Deme Architecture

Todd Davies Mike Mintz

Stanford University

Silicon Valley Code Camp, 2008

Outline

Outline

Outline

What is an item?

- All persistent data abstracted as items
  - E.g., documents, users, settings, etc.
- Items are instances of “item types”
  - Item types specify structure of fields
  - Items provide values for fields
- E.g. “Person” is item type, “Mike” and “Todd” are items of that type

First Name	Last Name
Mike	Mintz
Todd	Davies

What is an item?

- All persistent data abstracted as items
  - E.g., documents, users, settings, etc.
- Items are instances of “item types”
  - Item types specify structure of fields
  - Items provide values for fields
- E.g. “Person” is item type, “Mike” and “Todd” are items of that type

First Name	Last Name
Mike	Mintz
Todd	Davies

What is an item?

- All persistent data abstracted as items
  - E.g., documents, users, settings, etc.
- Items are instances of “item types”
  - Item types specify structure of fields
  - Items provide values for fields
- E.g. “Person” is item type, “Mike” and “Todd” are items of that type

First Name	Last Name
Mike	Mintz
Todd	Davies

Item inheritance

- Item types can inherit from other item types
  - “Item” is at top of the hierarchy
  - E.g., Item -> Document -> Comment
- Item types inherit existing fields, add additional fields
  - Item: name, description
  - Document: name, description, body
  - Comment: name, description, body, commented_item

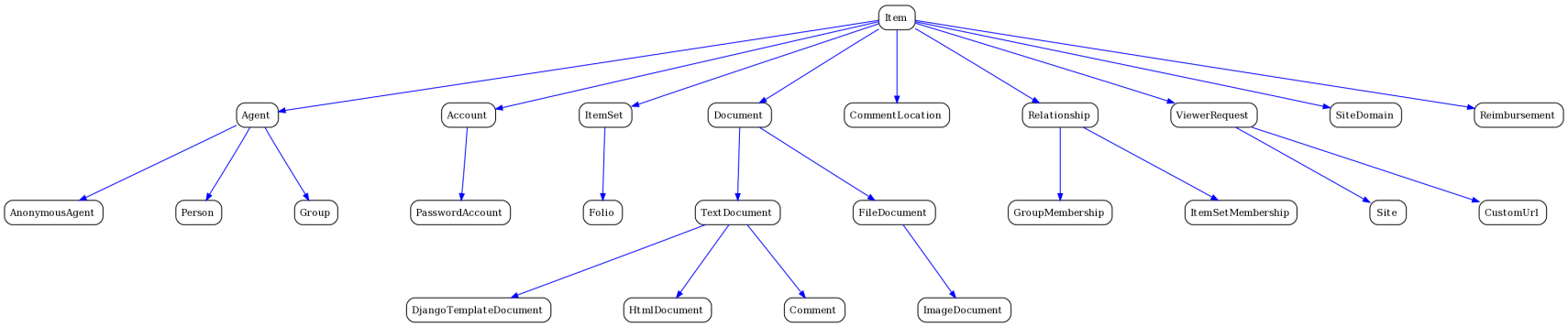
Item inheritance

- Item types can inherit from other item types
  - “Item” is at top of the hierarchy
  - E.g., Item -> Document -> Comment
- Item types inherit existing fields, add additional fields
  - Item: name, description
  - Document: name, description, body
  - Comment: name, description, body, commented_item

Outline

Overview

- Agents
- Collections
- Documents
- Relationships
- Permissions
- URLs



All Items

- All items share metadata structure:
  - Unique ID
  - Name
  - Description
  - Creator / Create Time
  - Updater / Update Time
  - Versions
  - Trashed?

