Services

- A lot of confusion with this term
  - Sometimes they're the public interface of your app
  - Other times they reflect connections to third parties, like Amazon or Salesforce
  - Doesn't matter - let's just move to the concept
- Services are the public face to your controllers
  - These classes allow for calling methods that create, persist, modify, and retrieve data
  - Used to translate and process information from user input or other helper classes

Services (con't)

- Services are responsible for providing the Data Model or collection to the consumer
- Services act as the layer to work with your external consumer or framework
- Services do processing that is the same no matter what Data Model, Data Mapper, or other Service is involved
  - retrieving data from a helper model, filtering search term and city/state, etc.

Services (con't)

- Harddrive example!
  - Service::save() would accept a harddrive model, grab the proper mapper (perhaps the harddrive mapper associated with mysql), and introduce those two. It would then reintroduce the saved model

  - Service::findBySerialNumber() would accept a serial number parameter, translate that into a parameter for the mapper, get the mapper, retrieve the collection, and pass that back to the consumer

Services (con't)

- Harddrive example (con't)
  - Service::validateUpdate() - would take user input in, validate that it is valid and could match up with the Data Model, and return whether the input was acceptable or not
  - Service::populateModel() - takes that validated user input and the model, calls the mapper with this information to populate it (mapper knows fields, not services), and returns it

Why are Services Important?

- Services are the public interfaces to the rest of the classes in your app
- Things that you allow 'the consumer' to do are only defined in the service layer
- If data sources, mappers, models change, your service stays the same, and so does the consumer side of your app

Summarize Services

- Public interface to the rest of your code
- Understand Data Models, Domain Models, Mappers
- Execute processing that is always the same, never differs no matter what objects are in use