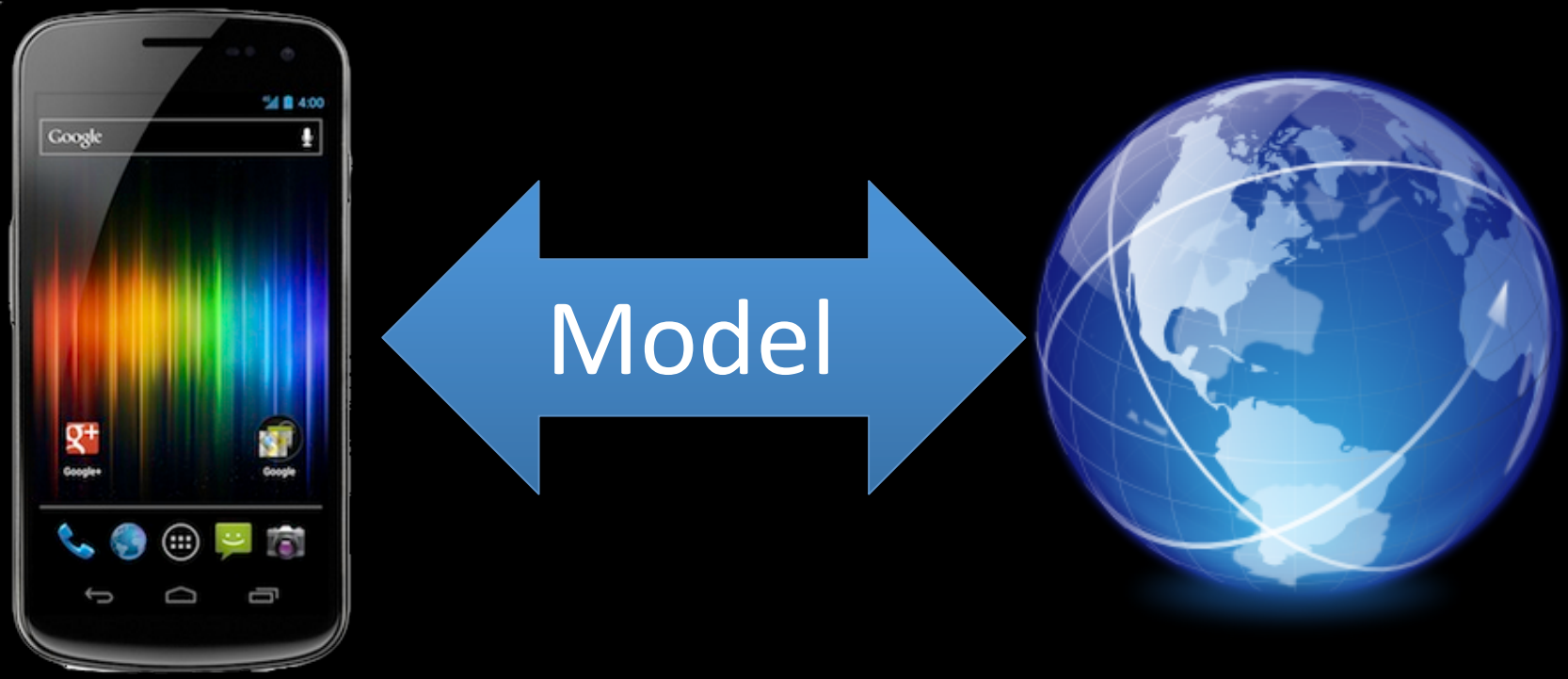
Web Services and Models

Check out our new model service on the web at models.com

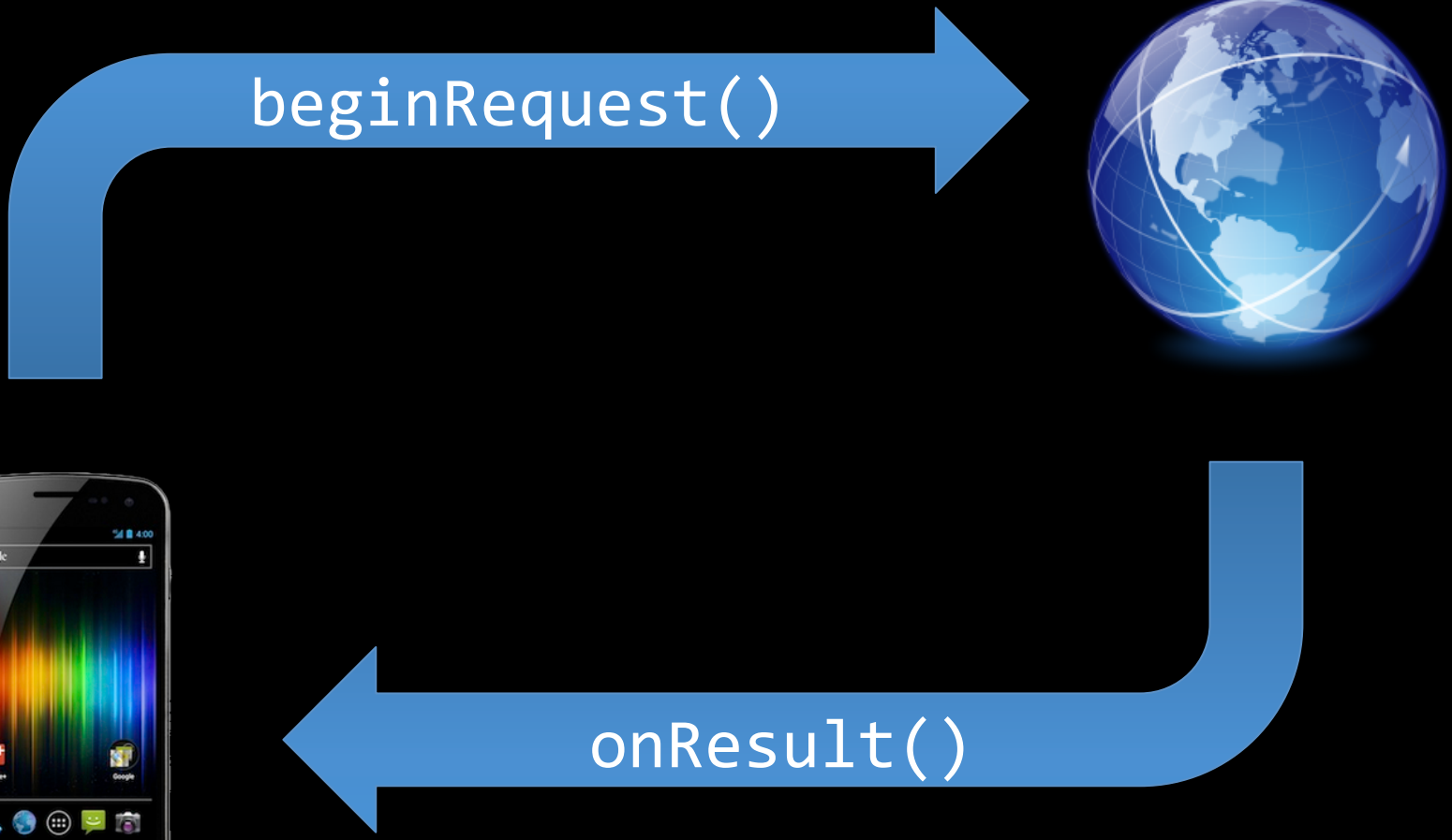
Model View Controller



Model for Remote Data

- Single class to handle all networking
- Implements caching where needed
- Generally shared or singleton
- Abstracts how data is stored

Asynchronous Models



Asynchronous Models

- Use non-blocking communication
- Request returns immediately and performs work on background thread
- Data is returned to delegate/listener later on
- Decouples the data and how the data is used

Model + Listener

- Model has a listener like a view
- All data for all requests go to same listener

```
Model.setListener(new Model.Listener() {
      onBegin(int method) {…}
      onResult(String result, int method) {…}
      onError(Exception e, int method) {…}
});
Model.beginXRequest();
```

Model + Delegate

- Every request has its own delegate
- Each delegate is specialized for the request

```
Model.beginXRequest(new Model.Delegate()
{
    onBegin() {…}
    onResult(String result) {…}
    onError(Exception e) {…}
});
```

Model + Cache

- CASH not CASH-AY (fyi)
- Reduces strain on server
- Saves data between sessions
- Can be loaded quickly while fresh data is requested from server
- Allows older data to be viewed offline

Model + Cache

- 2 kinds: transparent and explicit
- Transparent – model returns cached data rather than contact the server
- Explicit – programmer specifically asks for cached data
- Use internal storage cache or Persistent Storage class (see blog article on data persistence)

Model + Classes

- Need other classes to actually hold the data.
- Classes used by View/Controller to display, edit data from Model.
- Might include Student, Post, Comment, etc.

Accessing the Model

- Singleton: model instance is obtained through static method on class.
- Shared: model instance is obtained through custom Application subclass.

Questions?