Weblog and Content Management subsystem
Doubts: jpaweblogmanagerimpl and jpaweblogentrymanager not written

1. **Weblog (**Identity & Metadata,Configuration & Settings, Localization & Theme, Associations (Composition / Aggregation) **)**

   Attributes

   - id : String
   - handle : String
   - name : String
   - tagline : String
   - about : String
   - creator : String
   - dateCreated : Date
   - lastModified : Date
   - enableBloggerApi : Boolean
   - allowComments : Boolean
   - emailComments : Boolean
   - moderateComments : Boolean
   - defaultAllowComments : Boolean
   - defaultCommentDays : int
   - entryDisplayCount : int
   - visible : Boolean
   - active : Boolean
   - enableMultiLang : boolean
   - showAllLangs : boolean
   - analyticsCode : String
   - locale : String
   - timeZone : String
   - editorTheme : String
   - editorPage : String
   - iconPath : String
   - bloggerCategory : WeblogCategory
   - weblogCategories : List<WeblogCategory>
   - bookmarkFolders : List<WeblogBookmarkFolder>
   - mediaFileDirectories : List<MediaFileDirectory>
   - initializedPlugins : Map<String, WeblogEntryPlugin>

   Methods

   + getTheme() : weblogTheme
   + getId() : String
   + setId(String id)
   + getHandle() : String
   + setHandle(String handle)
   + getName() : String

+ setName(String Name)
+ getTagline() : String
+ setTagline(string tagline)
+ getCreator() : User
+ getCreatorUserName() : String
+ setCreatorUserName(String creatorUserName)
+ getEnableBloggerApi() : Boolean
+ setEnableBloggerApi(Boolean enableBloggerApi)
+ getBloggerCategory() : WeblogCategory
+ setBloggerCategory(WeblogCategory bloggerCategory)
+ getEditorPage() : String
+ setEditorPage(String editorPage)
+ getBannedwordslits() : String
+ setBannedwordslist(String bannedwordslist)
+ getAllowComments() : Boolean
+ setAllowComments(Boolean allowComments)
+ getDefaultAllowComments() : Boolean
+ setDefaultAllowComments(Boolean defaultAllowComments)
+ getDefaultCommentDays() : int
+ setDefaultCommentDays(int defaultCommentDays)
+ getModerateComments() : Boolean
+ setModerateComments(Boolean moderateComments)
+ getEmailComments() : Boolean
+ setEmailComments(Boolean emailComments)
+ getEmailAddress() : String
+ setEmailAddress(String emailAddress)
+ getEditorTheme() : String
+ setEditorTheme(String editorTheme)
+ getLocale() : String
+ setLocale(String locale)
+ getTimeZone() : String
+ setTimeZone(String timeZone)
+ getDateCreated() : Date
+ setDateCreated(final Date date)
+ getDefaultPlugins : String
+ setDefaultPlugins(String string)
+ setData(Weblog other)
+ getLocaleInstance() : Locale
+ getTimeZoneInstance() : TimeZone
+ hasUserPermission(User user, String action) : boolean
+ hasUserPermissions(User user, List<String> actions) : boolean
+ getEntryDisplayCount() : int
+ setEntryDisplayCount(int entryDisplayCount)
+ getVisible() : Boolean

