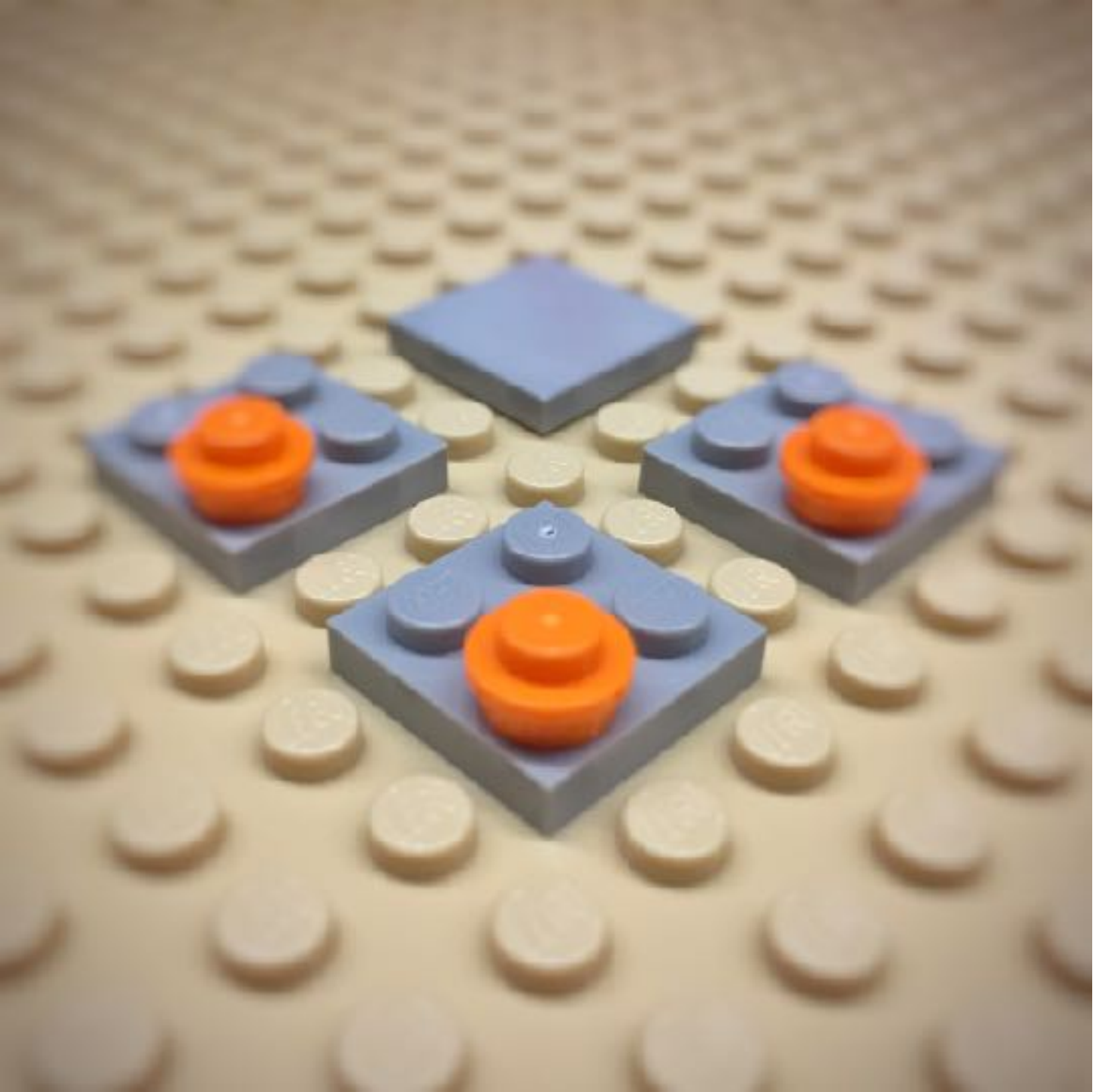


A Day of REST

Exploring Representational State Transfer

Rules for Wile E. Coyote & Road Runner

- The Road Runner cannot harm the coyote except by going "BEEP-BEEP!”
- No outside force can harm the coyote - only his own ineptitude or the failure of the Acme products.
- The coyote could stop anytime - if he were not a fanatic.
  - *"A fanatic is one who redoubles his effort when he has forgotten his aim."* ~ George Santayana
- No dialogue ever, except "BEEP-BEEP!”
- The Road Runner must stay on the road - otherwise, logically, he would not be called Road Runner.
- All action must be confined to the natural environment of the two characters - the southwest American desert.
- All materials, tools, weapons or mechanical conveniences must be obtained from the Acme Corportation.
- Whenever possible, make gravity the coyote's greatest enemy.
- The coyote is always more humilitated than harmed by his failures.

from "Chuck Amuck: The Life And Times Of An Animated Catoonist" by Chuck Jones ISBN 0374123489

Participation

- It is an electronic-technology-free day in the room.
  - Silence your phone, leave laptops at your desk and take analog notes.
  - Totally OK to step out if you have an emergency call.
- Take biology breaks whenever you need to do so.
  - We will have them at fairly regular intervals.
- This subject can be religiously argued -- elsewhere.
  - Respect opinions in a positive manner.
  - Equal time for everyone.
  - The facilitator will step in to ensure all voices can be heard.
- Please provide feedback / evaluation in real time.
