# Application Design

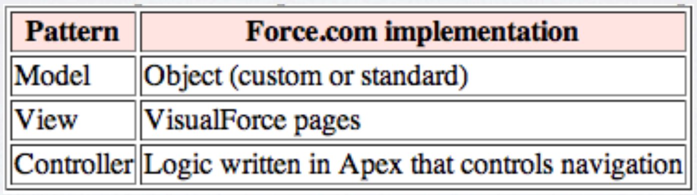
- Components of an Application
- Describe model-view-controller design paradigm
- Given a scenario, determine if there is sufficient info to plan/design/build application

## App Components

- 3 Layers make up the App: User Interface, Business Logic and Data Model
- User Interface: Page Layouts (Page Builder) or Visualforce pages
- Business Logic: set of rules and calculations that handle information exchange between User Interface and Database. *Declarative Business Logic*: Queues, Workflows, Validation and Assignment Rules, Rollup summary fields, Cross Object fields.
- Data Model: Tables are related to each other, allowing the definition of complex data schemas. Data is queried via a SQL-like language called SOQL (Salesforce Object Query Language). Full-text search is available through SOSL (Salesforce Object Search Language).

## Model-View-Controller

- It dictates the separation of data, visual elements that represent data and actions to the user, and logic that mediates between the two. If these three areas are allowed to collide and the codebase grows large enough, the cost to fix bugs and add features becomes prohibitive
- Visualforce adopts MVC by design. For example, its *view* components do not allow the expression of *business logic* and vice versa



| Pattern | Force.com implementation |
|---|---|
| Model | Object (custom or standard) |
| View | VisualForce pages |
| Controller | Logic written in Apex that controls navigation |

## Plan/Design/Build

- Advantages of declarative customization: Ease of upgrades, Ease of maintenance, No requirement for programming knowledge
- System Fields:
  - Can be set through API (CreatedBy, CreatedDate, LastModifiedByID
  - You can edit on Obj: Account, Oppty, Contact, Lead, Case, Task, Event, Custom Objects
  - Must be enabled by Salesforce.com support
- Encrypted Fields:
  - "View Encrypted Data" Permission required
  - Contact SF to enable encrypted fields
  - 128 bit AES Encryption
  - Fields are limited to 175 Chars
  - SSL should be enabled

# Force.com Platform

- Steps required to build/deploy application declaratively
- Describe building blocks of application provided by Force.com platform

## Build / Deploy

- Force.com is multi-tenant: It can share hardware and other infrastructure
- In a custom application you can also include a custom logo and landing page
- **Custom Object**: business information that needs to be captured and it may or may not have business logic associated with it. They come with reporting, auditing, and access control and search-able. __c is added as a suffix to the custom object names
- Apps that leverage the API are known as composite apps
- Setup -> Development Tools:
  (1) Apex classes
  (2) API
  (3) Components
  (4) Email Services
  (5) Pages
  (6) S-controls / Visualforce
  (7) Static resources
  (8) Tools
  (9) Remote Access
- Setup > My Personal Information > Personal Information-> *Development Mode* checkbox
- Development Mode enables: *Splitscreen page editor, URL/apex/pagename page creator*
- A change set can be used to deploy metadata between related orgs (ALL or NOTHING if it fails)

## Force.com Building Blocks

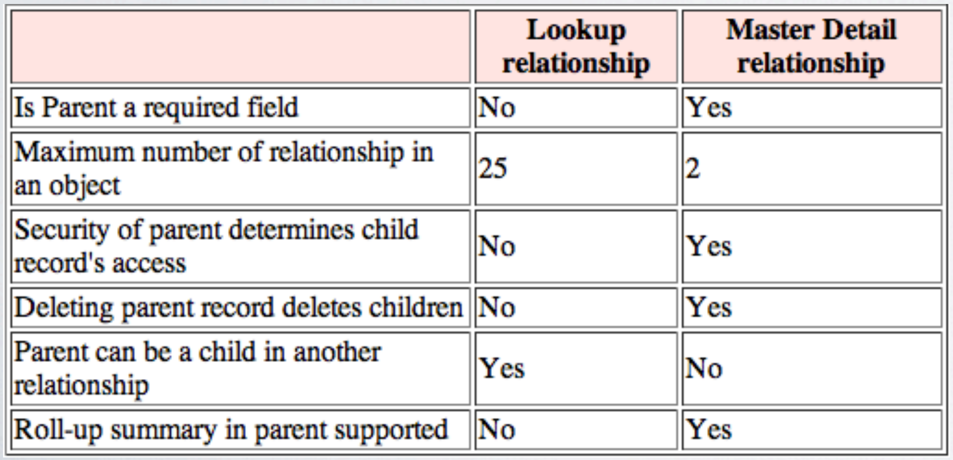
- The Building Blocks:
  - Applications, Tabs
  - Page Layouts, Record Types
  - Workflow, Validations Rules
- Types of Apps that can be build: *Data and Process centric applications are the best*.
- There are three type of page layouts: Detail, Mini, Console
- Objects have following fields that need to be entered by developers:
  - Label, Plural Label
  - Object Name, Description
  - Record Name, Record Type
- When customizing a page layout, you can change the field locations, page setting customizations, and related list customizations
- Items you can customize on Page Layout:
  - Related Lists
  - S-Controls
  - Buttons
  - Fields
  - Tab order

