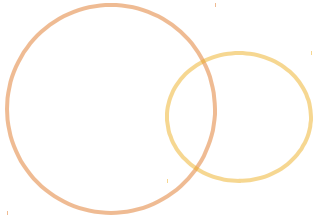
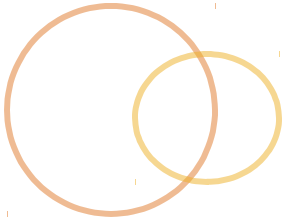
Overview Of REST

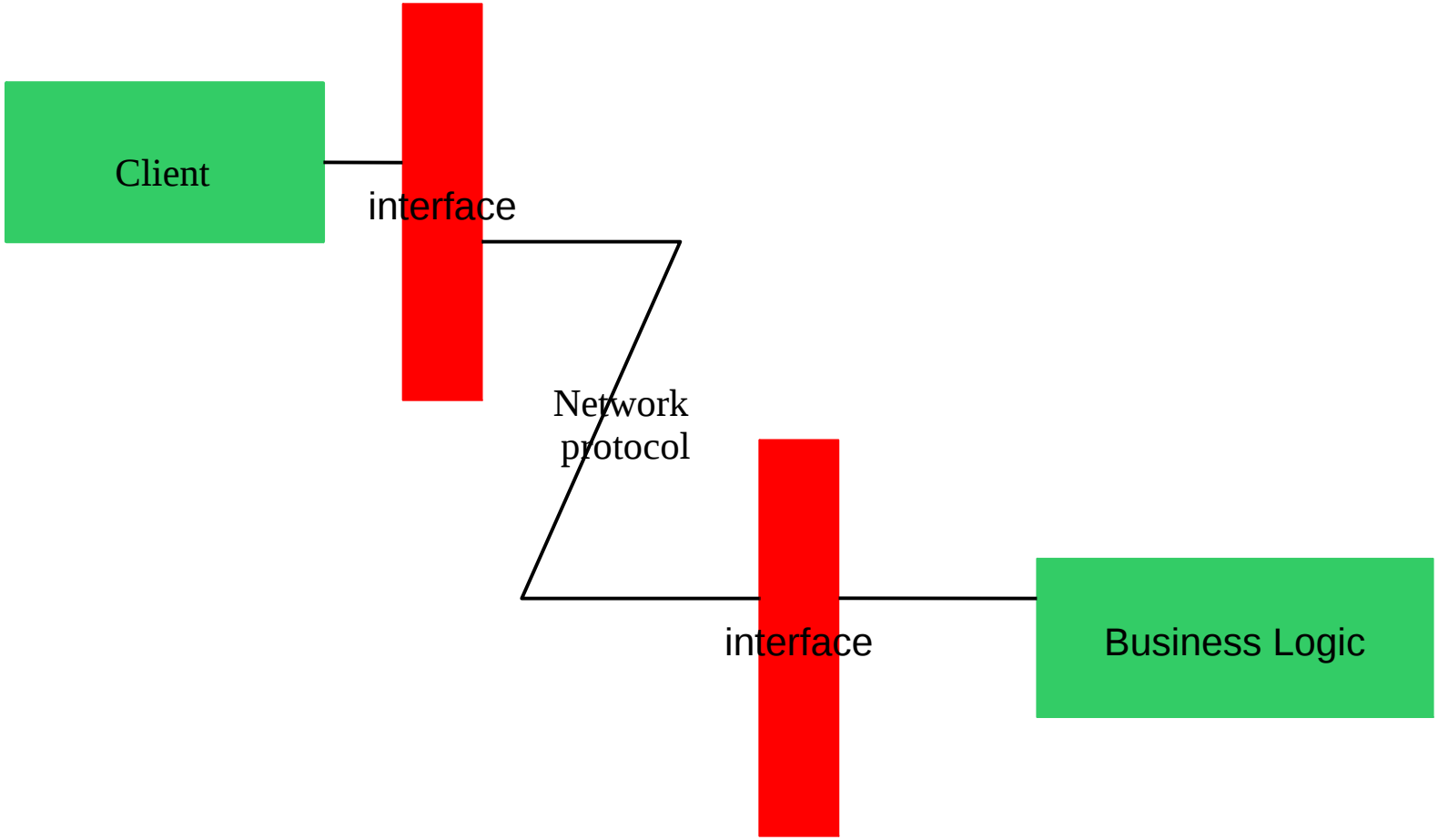




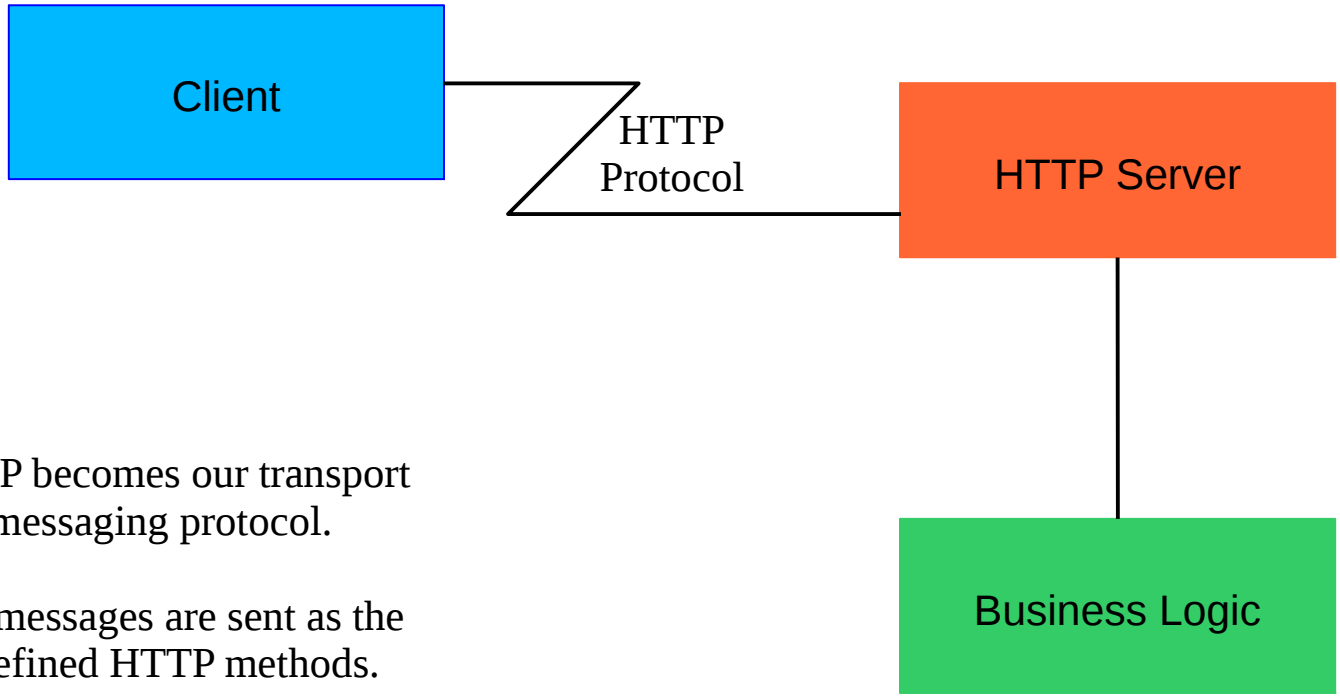
Objectives

- Identify the benefit of REST
- Select appropriate HTTP methods for the implementation of particular behaviors in a REST interface
- Map simple entity relationship data models to REST URIs
- Select appropriate mechanisms for carrying data and request meta data between service and client
- List three features of Intuit's TCRS “starting point project”

Traditional Network Communication



RESTful Network Communication



HTTP becomes our transport and messaging protocol.

The messages are sent as the predefined HTTP methods.

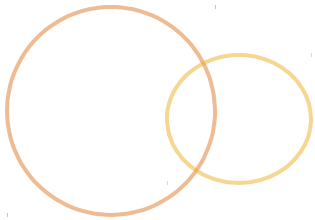
Why REST?