  - Notecards and pens are provided.
  - Write it down and put in the Feedback Box.
- Write down your questions and put in the feedback box even if we cover them.
  - Helps us iterate on the course content.

Room Thoughts…

What is REST?

Representational State Transfer (REST)

- REST is an accumulation of design constraints that induce architectural properties: Reliable, cacheable, intermediate processing, shared, scalable, extensible, multi-org., reusable, simple and visible.
- REST is not…
  - An implementation.
  - An architecture.
  - HTTP (although HTTP is RESTful).
  - A URI pattern.
  - Anything you read on the interwebz other than Fielding dissertation + his writings on the subject (for the most part).

The Null Style

Zen: Constraining For Harmony

Exercise: The Null Style

- Groups of 3 or 4.
- Setup
  - Everyone is a LEGO provider.
  - Everyone is a LEGO builder.
- 3 Minutes
- What did you observe with the exercise?

- Rules
- Everyone can obtain or place LEGO.
- Individuals can either obtain or place LEGO but not simultaneously.
- Each of you have to build your project.
- To get the LEGO you need:
  - You CAN ask to obtain with verbal descriptions only.
  - You CAN ask to place LEGO with another person.
  - You CANNOT point.
  - You CANNOT take.
  - You CANNOT show an image.

The Null Style

- An empty set of constraints.
- A system in which there are no distinguished boundaries between components.
- The starting point for our description of REST.

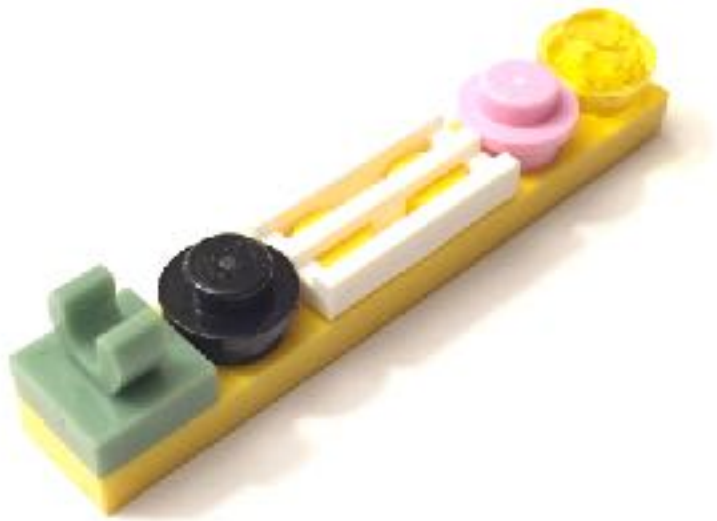
https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm#sec_5_1_1

