REPRESENTATIONAL STATE TRANSFER

REST

What is REST?

- REST is a design pattern.
- It is a certain approach to creating Web Services.
- To understand the REST design pattern, let's look at an example (learn by example).

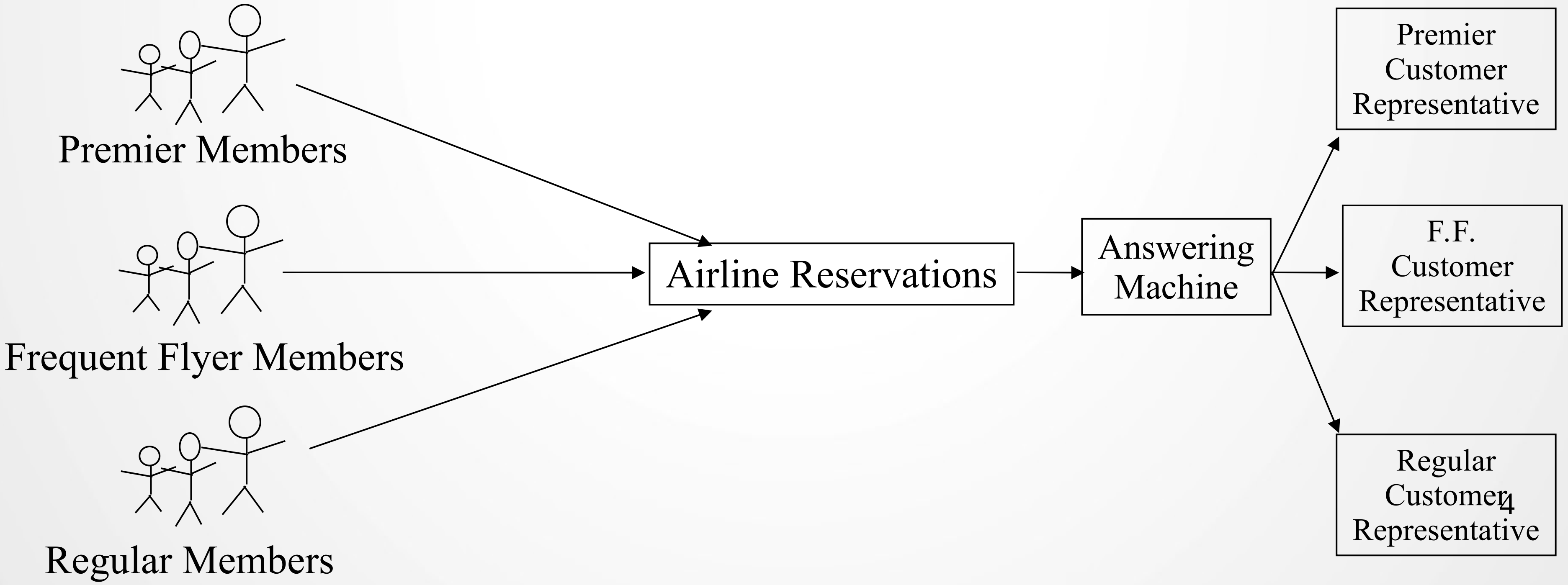
Example:
Airline Reservation Service

- Suppose that an airline wants to create a telephone reservation system for customers to call in and make flight reservations.
- The airline wants to ensure that its premier members get immediate service, its frequent flyer members get expedited service and all others get regular service.
- There are two main approaches to implementing the reservation service...

Approach 1
"Press 1 for Premier, Press 2 for…"