## Data Model

- Functional Data Model for business application with example scenario
- Object Relationships: choose pros/cons

### Functional Data Model

- *Objects* logically correspond to *database tables* in a relational database
- Dependent picklists: Standard picklists can be controlling fields but not dependent fields: Before defining a dependency, you should ensure that your picklist has at least one value
- You CANNOT create roll up summary fields based on lookup relationships, must be master-detail
- Type of field needs that needs to be wrapped in a function to be accessed: Picklist Values
- Custom fields can be made universally required
- Two types of records owners in Salesforce: Users & Queues
- Limit for cross-object formula unique relationships per object across all formulas and rules: **5**



| | Lookup relationship | Master Detail relationship |
|---|---|---|
| Is Parent a required field | No | Yes |
| Maximum number of relationship in an object | 25 | 2 |
| Security of parent determines child record's access | No | Yes |
| Deleting parent record deletes children | No | Yes |
| Parent can be a child in another relationship | Yes | No |
| Roll-up summary in parent supported | No | Yes |

### Object Relationships

- **1:M Relationship**: Lookup / Master-Detail, they are created as fields in the child record
- In **Master-Detail relationship**, if the parent record is deleted, then all its children records are deleted
- Child records in master-detail relationship do not have owners. They inherit ownership from the parent record
- In Master detail relationship, the parent field is required. *Also the parent field once specified cannot be changed*
- Standard Object cannot be on detail side of Master-Detail relationship
- Related list section of parent objects, only one field of the child object is displayed by default. This is the field specified as Record Name in the child object. To add more fields, search lookup layout needs to be updated
- The parent object can use roll-up summary field type to perform operations of sum, maximum, minimum, count among its children records
- **Many-to-Many relationships** are implemented using *two master-detail objects*. One Junction object is used as the child of the objects between which M:M relationship needs to be established
- **Self relationship** is a lookup relationship to itself. An example usage could be organization chart where an employee's manager is also an employee
- **Hierarchy Relationship** exists only for **User object**. In this, developers create a manager field in User object to relate to another object

# Data Model II

- Permission Sets and Data Access: appropriate features
- Optional Features of the Application: audit fields, encrypted fields, multi-currency

## Data Access

- Security, access, and deletion of child records are independent of the parent: **Lookup Relationship**
- A junction object is used to connect the two objects: **Many-to-Many relationship**
- Relationship can only be one layer deep: **Master-Detail Relationship**
- Access that can be granted through sharing rules: Read, Read/Write
- To ***transfer*** a record that a user does not own, the user needs to have the required user permissions and read sharing access on the record
- Public Groups are used to grant record level access to a dynamic group of users; for Calendars, Sync Profiles, Sharing Rules, Folder Access
- Users, Roles, Roles & Sub and Groups, Customer, Partner and Portal can be part of a Public Group
- 3 types of users that can grant *sharing privileges* on a given record: Record owner, System Admin, person higher in role hierarchy
- Field hidden by field-level security is NOT visible through the API
- Roles control record access. Profiles control everything else

## Optional Features

- ISCHANGED function (Validation): compares to a previous value and returns TRUE if value changed, else FALSE
- PRIORVALUE function (Validation): returns the previous value of a field
- ISNEW function (Validation): returns TRUE if new, FALSE if the record is simply being updated but already existed
- ISPICKVAL function (Validation): compares against a value in the picklist as part of additional logic (IF etc)
- VLOOKUP function (Validation): an object consisting of key value pair, returns value for a key
- REGEX function (Validation): compares formatting / regular expressions and returns TRUE if there is a match.
- Types of fields old and new values NOT tracked in Auditing: *Long Text Area* and *Multi-Select Picklists*.
- Visible + Read Only checked = Read-Only
- Visible checked – Editable
- Visible+Read Only both unchecked = Hidden
- Forecasting:
  - Direct Report – a person who reports to you in a role hierarchy
  - Subordinate – all your direct reports and all of their direct reports
  - Forecast History – an audit trail of your forecast submissions
  - “Override Forecast” permission required to override their own forecasts along with their direct reports forecasts