A word about constraints…

- We tend think the more constraints we have the more the system is ‘locked down’ – it can “wiggle less” as constraints are applied.
- Within REST, each added constraint will allow the overall system to breathe easier.
  - It’s exactly the opposite of locking down – it provides freedom within the system.
- Remember the Wile E. Coyote and Road Runner rules?
  - Those are constraints and they allow the system to function really, really smoothly and be recognizable.
  - Constraints = Rules and boundaries for the system.

Client-Server

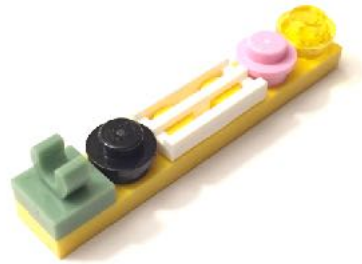
Everything You Eat Must Be A Smoothy … NOT!



Exercise: Client-Server

- Groups of 3 or 4.
- Rules
  - One person is a LEGO provider.
  - Everyone else is a LEGO builder.
  - The provider is the keeper of **all** of the LEGO.
  - Builders can only obtain or place **all** of the LEGO.
- Builders can build anything they want.
- Builders have to remember what they built when they place it back to the provider.
  - Draw it…!
- 30 seconds per person, per group x 2 iterations.
- What did you observe with the exercise?

Client-Server



- A server component, offering a set of services, listens for requests upon those services.
  - Reactive process.
- A client component, desiring that a service be performed, sends a request to the server via a connector.
  - Triggering process.
- The server either rejects or performs the request and sends a response back to the client.
- Separation of concerns is the principle behind the client-server constraints.
- Value
  - Improve scalability.
  - Evolve independently.

https://www.ics.uci.edu/%7Efielding/pubs/dissertation/net_arch_styles.htm#sec_3_4_1

Day 1 Review

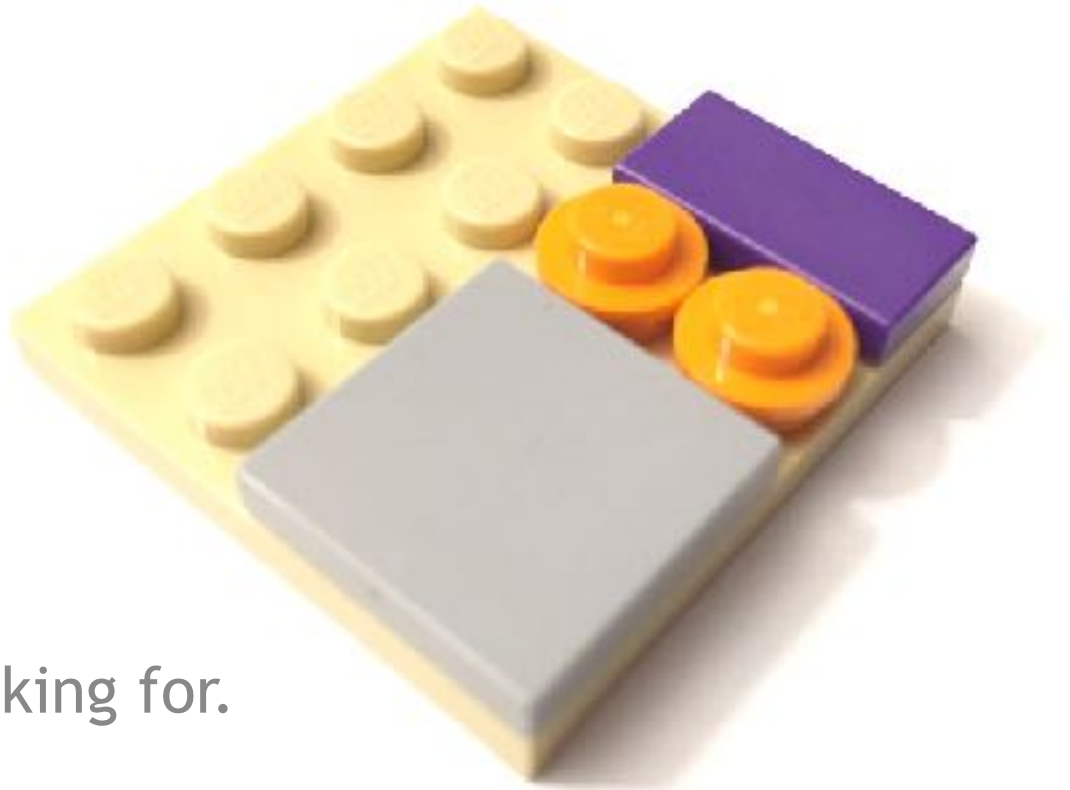
- What is REST?
  - An accumulation of design constraints that induce architectural properties: Reliable, cacheable, intermediate processing, shared, scalable, extensible, multi-org., reusable, simple and visible.
- What are constraints?
- Constraint One: The Null Style
- Constraint Two: Client-Server