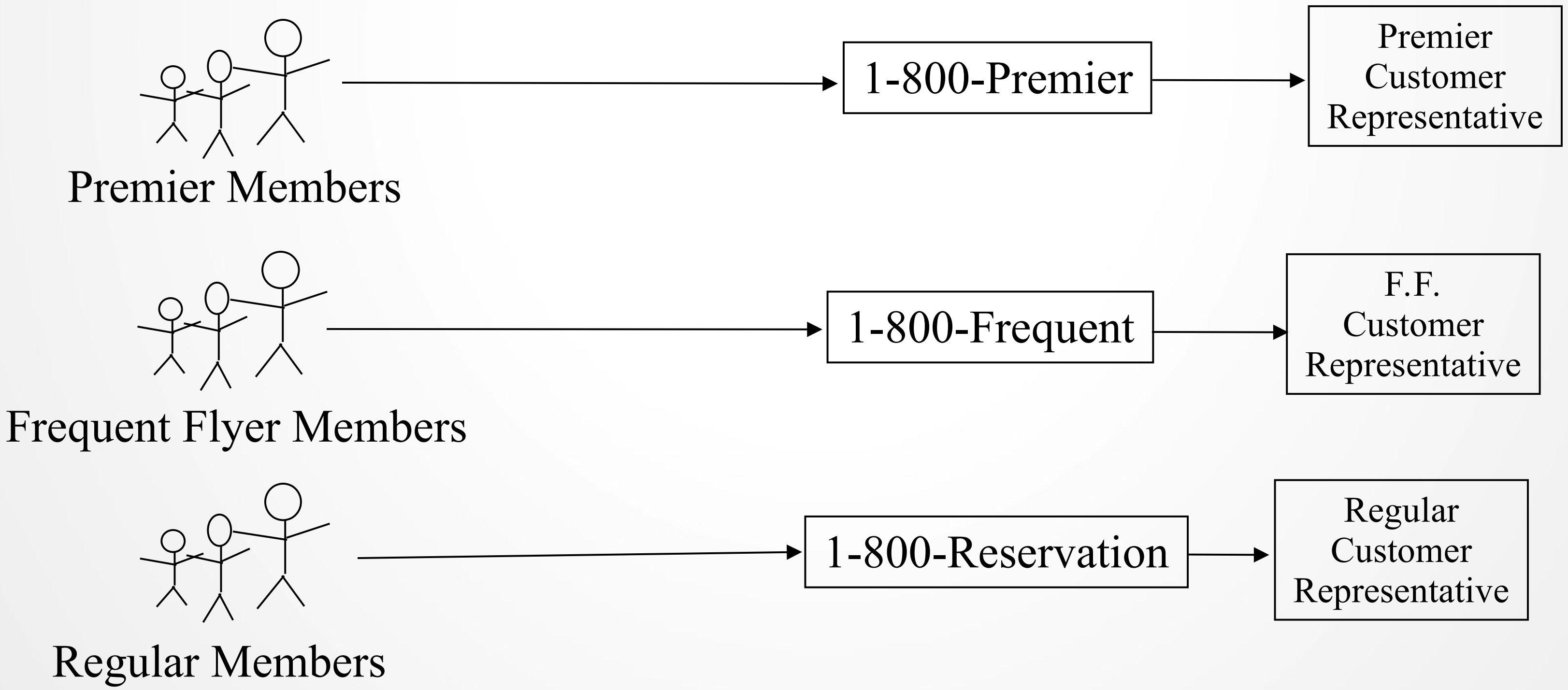
The airline provides a single telephone number.

Upon entry into the system a customer encounters an automated message, "Press 1 if you are a premier member, press 2 if you are a frequent flyer, press 3 for all others."


Premier Members
Frequent Flyer Members
Regular Members
Airline Reservations
Answering Machine
Premier Customer Representative
F.F. Customer Representative
Regular Customer Representative

Approach 2
Telephone Numbers are Cheap! Use Them!

The airline provides several telephone numbers - one number for premier members, a different number for frequent flyers, and still another for regular customers.