# User Interface

- UI Components of Force.com application: tabs, detail pages, list views
- Capabilities & Constraints of declarative framework
- Visualforce extension in the framework: *You cannot use both standard and custom controllers on the same page*
- Force.com sites

## UI Components

- Page Layout:
  - **Collapse**: collapse the palette if you want more room to drag and drop within the canvas
  - **Pallette**: Stays available as you scroll and scrolls automatically to expose valid drop targets
  - **Quick Find**: Filters the set of elements
  - **Properties**: Helps edit the related list properties
- Coarse metadata tags: Detail, Relatedlist, Listviews
- Data binding refer to the data set in the controller and use expression syntax
- The ***Page includes*** tag allow you to embed content within an iFrame in a web page
- Ajax is primarily used for partial page updates in VF
- Tab-order control affects direction you cycle through the fields using the Tab key

## Visualforce

- Visual Force: Tag-based markup language similar to HTML. *Hosted on SF platform*. Can consist of Tags, HTML, Javascript etc. VF tags are hierarchical and can include plain text (max ***15MB***).
- Visualforce Controllers: *<apex:page>* tag, set of instructions to specify what happens when a user interacts with components (e.g. button click)
- Visualforce Pages: Used to Override standard buttons, Override Tab overview pages, Define Custom tabs, Embed components in detail page layouts
- Access fields in VF with *{!object.Field}*, $User gives the details of the user: `{!$User.FirstName}` gives the first name
- All *custom component definitions* must be wrapped inside a single *<apex:component>* tag, it allows you to create your own custom, reusable components
- A single *Visualforce page*. All pages must be wrapped inside a single page component tag, *attributes to VF tags appear after the tag name in the start tag* (`attributeName="attributeValue"`)
- Standard Controller: `<:apex:page standardController="Account">`
  They are available for all API entities/objects, standard actions like save, edit , delete
- Custom Controllers: `<:apex:page controller="MyAccount">`
  They can be used to create wizards or leverage call outs.
- Controller Extensions: `<:apex:page standardController="Contact" extensions="ClassA, ClassB">`
  They are used to add custom behavior or additional data to controllers

## Force.com Sites

- Setup->Develop-->Sites
- Force.com sites are:
  - Public, unauthenticated websites
  - Accessed from branded domain names
  - Build with VF pages
  - From data and content in a Salesforce application
- Creating a Force.com Site requires:
  - Build a Force.com App
  - Design your web interface in visual force
  - Create a Force.com site
  - Activate the Force.com site
- Force.com Sites security:
  - Security for your site is controlled through public Access Settings
  - Access can also be restricted by specifying an IP range
- Global URLFOR variable: used to access actions available via the API on an object that are not currently exposed via the standard controllers

# Business Logic

- Apply business logic with: Formula Fields, validation rules and workflow for given scenario
- Force.com Approval Process
- Debugging and monitoring automated processes
- Extending business logic with Apex

## Business Logic

- Validation (attached to fields) rules are applied before the record is saved
- When writing a validation rule, developers must write the *error condition formula* and the *Error message*
- Workflow is a way to automate actions based on changes to a record
- Two methods for defining workflow criteria: Criteria or Formula
- Two parts of workflow: Criteria and Action
- Outbound Message are sent to external systems in XML format
- Cross-Object Formulas, we navigate from child object to parent or grand parent object(up to 5 levels). Field is displayed on child object.

## Approvals

- Create an approval process that skips steps by using a criteria based step that skips as step if the criteria is met.
- When approval processes have parallel approvals, developers select: First Response or Unanimous approval criteria
- Roll Up Summary Fields (RO), we navigate from Parent to Child object
- Formula fields do not store the value on the record but Roll Up Summary fields stores the value and changes can be seen on parent
- Cross Object Fields:
  - Can't reference cross object formula in roll up summary field
  - Cannot reference merge field for objects related to activities
  - Cannot reference record owner merge field for any object
  - **Limit to 5 per object**

## Debugging

- Security Controls->View Setup Audit Trail: Track Metadata changes
- By default 20 last configuration changes are displayed. Changes made in last 180 days can be exported into a csv file.
- **Field history tracking** can be enabled at object level. Up to 20 fields can be tracked for one object
- Custom Fields and Relationship area of Object configuration, "*Set History Tracking*", is used to enable field level tracking
- Debug logs track: Database changes, Resources used by an APEX script
- System Log: check the resource usage
- Debug Log: an issue with respect to approval process