+ setVisible(Boolean visible)
+ getActive() : Boolean
+ setActive(Boolean Active)
+ getCommentModerationRequired() : boolean
+ setCommentModerationRequired(boolean modRequired)
+ getLastModified() : Date
+ setLastModified(Date lastModified)
+ isEnableMultiLang() : boolean
+ setEnableMultiLang(boolean enableMultiLang)
+ isShowAllLangs() : boolean
+ setShowAllLangs(boolean showAllLangs)
+ getURL() : String
+ getAbsoluteURL() : String
+ getIconPath() : String
+ setIconPath(String iconPath)
+ getAnalyticsCode() : String
+ setAnalyticsCode(String analyticsCode)
+ getAbout() : String
+ setAbout(String about)
+ getInitializedPlugins() : Map<String, WeblogEntryPlugin>
+ getWeblogEntry(String anchor) : WeblogEntry
+ getWeblogCategory(String categoryName) : weblogCategory
+ getRecentWeblogEntries(String cat, int length) : List<WeblogEntry>
+ getRecentWeblogEntriesByTag(String tag, int length) : List<WeblogEntry>
+ getRecentComments(int length) : List<WeblogEntryComment>
+ getBookmarkFolder(String folderName) : weblogBookmarkFolder
+ getTodaysHits() : int
+ getPopularTags(int sinceDays, int length) : List<TagStat>
+ getCommentCount() : long
+ getEntryCount() : long
+ addCategory(weblogCategory category)
+ getWeblogCategories() : List<WeblogCategory>
+ setWeblogCategories(List<WeblogCategory> cats)
+ hasCategory(String name) : boolean
+ getBookmarkFolders() : List<WeblogBookmarkFolder>
+ setBookmarkFolders(List<WeblogBookmarkFolder> bookmarkFolders)
+ getMediaFileDirectories :List<MediaFileDirectory>
+ setMediaFileDirectories(List<MediaFileDirectory> mediaFileDirectories)
+ addBookmarkFolder(WeblogBookmarkFolder folder)
+ hasBookmarkFolder(String name) : boolean
+ hasMediaFileDirectory(String name) : boolean
+ getMediaFileDirectory(String name) : MediaFileDirectory

Role

- The Weblog class represents a blog/website entity in Apache Roller.
- It is the central domain model that holds configuration, metadata, and associations for a single blog.
- This class acts as the **root aggregate** for all blog-related content and is frequently used by service-layer managers during rendering, content fetching, and permission checking.

Interaction with Other Classes (need to confirm)

1. WeblogEntry - one to many
2. WeblogCategory - One to many
3. WeblogEntryManager - One-to-Many
4. **ThemeManager**
5. **PluginManager**
6. UserManager
7. WeblogBookmarkFolder - One-to-Many (Weblog owns bookmark folders)
8. MediaFileDirectory - One-to-Many (Weblog owns media directories)
9. User - Many-to-One (Many weblogs can have same creator)
10. WeblogPermission - One-to-Many (Weblog has many permission assignments)
11. WeblogTheme - Many-to-One (Many weblogs can use same theme)
12. WeblogEntryComment (**INDIRECT**)
13. WeblogEntryTag - One-to-Many (Weblog owns tags)
14. WeblogEntryPlugin

2. **User**

Attributes

- id : String
- userName : String
- password : String
- openIdUrl : String
- screenName : String
- fullName : String
- emailAddress : String
- dateCreated : Date
- locale : String
- timeZone : String
- enabled : Boolean
- activationCode : String

Methods

+ getId() : String
+ setId(id : String) : void

+ getUserName() : String
+ setUserName(userName : String) : void
+ getPassword() : String
+ setPassword(password : String) : void
+ resetPassword(newPassword : String) : void
+ getOpenIdUrl() : String
+ setOpenIdUrl(openIdUrl : String) : void
+ getScreenName() : String
+ setScreenName(screenName : String) : void
+ getFullName() : String
+ setFullName(fullName : String) : void
+ getEmailAddress() : String
+ setEmailAddress(email : String) : void
+ getDateCreated() : Date
+ setDateCreated(final Date date)
+ getLocale() : String
+ setLocale(locale : String) : void
+ getTimeZone() : String
+ setTimeZone(timeZone : String) : void
+ getEnabled() : Boolean
+ setEnabled(enabled : Boolean) : void
+ getActivationCode() : String
+ setActivationCode(code : String) : void
+ hasGlobalPermission(action : String) : boolean
+ hasGlobalPermissions(actions : List<String>) : boolean

Role

- User is a domain entity representing a registered user of the blogging system.