Discussion

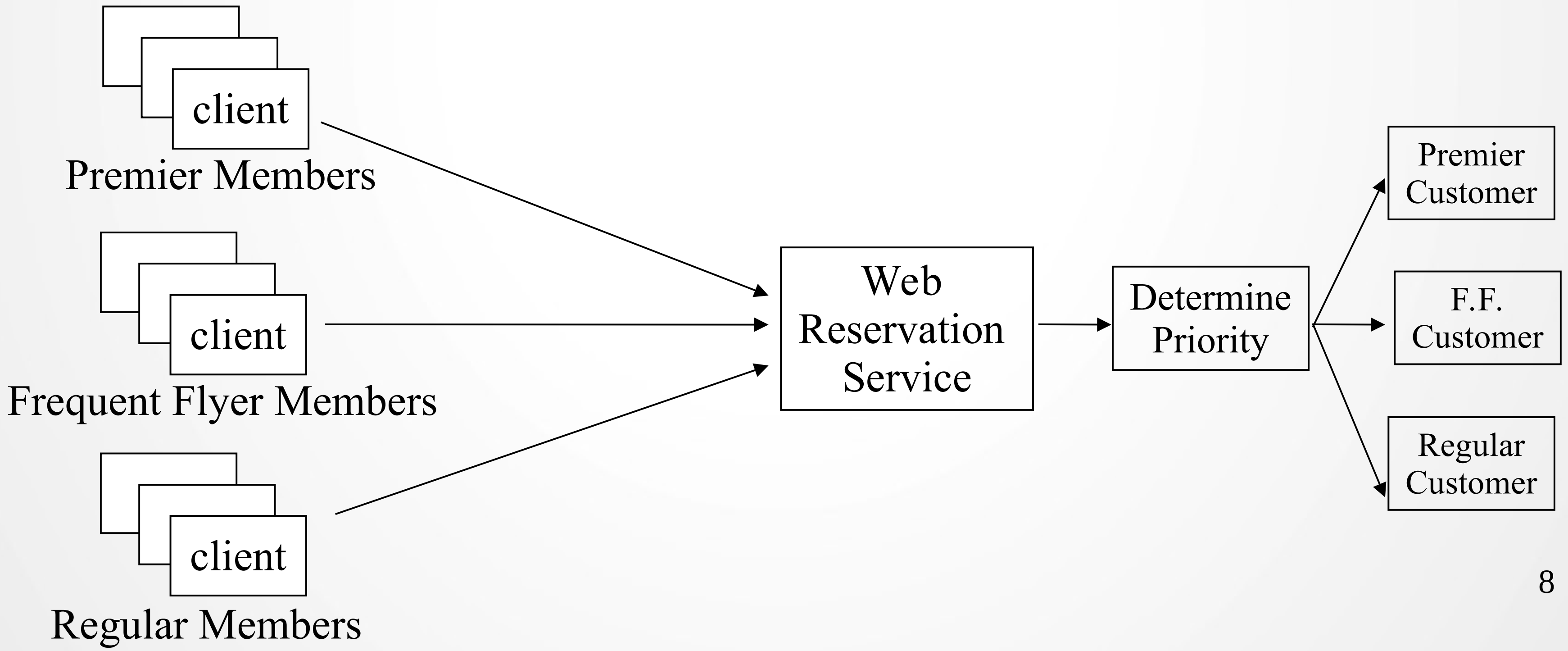
- In Approach 1 the answering machine introduces an extra delay, which is particularly annoying to premier members. (Doesn't everyone hate those answering systems)
- With Approach 2 there is no intermediate step. Premier members get instant pickup from a customer service representative. Others may have to wait for an operator.

Web-Based Reservation Service

- Suppose now the airline (kings-air.com) wants to provide a Web reservation service for customers to make flight reservations through the Web.
- Just as with the telephone service, the airline wants to ensure that its premier members get immediate service, its frequent flyer members get expedited service, all others get regular service.
- There are two main approaches to implementing the Web reservation service. The approaches are analogous to the telephone service...

Approach 1
One-Stop Shopping

The airline provides a single URL. The Web service is responsible for examining incoming client requests to determine their priority and process them accordingly.

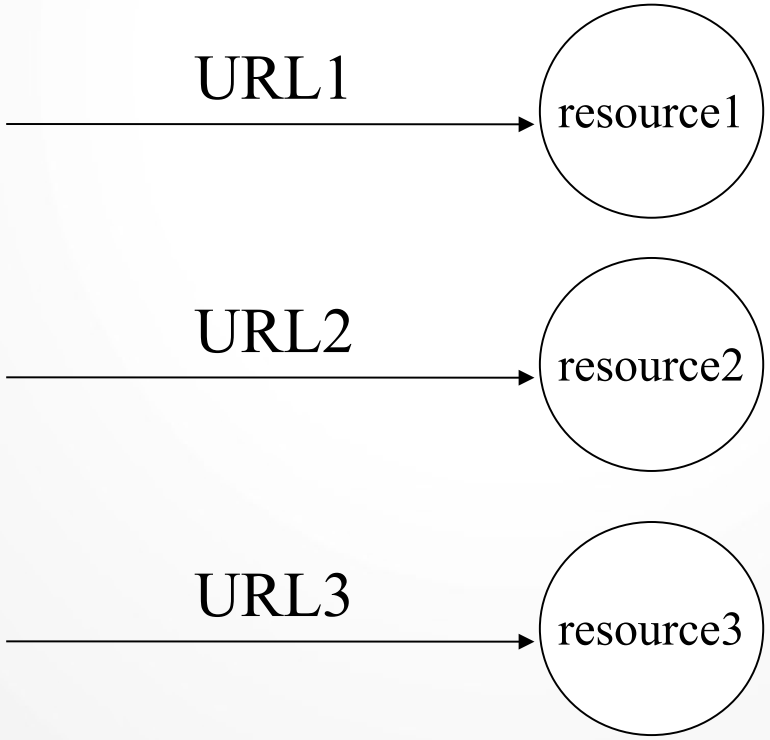


Approach 1 Disadvantages

- There is currently no industry accepted practice (rules) for expressing priorities, so rules would need to be made. The clients must learn the rule, and the Web service application must be written to understand the rule.
- This approach is based upon the incorrect assumption that a URL is "expensive" and that their use must be rationed.
- The Web service is a central point of failure. It is a bottleneck. Load balancing is a challenge.
- It violates Tim Berners-Lee Web Design, Axiom 0 (see next slide).