Agents

- Agents
  - Item that can “do” things
    - (Given permissions, can appear as creator/updater)
  - Application user is *always* a specific Agent
  - Subclasses: Person, AnonymousAgent, Group
- Groups
  - Collection of Agents
  - Members can share common permissions
- Accounts
  - Channel of authentication
  - Subclasses: PasswordAccount, OpenId

Agents

- Agents
  - Item that can “do” things
    - (Given permissions, can appear as creator/updater)
  - Application user is *always* a specific Agent
  - Subclasses: Person, AnonymousAgent, Group
- Groups
  - Collection of Agents
  - Members can share common permissions
- Accounts
  - Channel of authentication
  - Subclasses: PasswordAccount, OpenId

Agents

- Agents
  - Item that can “do” things
    - (Given permissions, can appear as creator/updater)
  - Application user is *always* a specific Agent
  - Subclasses: Person, AnonymousAgent, Group
- Groups
  - Collection of Agents
  - Members can share common permissions
- Accounts
  - Channel of authentication
  - Subclasses: PasswordAccount, OpenId

Collections

- ItemSets
  - Arbitrary collection of items
  - Subclasses: Folio
- Folio
  - Every group has one folio
  - Contains items shared among all members

Collections

- ItemSets
  - Arbitrary collection of items
  - Subclasses: Folio
- Folio
  - Every group has one folio
  - Contains items shared among all members

Documents

- Main unit of collaborative work
- Different item types for different formats
- Subclasses
  - TextDocument
    - HtmlDocument
    - DjangoTemplateDocument
    - Comment
  - FileDocument
    - ImageDocument
- Comments refer to other Items
  - Sometimes specific versions and locations

Documents

- Main unit of collaborative work
- Different item types for different formats
- Subclasses
  - TextDocument
    - HtmlDocument
    - DjangoTemplateDocument
    - Comment
  - FileDocument
    - ImageDocument
- Comments refer to other Items
  - Sometimes specific versions and locations

Documents

- Main unit of collaborative work
- Different item types for different formats
- Subclasses
  - TextDocument
    - HtmlDocument
    - DjangoTemplateDocument
    - Comment
  - FileDocument
    - ImageDocument
- Comments refer to other Items
  - Sometimes specific versions and locations

Relationships

- Explicit items for relationships between items
- Subclasses: GroupMembership, ItemsetMembership

Permissions

- Specifies who can do what to which items
  - Agent
  - Ability
  - Item
- Three levels
  - Direct
  - Groupwide
  - Default
- Roles
  - Customizable sets of abilities

Permissions

- Specifies who can do what to which items
  - Agent
  - Ability
  - Item
- Three levels
  - Direct
  - Groupwide
  - Default
- Roles
  - Customizable sets of abilities

Permissions

- Specifies who can do what to which items
  - Agent
  - Ability
  - Item
- Three levels
  - Direct
  - Groupwide
  - Default
- Roles
  - Customizable sets of abilities

Custom URLs

- Users can create custom URLs to point to items
- Each domain can have separate namespace of URLs

Custom URLs

- Users can create custom URLs to point to items
- Each domain can have separate namespace of URLs

Outline

What is a viewer?

- A Django class that processes browser or API requests
- URLs that begin with /resource/ are routed to viewers
- Viewers handle items of specific type
- Viewers inherit from other viewers
  - Everything ultimately inherits from ItemViewer
  - ItemViewer is about 80% of view code

What is a viewer?

- A Django class that processes browser or API requests
- URLs that begin with /resource/ are routed to viewers
- Viewers handle items of specific type
- Viewers inherit from other viewers
  - Everything ultimately inherits from ItemViewer
  - ItemViewer is about 80% of view code

What is a viewer?

- A Django class that processes browser or API requests
- URLs that begin with /resource/ are routed to viewers
- Viewers handle items of specific type
- Viewers inherit from other viewers
  - Everything ultimately inherits from ItemViewer
  - ItemViewer is about 80% of view code

What is a viewer?