Interaction with other parts of subsystem

1. UserManager - Uses
2. Weblog - One to Many
3. **WeblogPermission** - Many to Many (User has permissions on multiple weblogs)
4. GlobalPermission -
5. WeblogEntry - One-to-Many (User creates multiple entries)

3. **WeblogTheme (**Inherited from Theme **Interface)**
   Attributes
   # weblog : Weblog

   Methods
   + getWeblog() : Weblog

   Role
- WeblogTheme represents a theme bound to a specific weblog and is a core abstraction used during rendering.
- This class is central to the rendering pipeline and is used whenever weblog content is transformed into HTML output.

   Interaction with Other classes
   1. Weblog - One to One
   2. Theme (**Interface**)

4. **WeblogTemplate (**Inherited from ThemeTemplate **Interface)**
   Attributes
   - id : String
   - action : ComponentType
   - name : String
   - description : String
   - link : String
   - lastModified : Date
   - hidden : boolean
   - navbar : boolean
   - outputContentType : String
   - weblog : Weblog
   - templateRenditions : List<CustomTemplateRendition>

   Methods
   + getId() : String
   + setId(String id)
   + getWeblog() : Weblog
   + setWeblog(Weblog website)
   + getAction() : ComponentType
   + setAction(ComponentType action)
   + getName() : String
   + setName(String name)
   + getDescription() : String
   + setDescription(String description)
   + getLink() : Strong

+ setLink(STring link)
+ getLastModified() : Date
+ setLastModified(final Date newtime)
+ isNavbar() : boolean
+ setNavbar(boolean navbar)
+ isHiddent() : boolean
+ setHidden(boolean isHidden)
+ getOutputContentType() : String
+ setOutputContentType(String outputContentType)
+ isRequired() : boolean
+ isCustom() : boolean
+ getTemplateRenditions() : List<CustomTemplateRendition>
+ setTemplateRenditions(List<CustomTemplateRendition> templateRenditions)
+ getTemplateRendition(CustomTemplateRendition.RenditionType desiredType) : CustomTemplateRendition
+ addTemplateRendition(CustomTemplateRendition newRendition)
+ hasTemplateRendition(customTemplateRendition proposed) : boolean

Role

- WeblogTemplate represents a single user-defined template page belonging to a specific weblog.
- It defines how weblog content is rendered into output (HTML, RSS, etc.).

Interaction with other parts of the subsystem

1. Weblog - (One weblog can have many weblogTemplate objects and Each weblogTemplate belongs to exactly one weblog) (**WHICH RELATION**)
2. CustomTemplateRendition
3. ThemeTemplate (**Interface**)

5. **WeblogBookmark**

Attributes

- id : String
- name : String
- description : String
- url : String
- priority : Integer
- image : String
- feedUrl : String
- folder : WeblogBookmarkFolder

Methods

+ getId() : String
+ setId(id : String)

+ getName() : String
+ setName(name : String)
+ getDescription() : String
+ setDescription(description : String)
+ getUrl() : String
+ setUrl(url : String)
+ getPriority() : Integer
+ setPriority(priority : Integer)
+ getImage() : String
+ setImage(image : String)
+ getFeedUrl() : String
+ setFeedUrl(feedUrl : String)
+ calculatePriority() : void
+ getFolder() : WeblogBookmarkFolder
+ setFolder(folder : WeblogBookmarkFolder)
+ getWebsite() : Weblog
+ compareTo(o : WeblogBookmark) : int

Role

- WeblogBookmark models a single bookmark (favorite link) stored in a weblog's bookmark collection.
- It's used to manage and display links to other websites, blogs, or resources that a blog owner wants to recommend or showcase.

