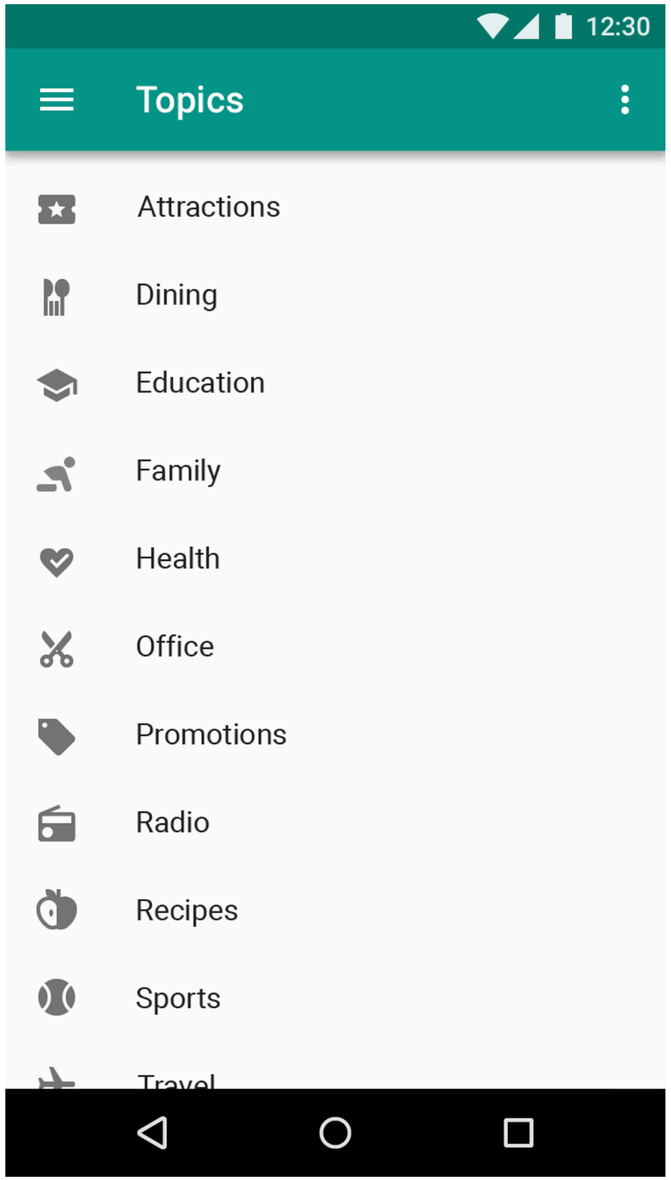


MOBILE APPLICATION DEVELOPMENT

ANDROID (2017)

LECTURE 10: DATA-DRIVEN UI

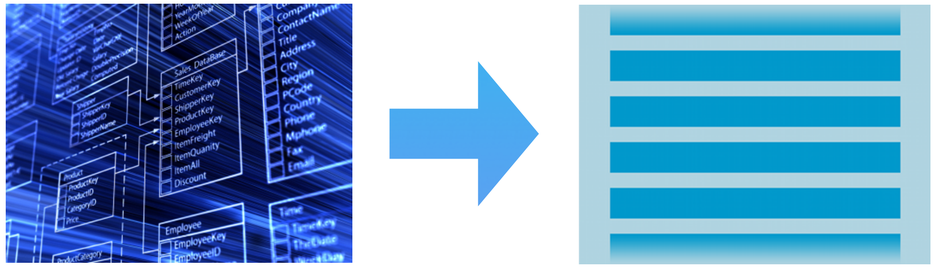
DATA-DRIVEN UI

- The goal of many applications is to provide some sort of an interface which manipulates a user's data.
- Datasets generated by users can be quite large.
  - Social media applications manipulate thousands of images and posts.
  - Document-based applications handle arbitrary numbers of documents.
  - Most applications allow the user to store a number of custom settings.
- Connecting very large (or arbitrarily large) sets of data with a user interface is the purpose of data-driven UI.