A note about the exercises…

- They do not build upon each other.
  - The exercise we just did has no relation to the next one.
- They are designed to illustrate the constraint in isolation.
  - That's kinda / sorta how REST constraints are in the real world.
  - …because you can leverage the one, some or all constraints individually.
  - …but it's not REST unless you have them all.
  - …but I digress (and we will probably get to this later).
- It is totally OK to compare / contrast exercises to understand the constraint better.
- Everybody gets to keep the LEGO from the exercises – make sure you get a complete set...!

Stateless

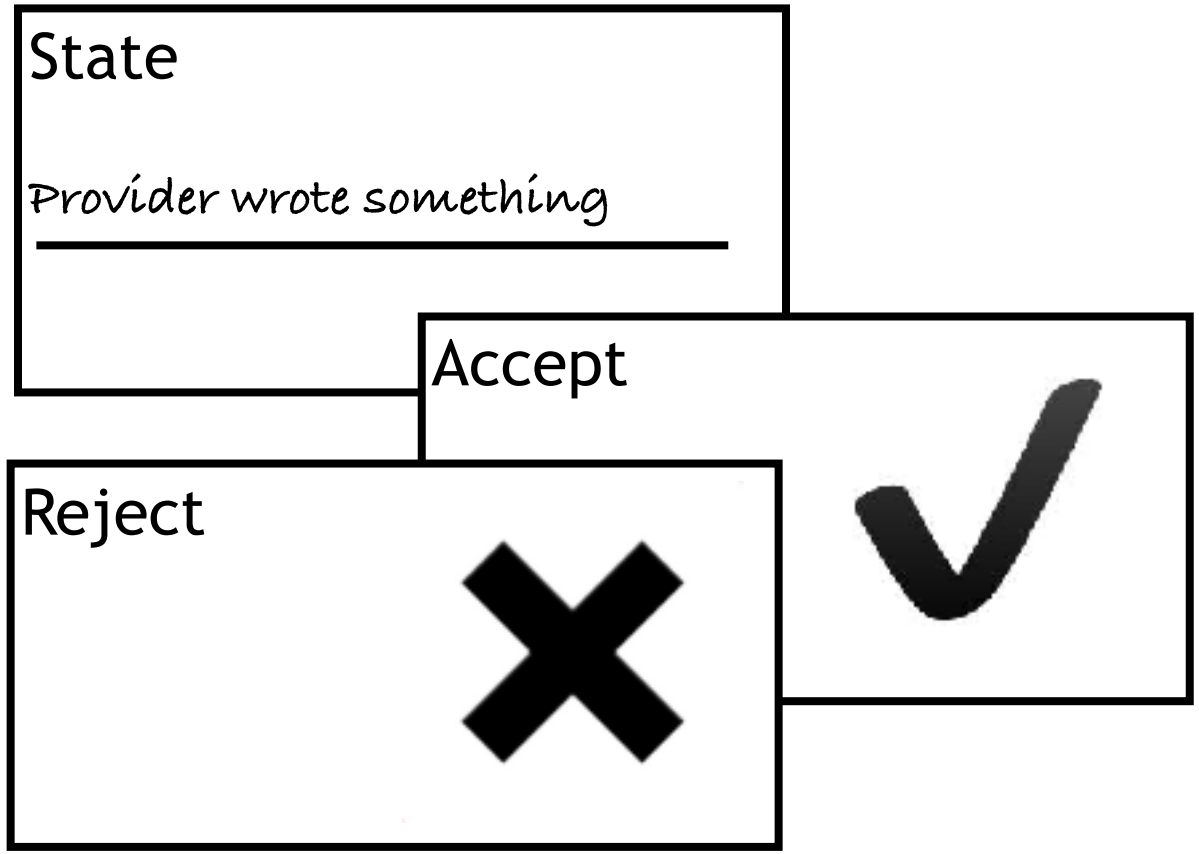
These aren’t the droids you are looking for.