Interaction with other parts of the subsystem

1. WeblogBookmarkFolder - Many to one (Many books belong to one folder)
2. Weblog - Many to one (**INDIRECT**)
3. BookmarkManager (**INTERFACE**)

6. **WeblogBookmarkFolder**

Attributes

- id : String
- name : String
- weblog : Weblog
- bookmarks : List<WeblogBookmark>

Methods

+ getId() : String
+ setId(String id)
+ getName() : String
+ setName(String name)
+ getWeblog() : Weblog
+ setWeblog(Weblog website)
+ getBookmarks() : List<WeblogBookmark>

+ setBookmarks(List<WeblogBookmark> bookmarks)
+ addBookmark(WeblogBookmark bookmark)
+ hasBookmarkOfName(String bookmarkName) : boolean
+ retrieveBookmarks() : List<WeblogBookmark>
+ compareTo(WeblogBookmarkFolder other) : int

Role

- WeblogBookmarkFolder represents a folder of bookmarks inside a Weblog in Apache Roller.
- It acts as a categorization mechanism for grouping related links together.

Interaction with other parts of the subsystem

1. WeblogBookmark - One folder has many bookmarks and each bookmark belongs to one folder.

2. Weblog - One weblog can have many BookmarkFolders and Each BookmarkFolder belongs to exactly one weblog.

3. BookmarkManager (**INTERFACE**).

7. **WeblogEntry**

Attributes (FOCUS ON THESE ATTRIBUTES ONLY - GPT**)**

- id : String
- title : String
- link : String
- summary : String
- text : String
- contentType : String
- contentSrc : String
- anchor : String
- pubTime : Timestamp
- updateTime : Timestamp
- plugins : String
- allowComments : Boolean
- commentDays : Integer
- rightToLeft : Boolean
- pinnedToMain : Boolean
- status : PubStatus
- locale : String
- creatorUserName : String
- searchDescription : String
- refreshAggregates : Boolean
- website : Weblog
- category : WeblogCategory
- attest : Set<WeblogEntryAttribute>
- tagSet : Set<WeblogEntryTag>

- removedTags : Set<WeblogEntryTag>
- addedTags : Set<WeblogEntryTag>

Methods
+ setData(WeblogEntry other)
+ getId() : String
+ setId(String id)
+ getCategory() : WeblogCategory
+ setCategory(WeblogCategory cat)
+ getCategories() : List<WeblogCategory>
+ getWebsite() : Weblog
+ setWebsite(Weblog website)
+ getCreator() : User
+ getCreatorUserName() : String
+ setCreatorUserName(String creatorUserName)
+ getTitle() : String
+ setTitle(String title)
+ getSummary() : String
+ setSummary(String summary)
+ getSearchDescription() : String
+ setSearchDescription(String searchDescription)
+ getText() : String
+ setText(String text)
+ getContentType() : String
+ setContentType(String contentType)
+ getContentSrc() : String
+ setContentSrc(String contentSrc)
+ getAnchor() : String
+ setAnchor(String anchor)
+ getEntryAttributes() : Set<WeblogEntryAttribute>
+ setEntryAttributes(Set<WeblogEntryAttribute> atts)
+ findEntryAttribute(String name) : String
+ putEntryAttribute(String name, String value)
+ getPubTime() : Timestamp
+ setPubTime(Timestamp pubTime)
+ getUpdateTime() : Timestamp
+ setUpdateTime(Timestamp updateTime)
+ getStatus() : PubStatus
+ setStatus(PubStatus status)
+ getLink() : String
+ setLink(String link)
+ getPlugins() : String
+ setPlugins(String string)
+ getAllowComments() : Boolean