PROBLEMS IN DATA-DRIVEN UI

- When working with datasets of unknown size (but a known structure), programmers will encounter the following issues:
  - How can an app running on a small screen present a lot of data effectively?
  - Given the limited resources of any computer, how can an arbitrarily-large dataset be represented efficiently in memory?
  - When attempting to present an arbitrary number of Views to a user, how can the interface remain responsive?
- Android solves these problems with specialized, data-oriented classes designed to present arbitrary amounts of data to the user.

ADAPTER VIEWS

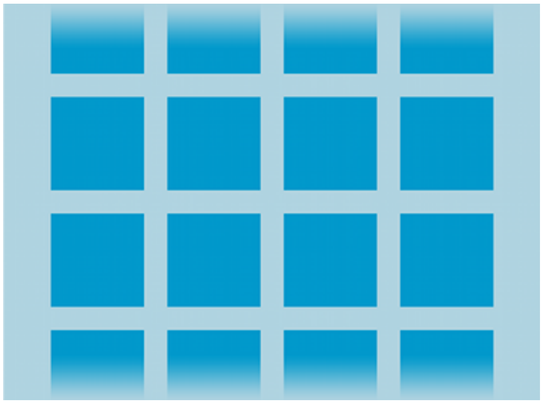


- Android expresses the concept of Data-Driven UI using AdapterViews, which use Adapters to obtain and structure content.
- AdapterViews (and similar classes) allow Android to display potentially-infinite datasets efficiently while maintaining structure and customizability.
- Custom Adapters define the bridge between backend data and frontend presentation.
- Adapter is just an interface and requires a concrete implementation to be useful.
  - Designed to retrieve data as associated Views appear on screen.
  - Used by the AdapterViews to assist in caching data and Views for efficient re-use.

LIST VIEWS

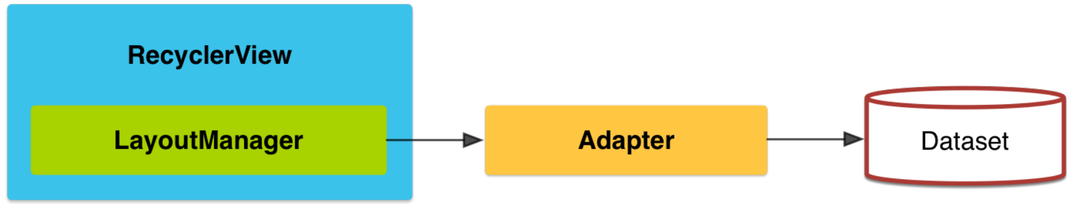
- ListViews present scrollable lists of information.
- Use Adapters to obtain data for presentation.
  - Use ArrayAdapters by default.
  - Custom Adapters should provide data presented in Views which are optimized for use in a scrolling list.
- Can use `setOnItemClickListener()` to provide a listener which responds to clicks on specific rows of the ListView.

GRID VIEWS



- GridViews present scrollable grids of information.
- Use Adapters to obtain data for presentation.
  - Use ArrayAdapters by default.
  - Custom Adapters should provide data presented in Views which are optimized for use in a scrolling grid of items.
- Can use `setOnItemClickListener()` to provide a listener which responds to clicks on specific items within the GridView.
- Can set `numColumns` to change the resolution of the grid.

RECYCLER VIEWS



- RecyclerView is a modern, powerful UI class which can handle flexible layouts and present data in many different ways.
- Uses RecyclerView.Adapters to obtain data for presentation.
  - Does not use a default Adapter class, in exchange for being more flexible.
  - By default, may present content in a list, grid, or staggered grid layout.
  - Custom RecyclerView.LayoutManagers can be provided to completely customize the layout of items in the AdapterView - the Adapter used should work in tandem with the selected LayoutManager to provide appropriate Views.
- AdapterView does not have a default listener interface, so one must be created.