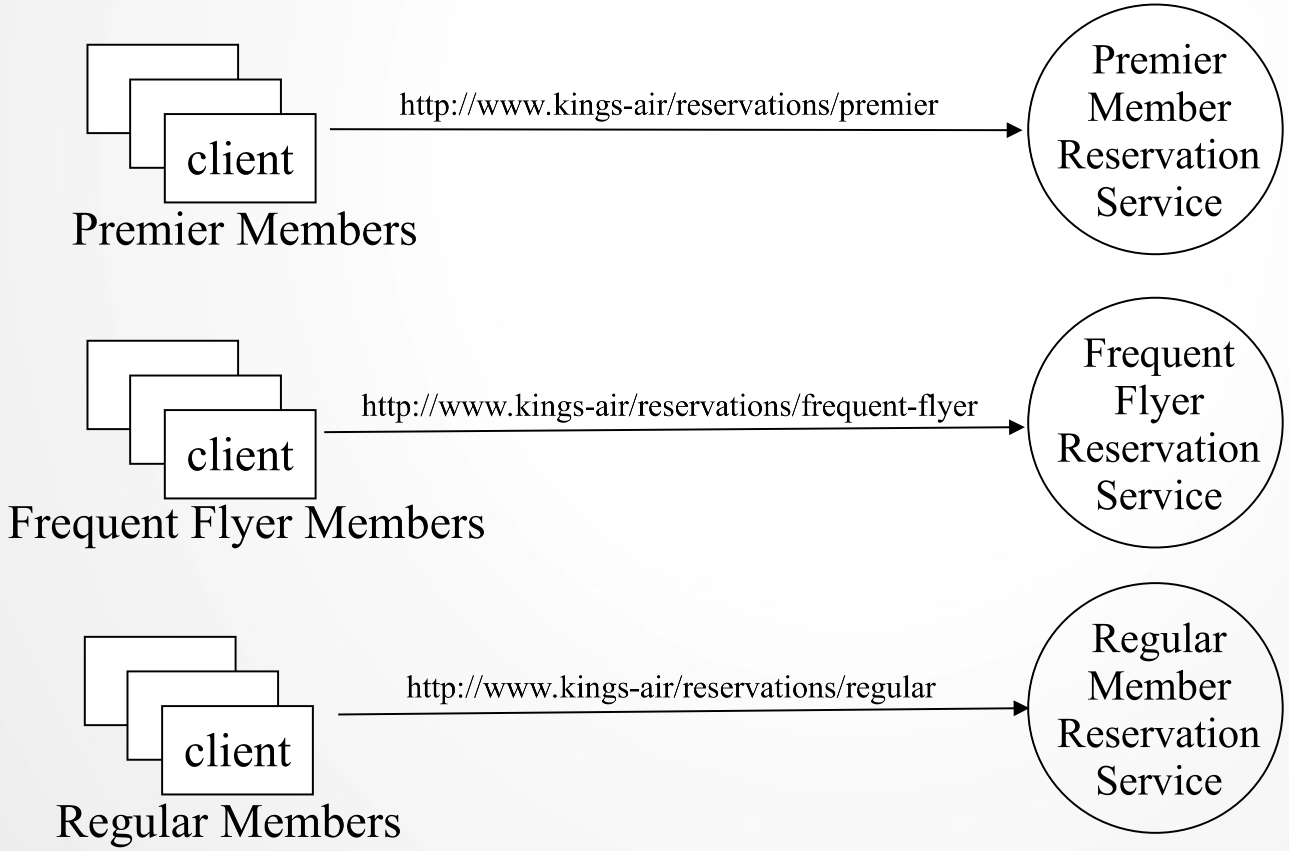
Web Design, Axiom 0
(Tim Berners-Lee, director of W3C)

- Axiom 0: all resources on the Web must be uniquely identified with a URI.



Approach 2: URLs are Cheap! Use Them!