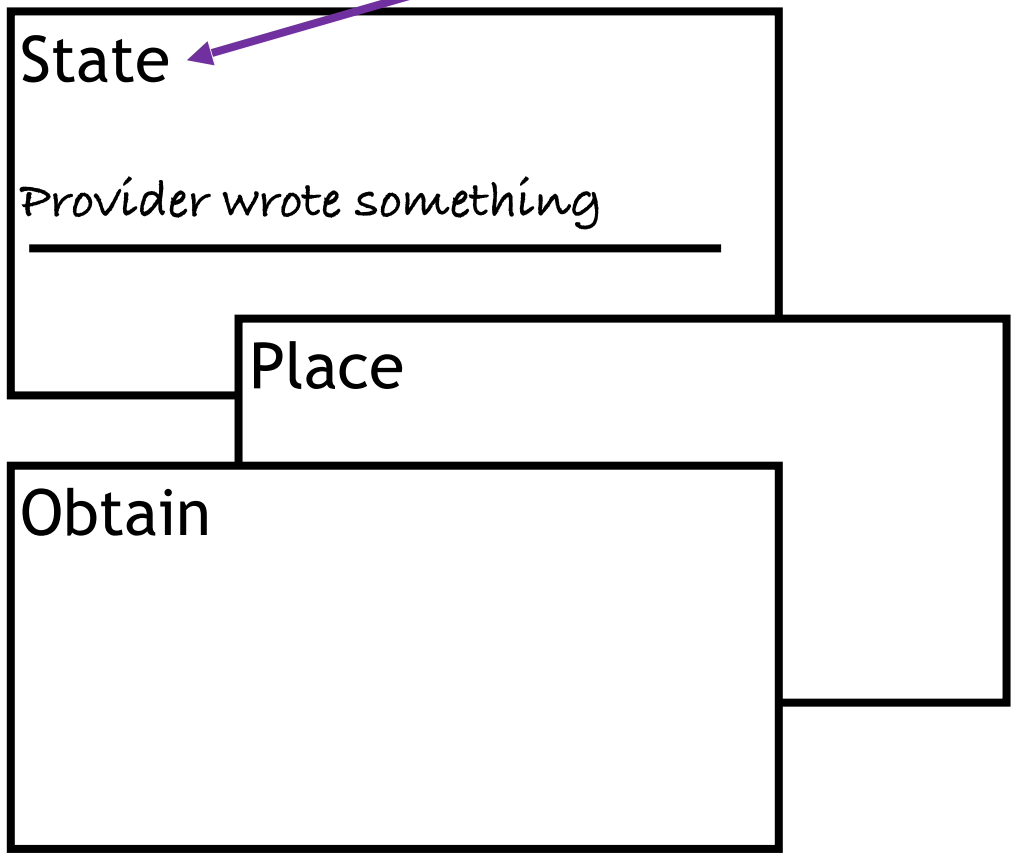
Exercise: Stateless – Playing Cards

Builder can only use what the provider gives to builder.

