MVC Pattern

- An object-oriented method for separating
  - presentation logic
  - business logic
  - business model
- It facilitates reuse

http://java.sun.com/developer/technicalArticles/javase/mvc
Model

- Represents application data and business rules that govern access to and updates of this data
  - In enterprise software, a model often serves as a software approximation of a real-world process
- Notifies views when it changes and enables the view to query
- Allows the controller to access application functionality encapsulated by the Model

View

- Renders the contents of a model
- Specifies how the model data should be presented
- When the model changes, the view must update its presentation
  - push model
    - the view registers itself with the model for change notifications
  - pull model
    - the view is responsible for calling the model when it needs to retrieve the most current data
- Forwards user gestures to the controller

Controller

- Defines application behavior
- Interprets user gestures and maps them into actions
  - For the model to perform
  - In selecting a different view
    - E.g., a web page of results to present back to the user
- In a Web application, user gestures appear as HTTP requests

MVC Architecture
Best Practices: MVC Separation

- Organize model elements, views and controller actions into “physically” separate:
  - Source code units
  - Source code packages
- Benefits:
  - Helps enforce logical separation of concerns
  - In a large project, a developer with specialized skillset may work on just one of M, V or C
  - Simplifies maintenance, e.g. porting app to new platform
- Frameworks that emphasize MVC separation:
  - Apache Struts
  - JavaServer Faces
  - Ruby on Rails

Interaction between MVC Components (1)

Once the model, view, and controller objects are instantiated, the following occurs:

1. The view registers as a listener on the model
   - Any changes to the underlying data of the model immediately result in a broadcast change notification, which the view receives
   - This is an example of the push model described earlier
   - The model is not aware of the view or the controller
     - It simply broadcasts change notifications to all interested listeners
2. The controller is bound to the view
   - I.e., any user actions that are performed on the view will implicitly invoke a registered event listener method in the controller class
3. The controller is given a reference to the underlying model

Interaction between MVC Components (2)

Once a user interacts with the view, the following actions occur:

1. The view recognizes that a user action has occurred
2. The view generates an event, which implicitly invokes appropriate method in the controller
3. The controller accesses the model
   - Possibly updating it with respect to the user’s action
4. If the model has been altered, it notifies interested listeners, such as the view, of the change
   - The controller may also update the view

MVC Example
Known Examples

- A few well-known examples of RESTful, resource-oriented web services include:
  - Services that expose the Atom Publishing Protocol
  - Atom’s variants such as GData
    - http://code.google.com/apis/gdata/
  - Amazon’s Simple Storage Service (S3)
    - http://aws.amazon.com/s3
  - Most of Yahoo!’s web services
    - http://developer.yahoo.com/
  - Most other read-only web services that don’t use SOAP
  - Static web sites
  - Many web applications, especially read-only ones like search engines

RPC vs. RESTful

- An RPC application (e.g. SOAP-based app) is exposed as one or more network objects, each with an often unique set of functions that can be invoked
  - A client must have knowledge of the object identity to locate it and knowledge of the object type to communicate with it
- A RESTful web application requires a different design approach from an RPC (remote procedure call) application
  - RESTful design constrains the aspects of a resource that define its interface (the verbs and content types)
  - This leads to the definition of fewer types on the network than an RPC-based application but more and unrestricted resource identifiers (nouns)
  - RESTful design seeks to
    - define a set of resources that clients can interact with uniformly,
    - and to provide hyperlinks between resources that clients can navigate without requiring knowledge of the whole resource set

RESTful Design

- RESTful design constrains the aspects of a resource that define its interface (the verbs and content types)
- This leads to the definition of fewer types on the network than an RPC-based application but more and unrestricted resource identifiers (nouns)
- RESTful design seeks to
  - define a set of resources that clients can interact with uniformly,
  - and to provide hyperlinks between resources that clients can navigate without requiring knowledge of the whole resource set
Actions (Verbs)

- Controllers cannot have arbitrary verbs, only these:
  - POST
    - Create - create a new resource
  - GET
    - Retrieve - retrieve a representation of a resource
  - PUT
    - Update - update a resource
  - DELETE
    - Delete - delete a resource

Example: Turn On Light

- RPC style:
  ```java
  proxy.turnLightsOn("ba2185");
  ```

- REST style:
  ```
  PUT http://control.com/ba2185/lights HTTP/1.1
  on
  ```

Success Factor: Uniformity

- The scoping information (“why should the server send this data instead of that data?”) is kept in the URI
  - This is the principle of uniform addressability
- The method information (“why should the server send this data instead of deleting it?”) is kept in the HTTP method
  - This is the principle of the uniform interface
- There are only a few HTTP methods, and everyone knows ahead of time what they do

REST
(Representational State Transfer)


2000
What is REST

- An architectural style for distributed hypermedia systems

Key Constraints

- Client server style
  - Separating the user interface concerns from the data storage concerns
- Stateless
  - Server keeps no session state (only resource state)
  - Each request from client to server must contain all of the information necessary to understand the request
  - Cannot take advantage of any stored context on the server

Stateless: Disadvantages and Benefits

- Disadvantages
  - Client needs to manage session state and its path through the resources
  - Client needs to re-send certain data with each request, thus decreasing network performance
- Benefits
  - Scalability
    - Server can deal with every client in the same way
  - Cacheability
    - GET and PUT are idempotent
    - all methods have no “side effects”
  - Longevity
    - meaning of actions never evolves

Key Constraints (cont’d)

- Uniform interface between components
  - Resource is the unit of identification
  - Resource is manipulated through exchange of representations
  - Resource-generic interaction semantics
  - Self-descriptive messaging
What is a Feed

- A web feed is a data format used for providing users with frequently updated content
- Designed to be machine-readable
- Making a collection of web feeds accessible in one spot is known as aggregation, which is performed by a feed aggregator
  - Receiver end
  - Pulling data from servers
- How it relates to REST
  - A feed is a collection of resources
  - Feed helps to discover resources

Web Syndication

- Syndication makes available a feed for an information source
- Web syndication makes available a section of a website for other sites or individual subscribers
  - Recently added content
  - Weblog entries
  - News headlines
**Atom**
- Atom is designed to be a universal publishing standard for personal content and weblogs
- The Atom Syndication Format is an XML language used for web feeds

**GData**
- A GData feed conforms to either the Atom or RSS syndication formats
  - Atom is the default underlying data format for Google web services
  - To acquire information from a service that supports GData, you send an HTTP GET request; the service returns results as an Atom or RSS feed
  - You can update data (where supported by a particular GData service) by sending an HTTP PUT request

**Google Calendar**
- Feeds are not essential for calendar service
  - Google provides it as a platform for all its web services
  - Google calendar gadget
  - Calendar feeds
    - metafeed
    - allcalendars
    - owncalendars
    - event
      - Requesting a representation
        - user ID (or default)
        - visibility (private, public)
        - projection (full, full-noattendees, composite, attendees-only, free-busy, basic)
  - [http://www.google.com/calendar/feeds/default/private/full](http://www.google.com/calendar/feeds/default/private/full)