The airline provides several URLs - one URL for premier members, a different URL for frequent flyers, and still another for regular customers.


client
Premier Members
http://www.kings-air/reservations/premier
Premier Member Reservation Service
client
Frequent Flyer Members
http://www.kings-air/reservations/frequent-flyer
Frequent Flyer Reservation Service
client
Regular Members
http://www.kings-air/reservations/regular
Regular Member Reservation Service

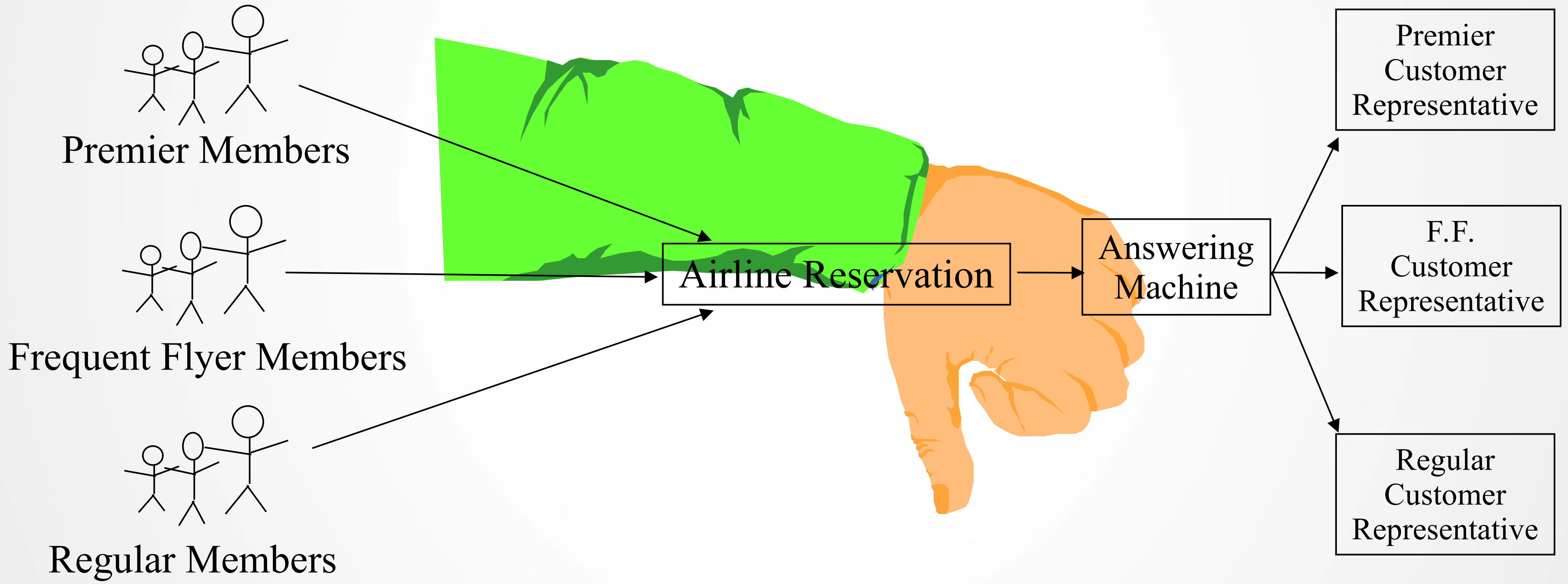
Approach 2 Advantages

- The different URLs are discoverable by search engines and UDDI registries.
- It's easy to understand what each service does simply by examining the URL, *i.e.,* it exploits the Principle of Least Surprise.
- There is no need to introduce rules. Priorities are elevated to the level of a URL. "What you see is what you get."
- It's easy to implement high priority - simply assign a fast machine at the premier member URL.
- There is no bottleneck. There is no central point of failure.
- Consistent with Axiom 0.