Provider Cards



Builder Cards

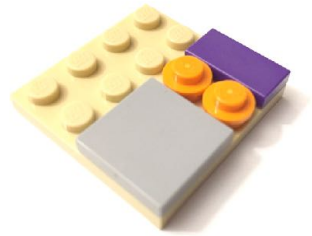


Exercise: Stateless

- Groups of 3 or 4.
- Setup
  - One person is a LEGO provider.
  - Everyone else is a LEGO builder.
- 30 seconds per person, per group x 2 iterations.
- What did you observe with the exercise?

- Provider Rules
  - (a) The provider is all powerful when it comes to the *resource*.
  - (b) Suggestion: The provider can only accept a *representation* placement from a builder if the state the builder gives matches the provider understanding of the resource -- see (a).
  - The provider is the keeper of the LEGO *resource*.
  - The provider can give out many *representations* (aka copies) of the LEGO *resource*.
  - The provider keeps track of changes to the *resource*.
  - The provider provides some way for builders to understand the ***state*** of the *resource*.
- Builder Rules
  - Builders can only obtain or place the LEGO *representations*.
  - For a place: Builder MUST provide the State card.
  - Builders can build anything they want.

Stateless



- No session state is allowed on the server component.
- Each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server.
- Session state is kept entirely on the client.
- Value
  - **Visibility** is improved because a monitoring system does not have to look beyond a single request datum in order to determine the full nature of the request.
  - **Reliability** is improved because it eases the task of recovering from partial failures.
  - **Scalability** is improved because not having to store state between requests allows the server component to quickly free resources, and further simplifies implementation because the server doesn't have to manage resource usage across requests.

https://www.ics.uci.edu/%7Efielding/pubs/dissertation/net_arch_styles.htm#sec_3_4_3

Cache

Reuse / Freecycle / Upcycle / Recycle

Exercise: Cache

- Two groups.
- Setup
  - One person is a LEGO provider prime.
  - Everyone else: provider + builder.
- Walkthrough + discussion.
- What did you observe with the exercise?

- Provider Prime Rules
  - Provider prime is the keeper of the LEGO *resource*.
  - Provider prime provides data for how long the *representations* are valid.
    - Suggestion: Timestamp.
- Provider + Builder Rules
  - Obtain the LEGO *representations*.
  - Provider + builders can provide *representations* to anyone as long as it is still valid.
  - Provider + builders must give data for how long the representations are valid.