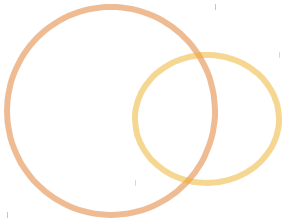
- Universal client
  - So called “low entry barrier”
  - All it takes is HTTP & JSON
- Other reasons are less clear in practice
  - Extensibility—but in practice, changes have a tendency to break clients
  - Uniform Interface—but this is hard to do, and developers often have to learn data structures and URIs
  - Scalability—from caching, but, data often cannot be properly cached due to concurrent server-side changes

REST Interactions with HTTP

- HTTP methods are used in a database-like mode
  - POST - Creates resource
  - GET - Reads resource
  - PUT - Updates resource
  - DELETE - Deletes resource
- And sometimes:
  - PATCH - Partial update of resource
- HTTP specification allows arbitrary “user-defined” methods, but this should be avoided

Example





- System has Customer entities, with name and address. Address has street, city, state, zip
- Read all customers:
  - `GET /customers`
- Read customer with PK 1234
  - `GET /customers/1234`
- Get address of customer with PK 1234
  - `GET /customers/1234/address`
- Get zip of customer with PK 1234
  - `GET /customers/1234/address/zip`

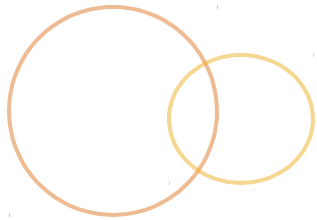
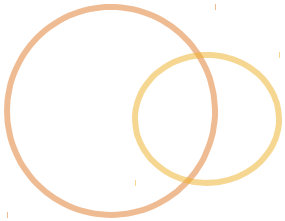
Example

- GET /customers/1234
  Accept: application/json

- HTTP response entity:

```
{ “name” :  “FloorMart”,
  “address” : {
    “street” : “123 Acacia Gdns”,
    “city” : “Rainbowville”,
    “state” : “OZ”,
    “zip” : “00000”
  }
}
```

Example


- Create a new customer
- POST /customers
  Content-type: application/json

```
{ “name” :  “GarageDepot”,
  “address” : {
    “street” : “1 Storage Lane”,
    “city” : “Exhausttown”,
    “state” : “OZ”,
    “zip” : “00000”
  }
}
```

Hierarchical Structure

- URI structure, left to right qualifies scope
- A whole customer
  - /customers/1234
- Find a customer, then select the name
  - /customers/1234/name
- Find the customer, get the list of suppliers, get the third item in the list, then get the name
  - /customers/1234/suppliers/2/name

Handling Actions

- CRUD operations are easy to understand, but REST services aren't just for database access
  - How can we trigger behavior? E.g. pay this invoice, print this document
- Intuit standards prefer avoiding verbs in URIs
  - Avoid: POST /invoices/1234/make-payment

Action Jobs

- `POST /print-requests`
  [[Entity body is document to print]]
  - Response: `Location: 1234`
- Allows, optionally, job control features:
- `GET /print-requests/1234`
  [[Might reports status of job]]
- `DELETE /print-requests/1234`
  [[Cancel the print job]]

The “operation” Query Parameter

- If the “action jobs” approach is unmanageable for a given situation, Intuit standards permit using a query parameter “operation” to specify the action to be taken

Options For Data Transfer

- Client sends:
  - HTTP method
  - URI
  - Query parameters
  - Headers
  - Entity body
- Server returns:
  - Status code
  - Headers
  - Entity body

Data / Metadata From Client

- URI may contain “parameters”, such as primary key or index into a list
  - Use to qualify what is being addressed
- Query parameters are suitable for non-hierarchical “qualification”
  - E.g. query language expression

Data / Metadata From Client

- Headers should not change the essential nature of the request, but are suited for language selection, security credentials, and other metadata
  - Used extensively by HTTP standard for such things as specifying/requesting entity and character encoding
- Entity body carries representation, possibly partial, of the target resource
  - Note that GET requests do not carry entity bodies

Data / Metadata From Server

- Status code indicates success, failure, deferment, etc. of request
  - Might carry some information about nature of failure
- Headers carry similar metadata as in client request: charset / entity body encoding, date/time, and other “metadata”
  - Intuit uses several proprietary headers
- Entity body carries representation, possibly partial, of the result

Intuit's Tomcat Reference Service

- Intuit "starting point" project
- A.k.a. "TCRS"
- Find it on The Portal, search for TCRS or Tomcat
  - devinternal.intuit.com
- Starting point includes:
  - Jersey pre-installed & configured
  - Swagger pre-installed & configured
  - Gateway security/IUS integration
  - Splunk & logging integration
  - CORS handling

Lab Exercise

- Identify a service you are familiar with (theoretical or real)
- Identify a domain entity it represents
- Describe three levels (/x, /x/y, /x/y/z) of URI that the service could use
- Identify one “operation” the service might provide and alternative APIs for that operation
- Present questions, difficulties, and conclusions to the class