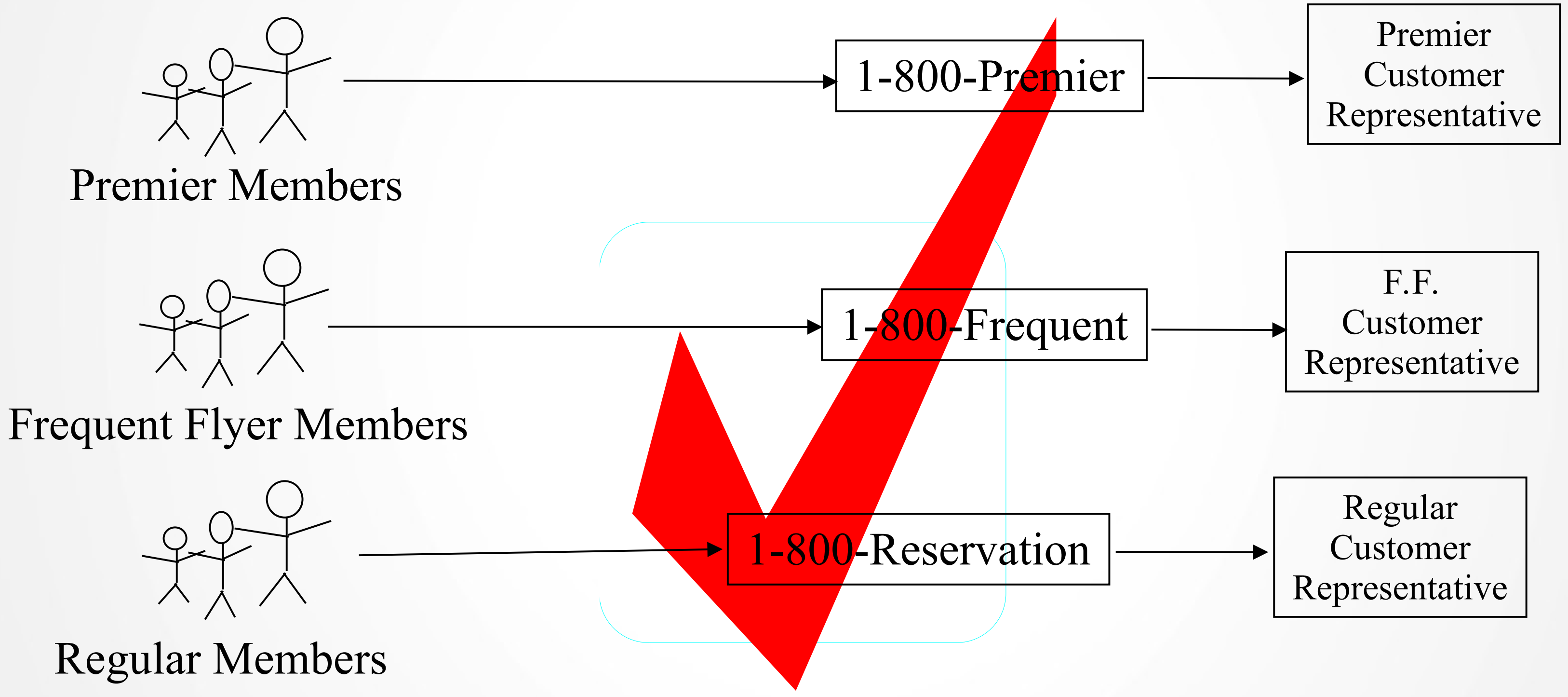
Recap

- We have looked at a reservation service.
- We have seen a telephone-based version and a Web-based version of the reservation service.
- With each version we have seen two main approaches to implementing the service.
- Which approach is the REST design pattern and which isn't? See the following slides.