Cache

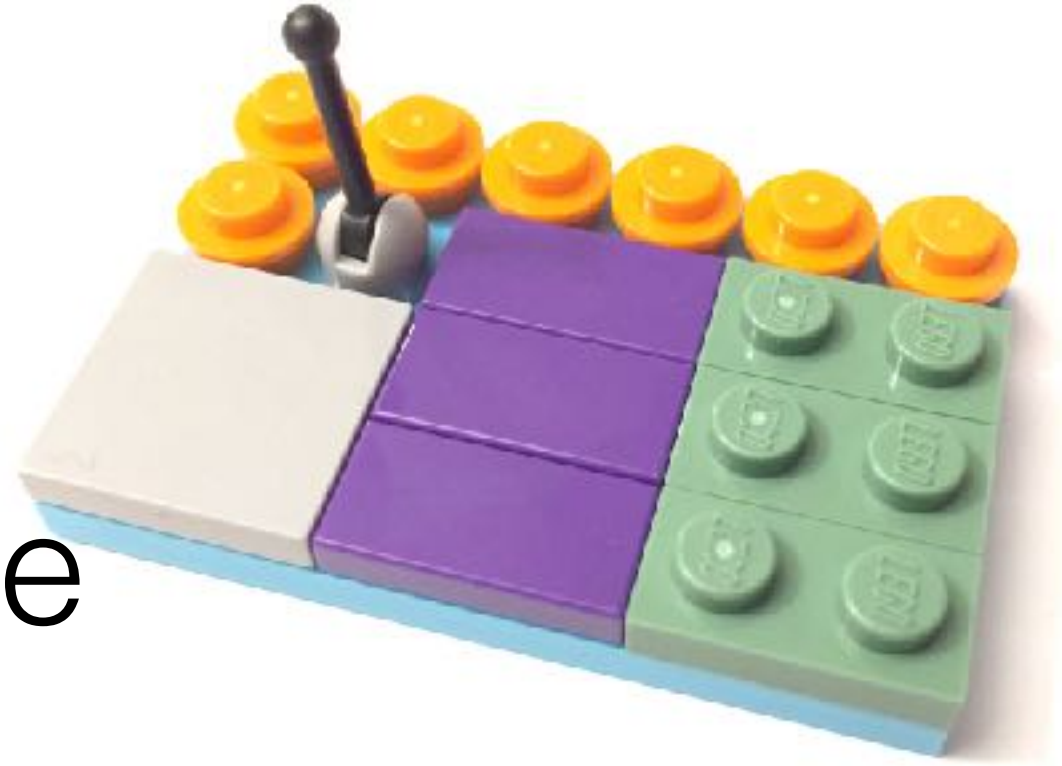
- The data within a response to a request be implicitly or explicitly labeled as cacheable or non-cacheable.
- If a response is cacheable, then a client cache is given the right[1] to reuse that response data for later, equivalent requests.
- A cache acts as a mediator between client and server in which the responses to prior requests can, if they are considered cacheable, be reused in response to later requests that are equivalent and likely[1] to result in a response identical to that in the cache if the request were to be forwarded to the server.
- Value
  - Partially or completely eliminate some interactions, improving efficiency, scalability, and user-perceived performance by reducing the average latency of a series of interactions.

[1] 'Note: '*right*' and '*likely*' are important distinctions. These are cache *hints* and not guarantees because of the client-server and stateless constraints.

https://www.ics.uci.edu/%7Efielding/pubs/dissertation/rest_arch_style.htm#sec_5_1_4

Uniform Interface

Stitching The Uniform



Uniform Interface

1. Identification of resources: All important resources are identified by one resource identifier mechanism.
2. Manipulation of resources through representations: Resources are manipulated through the exchange of representations.
3. Self-descriptive messages: Representations are exchanged via self-descriptive messages.

   A representation consists of data, metadata describing the data, and, on occasion, metadata to describe the metadata (usually for the purpose of verifying message integrity).

4. Hypermedia as the engine of application state: Hypertext as the engine of application state.
5. Access methods have the same semantics for all resources.[1]

Dissertation: https://www.ics.uci.edu/%7Efielding/pubs/dissertation/rest_arch_style.htm#sec_5_1_5
A bit of REST (Representational State Transfer), 2015: http://roy.gbiv.com/talks/201511_Fielding_REST_CF.pdf

[1]*The "Access methods have the same semantics for all resources" does not appear in the original Fielding dissertation, explicitly stated.*