## Apex

- Apex: Java like syntax language runs natively on platform. Can be stored as a Class or a Trigger (script) on the system. Class execute permissions are only checked at the top-level of Apex code
- The methods to create pages to support multiple mobile platforms:
  - Lowest common denominator
  - Conditional code
- Apex:**inputField**:
  - Automatically displays message next to fields
- Apex:**commandButton**:
  - Attaches an action to a button
- Apex:**messages**:
  - Displays messages at the top of the page
- Apex:**actionPoller**:
  - Calls actions periodically

# Data Management

- Import Wizard: capabilities and constraints
- API-based tools and Force.com Data Loader (GUI and command Line Interface
- External IDs with Upserts
- Force.com Record IDs

## Import Wizard

- Import wizard can be used to import accounts, contacts, leads, solutions and custom objects
- Import wizard can be used to load up to 50,000 records
- Import wizard can be used for de-duplication of data
- When loading data into a salesforce application dependencies between objects determine the order in which objects are loaded. If there is a one-to-many relationship between A and B, the data for A should be loaded before the data for B
- Junction object data CANNOT be imported with Import Wizard

## API & Data Loader

- The Extract operation uses a SOQL string to export a set of records from Salesforce.
- Is it NOT possible to leverage the upsert with relationships capability to load record owners
- Configure the Data Loader to run behind a proxy server by changing the *sfdc.endpoint* value
- If "Modifiable System Field" configuration is enabled, then system fields (like creation date) can be set to any value during initial data load
- Web Services API based tools can load more than 50,000 records and can support all object types
- Mass Transfer Tool: transfer multiple objects data from one user to another, require Transfer & Edit permission

## External IDs

- When importing custom objects, solutions, or person accounts, you can use external IDs (!= date field) to prevent duplicate records from being created as a result of the import operation
- A custom field that has the “External ID” attribute (3 per obj), contains unique record identifiers from a system outside of Salesforce. When you select this option, the import wizard will detect existing records in Salesforce that have the same external ID.
- This operation is NOT case-sensitive
- Advantage of performing Upserts: use the *Salesforce ID* OR *External ID* to prevent data duplication and create missing records
- Can be marked Unique but not required to be Unique

## Record IDs

- Can be either 18 digit case-insensitive or 15 digit case-sensitive
- Same ID value in Production as in Sandbox
- Ways in which you can obtain a Record’s ID:
  - URL
  - Reports
  - API
- Dangerous to work with the 15-digit case-sensitive form of the record ID - some external systems may not be case sensitive

# Reporting & Analytics

- Reporting
- Analytics

## Reporting

- Three types of reports: Tabular, Summary, Matrix.
- Reports will only save the report settings, not a snapshot of that data
- Two report folders to which all users have access by default: *My Personal* & *Unfiled Public*
- Dashboards comprised (maximum 20) components : Chart, Table, Gauge, Metric, S-Control, VF page
- To email a dashboard to someone, they must be a Salesforce user
- A user can see a specific dashboard if the user has access rights to the folder the dashboard is in
- Users with permission "*Manage Custom Report Types*" can create custom report types
- Custom report types provide: Modify Report Wizard interface, Without Joins, lookup relationship to other objects/fields: A standard report type is created upon creation of a custom object (automatically)
- The "**power of one**": A technique for counting unique records. Uses a custom formula field that is hidden from the page layout but used in reports and formula calculations: Formula = 1
- You can *use formula fields to create buckets to group the information* and make it easier to digest
- To email a report to other users, the report must be in a public folder
- Running user determines the data displayed on a dashboard
- Available dashboard layouts: Two column, Three Column

## Analytics

- Some **RaaS** and **BIaaS** partners. Reporting as a service (Jaspersoft, CrystalReports.com). Business Intelligence as a service (BIaaS - Lucidera, Cloud9, SuccessMetrics)
- Set up the snapshot: *Setup->Data Management*
- **Analytic snapshots** allow developers to create a *tabular/summary* report, then take a "snapshot" of the data on that report by *mapping the fields on the report to fields on a custom object*
- Users can report on the data in the custom object to view historical data and analyze trends
- To create an analytic snapshot: Create a source report, create a target object, set up the analytic snapshot (select the source report (*must be custom tabular report*), select the target object, map the fields, schedule the frequency)
- “With or without” reports show show all “A” whether or not there is a “B”
- Max objects in a CRT you can have when defining which related records from different objects: **4**