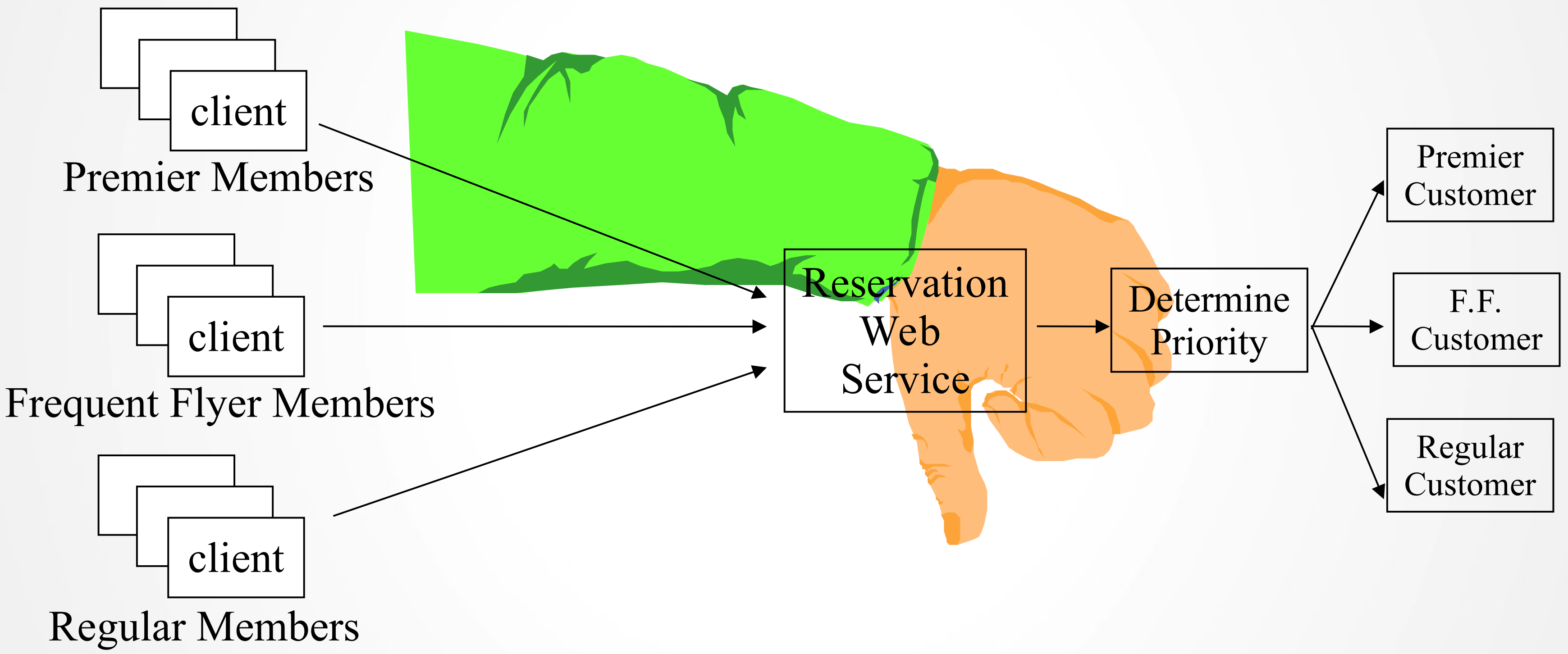
This Ain't the REST Design Pattern


Premier Members
Frequent Flyer Members
Regular Members
Airline Reservation
Answering Machine
Premier Customer Representative
F.F. Customer Representative
Regular Customer Representative

