API Adventures

API Adventures

I have mentioned “decoupled design” a few times in lectures. I have specifically talked about decoupling UI.

This manifests the single-responsibility principle from SOLID and creates a decoupled design.

Coupling

Types of Coupling (least to most-coupled):

- **Data Coupling:** passing or sharing simple parameters
- **Stamp Coupling**: passing or sharing a data structure
- **Control Coupling:** If the modules communicate by passing control information

Coupling

Types of Coupling (least to most-coupled):

- **Data Coupling:** passing or sharing simple parameters
  - myAdv.doAction(“go”, “north”); // parameters: verb, object
  - myAdv.doAction(myArgsList[0], myArgsList[1]);
  - myAdv.doAction(myUI.getVerb(), myUI.getObject());

What about this?

```
myAdv.DoAction(myUI);
```

Coupling

Types of Coupling (least to most-coupled):

- **Data Coupling:** passing or sharing simple parameters
- **Stamp Coupling**: passing or sharing a data structure
  - myAdv.doAction(argsList); // assumed location of verb, object
  - myAdv.doAction(argsMap);
- **Control Coupling:** If the modules communicate by passing control information

Coupling

Types of Coupling (least to most-coupled):

- **External Coupling:** the modules depend on other modules, external to the software being developed or to a particular type of hardware. Ex- protocol, external file, device format, etc.
- **Common Coupling:** The modules have shared data such as global data structures.
- **Content Coupling:** In a content coupling, one module can modify the data of another module, or control flow is passed from one module to the other module. This is the worst form of coupling and should be avoided.

API Adventures

- Your UI will need to be decoupled from your core logic.
- You will employ the use of a REST API

Decoupled Design Core Logic

Decides what to do with the response and manages the game state

Think of a REST API as a dumb data server. It should do no work for you other than serving up the data as requested. It is stateless. That is, it does not keep previous interactions.

Your logic will decide what data is needed in response to the user’s interaction.

REST

REST is an acronym for RE*presentational* S*tate* T*ransfer*

Let’s think about this for a moment:

“Representational state transfer”

REST

REST is a set of architectural constraints, not a protocol or a standard. Among these constraints:

1. **Client-Server Architecture**: the user interface of the website/app should be separated from the data request/storage.
2. **Statelessness**: the communication should have no client context stored on server. This means each request to the server should be made with all the required data and no assumptions should be made if the server has any data from previous requests.
3. **Layered system**: client should not be able to tell if it is communicating directly with the server or some intermediary.

Representation Formats

REST is a way to exchange STATE via some representation of an object.

That representation can be in any one of several formats:

- JSON
- HTML
- XLT
- Python
- PHP

JSON is a representation of a data state. You read the serialized data (JSON text) into POJOs (de-serialized) and vice-versa.

Resources

The key abstraction of information in REST is a resource. Any information that we can name can be a resource.

A REST resource can be a document or image, audio file, etc

Resources

For this assignment, you will use resources other than text in order to embellish or enrich the user experience.

Resource representations consist of:

- the data itself
- the metadata - data describing the data
- and the hypermedia links that can help the clients in transition to the next desired state.

HTTP Requests

You may be familiar with the four basic HTTP requests a client can make:

- GET: To retrieve a resource.
- POST: To create a new resource.
- PUT: To edit or update an existing resource.
- DELETE: To delete a resource.

REST != HTTP

REST is a set of architectural constraints. The HTTP requests are how you will interact with the server.

Assignment

https://courses.grainger.illinois.edu/cs126/sp2021/assignments/api-adventures/

Helpful Links

REST APT
https://www.redhat.com/en/topics/api/what-is-a-rest-api

REST one-stop shop
https://www.freecodecamp.org/news/rest-api-tutorial-rest-client-rest-service-and-api-calls-explained-with-code-examples/

Example code REST API
https://happycoding.io/tutorials/java-server/rest-api#simple-java-rest-client

Coupling
https://www.toolbox.com/tech/enterprise-software/blogs/design-principles-coupling-data-and-otherwise-050307/