- A Django class that processes browser or API requests
- URLs that begin with /resource/ are routed to viewers
- Viewers handle items of specific type
- Viewers inherit from other viewers
  - Everything ultimately inherits from ItemViewer
  - ItemViewer is about 80% of view code

RESTful URLs

- Entry actions
  - /resource/<viewer>/<id>/<action>
  - E.g., /resource/person/5/edit
  - E.g., /resource/person/5
- Collection actions
  - /resource/<viewer>/<action>
  - E.g., /resource/person/new
  - E.g., /resource/person
  - E.g., /resource/person.json

RESTful URLs

- Entry actions
  - /resource/<viewer>/<id>/<action>
  - E.g., /resource/person/5/edit
  - E.g., /resource/person/5
- Collection actions
  - /resource/<viewer>/<action>
  - E.g., /resource/person/new
  - E.g., /resource/person
  - E.g., /resource/person.json

Authentication

- There is always an authenticated agent in a viewer
  - (If user does not log in, it is AnonymousAgent.)
- Viewer takes care of checking permissions for authenticated agent

Authentication

- There is always an authenticated agent in a viewer
  - (If user does not log in, it is AnonymousAgent.)
- Viewer takes care of checking permissions for authenticated agent

Deme is written in Django

- Deme is:
  - A collection of Django models
  - A system of hierarchical viewers that use Django templates
  - Additional components that handle Deme-specific features
    - Versioning
    - Permissions
    - Commenting

Why Django?

- Big developer community
- Now supports multi-table inheritance

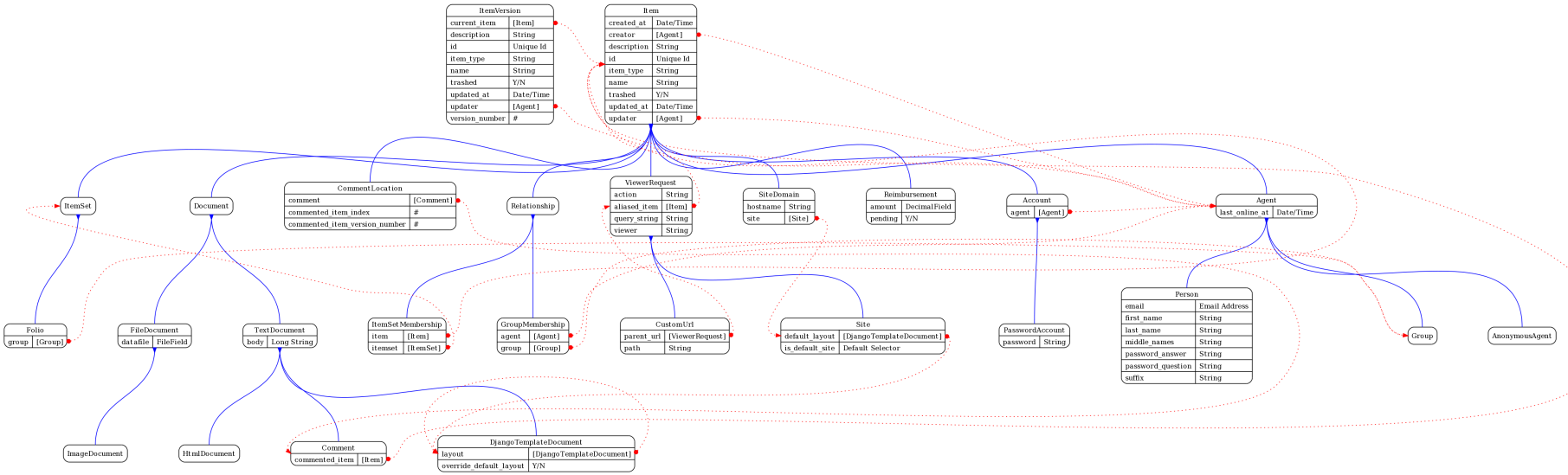
Models –> Item types

- A Django model corresponds to an item type
- An item is:
  - A Django model instance
  - A row in a table

Models –> Item types

- A Django model corresponds to an item type
- An item is:
  - A Django model instance
  - A row in a table

Database Visual



Versioning

- Every table has a dual version-table

Person

ID	First Name	Last Name
1	Mike	Mintz

PersonVersion

ID	Version	First Name	Last Name
1	1	Michael	Mintz
1	2	Mike	Mintz

Demo time!