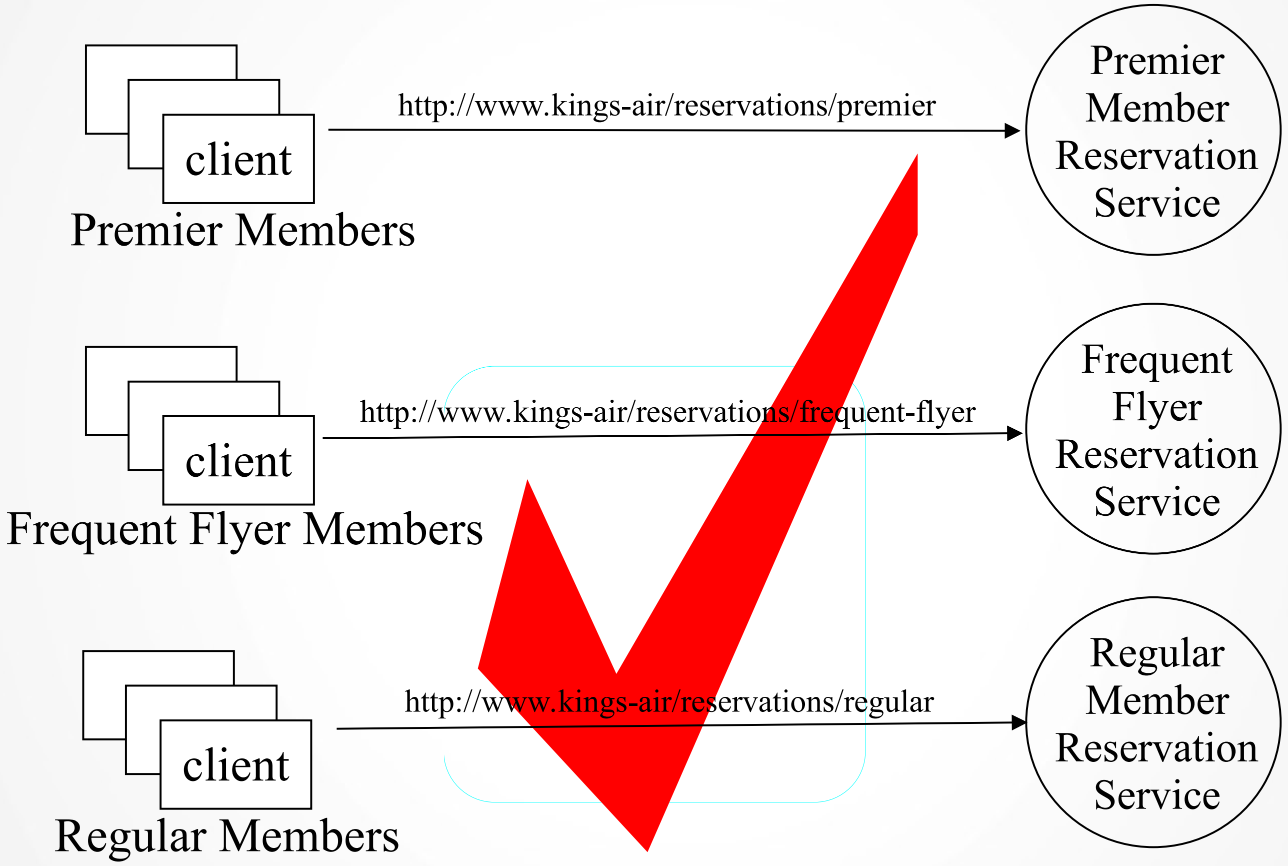
This is the REST Design Pattern



This ain't the REST Design Pattern



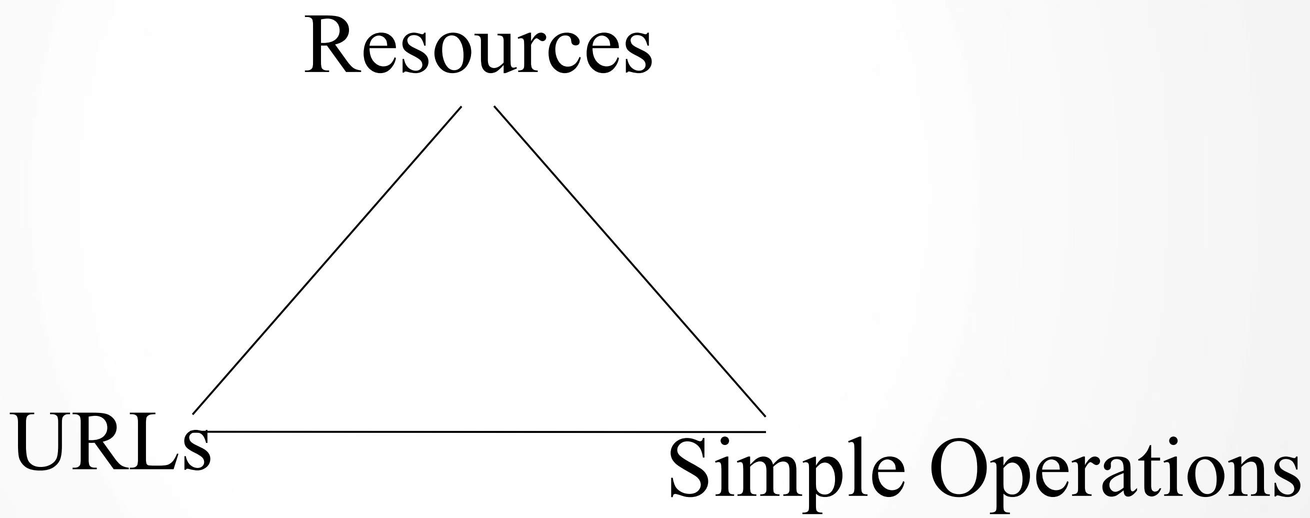
This is the REST Design Pattern


client
Premier Members
http://www.kings-air/reservations/premier
Premier Member Reservation Service
client
Frequent Flyer Members
http://www.kings-air/reservations/frequent-flyer
Frequent Flyer Reservation Service
client
Regular Members
http://www.kings-air/reservations/regular
Regular Member Reservation Service

Two Fundamental Aspects of the REST Design Pattern

- **Resources**
  Every distinguishable entity is a resource. A resource may be a Web site, an HTML page, an XML document, a Web service, a physical device, *etc.*
- **URLs Identify Resources**
  Every resource is uniquely identified by a URL. This is Tim Berners-Lee Web Design, Axiom 0.

The Three Fundamental Aspects of the REST Design Pattern



In this tutorial we discussed how Resources and URLs are fundamental to REST. In a follow up tutorial we will discuss how Simple Operations are also fundamental to REST.