Exercise: Uniform Interface

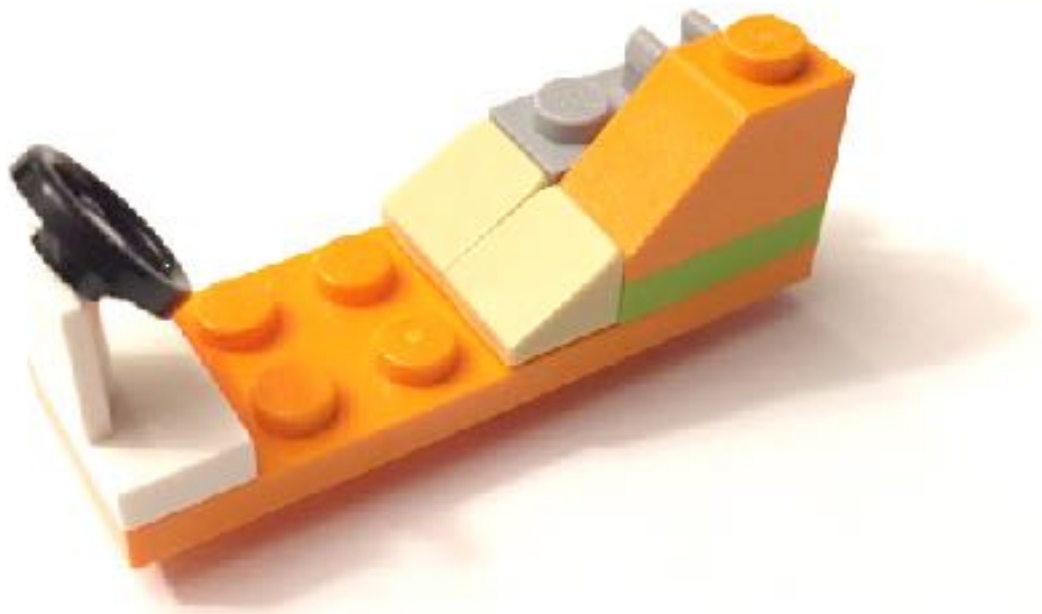
- Groups of two or three.
- Setup
  - Everyone collaborates in the provider role to design the Uniform Interface for their LEGO.
- Design.
- What did you observe with the exercise?

- Rules
  - Design these parts of the Uniform Interface:
    - Identifiers
    - Representations[1]
    - Self-descriptive messages.
    - HATEOAS
    - Access methods[2]

[1] We learned about and worked with these in Stateless and Cache constraints.
[2] We've got these in 'place' and 'obtain'.

Layered System

Having Our Cake And Eating It Too

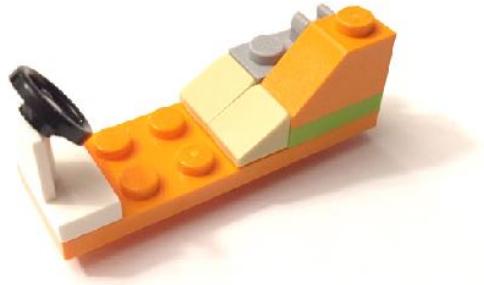


Exercise: Layered System

- Setup
  - 1 provider prime of LEGO
  - 2 provider of LEGO
  - Everyone else: A builder
- What did you observe with the exercise?

- Rules
  - Provider Prime
    - The provider prime can obtain LEGO from the 2 other providers.
    - The provider prime needs to obtain 1 representation from the provider of LEGO.
  - Providers
    - Supply LEGO to Provider Prime when asked.
  - Builder
    - The builder can obtain LEGO from the provider prime.
  - Fast Substitutes
    - Can replace any provider at any time, as often as you wish.

Layered System



- Organized hierarchically, each layer providing services to the layer above and using services of the layer below.
- Value
  - Reduce coupling across multiple layers by hiding the inner layers from all except the adjacent outer layer, thus improving evolvability and reusability.
  - Adds proxy and gateway components to the client-server style.

https://www.ics.uci.edu/%7Efielding/pubs/dissertation/rest_arch_style.htm#sec_5_1_6

Wrap Up

Representational State Transfer

- The Null Style
- Client-Server
- Stateless
- Cache
- Uniform Interface
  - Identification of resources
  - Manipulation of resources through representations
  - Self-descriptive messages
  - Hypermedia as the engine of application state
  - Access methods have the same semantics for all resources.
- Layered System
- Code On Demand ← Optional, so of course we didn't talk about it. ;-)