+ setAllowComments(Boolean allowComments)
+ getCommentDays() : Integer
+ setCommentDays(Integer commentDays)
+ getRightToLeft() : Boolean
+ setRightToLeft(Boolean rightToLeft)
+ getPinnedToMain() : Boolean
+ setPinnedToMain(Boolean pinnedToMain)
+ getLocale() : String
+ setLocale(String locale)
+ getTags() : Set<WeblogEntryTag>
+ setTags(Set<WeblogEntryTag> tagSet)
+ addTag(String name)
+ getAddedTags() : Set<WeblogEntryTag>
+ getRemovedTags() : Set<WeblogEntryTag>
+ getTagsAsString() : String
+ setTagsAsString(String tags)
+ getCommentsStillAllowed() : boolean
+ setCommentsStillAllowed(boolean ignored)
+ formatPubTime(String pattern) : String
+ formatUpdateTime(String pattern) : String
+ getComments() : List<WeblogEntryComment>
+ getComments(boolean ignoreSpam, boolean approvedOnly) :
List<WeblogEntryComment>
+ getCommentCount() : int
+ getPermalink() : String
+ getPermaLink() : String
+ getCommentsLink() : String
+ getDisplayTitle() : String
+ createAnchor() : String
+ createAnchorBase() : String
+ setPermalink(String string)
+ setPermaLink(String string)
+ setDisplayTitle(String string)
+ getPluginsList() : List<String>
+ isDraft() : boolean
+ isPending() : boolean
+ isPublished() : boolean
+ getTransformedText() : String
+ getTransformedSummary() : String
+ hasWritePermissions(User user) : boolean
+ render(String str) : String
+ displayContent(String readMoreLink) : String
+ getDisplayContent() : String
+ getRefreshAggregates() : Boolean

+ setRefreshAggregates(Boolean refreshAggregates)

Role

- WeblogEntry represents a single blog post in Apache Roller.

Interaction with other classes in the subsystem

1. Weblog - Each entry belongs to exactly one weblog. (Weblog 1 -------- 0..* WeblogEntry)
2. WeblogCategory - Each entry belongs to one category. (WeblogCategory 1 -------- 0..* WeblogEntry)
3. User - (User 1 -------- 0..* WeblogEntry)
4. WeblogEntryTag - Each entry can have multiple tags. (WeblogEntry 1 -------- 0..* WeblogEntryTag)
5. WeblogEntryComment (WeblogEntry 1 -------- 0..* WeblogEntryComment)
6. WeblogEntryAttribute (WeblogEntry 1 -------- 0..* WeblogEntryAttribute)
7. PubStatus (ENUM)
8. WeblogEntryManager
9. WeblogEntryPlugin

8. **WeblogEntryAttribute**

Attributes

- id : String
- entry : WeblogEntry
- name : String
- value : String

Methods

+ getId() : String
+ setId(String Id)
+ getEntry() : WeblogEntry
+ setEntry(WeblogEntry entry)
+ getName() : String
+ setName(String name)
+ getValue() : String
+ setValue(String value)
+ compareTo(WeblogEntryAttribute att)

Role

- WeblogEntryAttribute represents a custom name–value metadata pair attached to a WeblogEntry

Interaction with other parts of the subsystem

1. WeblogEntry
    - One weblogEntry has many attributes

- Each attribute has exactly one entry.
- String ownership (Composition)
- WeblogEntry 1 ◆———— 0..* WeblogEntryAttribute

9. **WeblogEntrySearchCriteria**

Attributes

- weblog : Weblog
- user : User
- startDate : Date
- endDate : Date
- catName : String
- tags : List<String>
- status : PubStatus
- text : String
- sortBy : SortBy
- sortOrder : SortOrder
- locale : String

Methods

+ getWeblog() : Weblog
+ setWeblog(Weblog weblog)
+ getUser() : User
+ setUser(User user)
+ getStartDate() : Date
+ setStartDate(Date startDate)
+ getEndDate() : Date
+ setEndDate(Date endDate)
+ getCatName() : String
+ setCatName(String catName)
+ getTags() : List<String>
+ setTags(List<String> tags)
+ getStatus() : PubStatus
+ setStatus(PubStatus status)
+ getText() : String
+ setText(String text)
+ getSortBy() : SortBy
+ setSortBy(SortBy sortBy)
+ getSortOrder() : SortOrder
+ setSortOrder(SortOrder sortOrder)
+ getLocale() : String
+ setLocale(String locale)

Role

- WeblogEntrySearchCriteria is used to collect all conditions needed to search weblog entries

Interaction with other parts of the subsystem (NEED TO CHECK THE CARDINALITY)

1. Weblog (0..1)
2. User (0..1)
3. SortBy (ENUM)
4. SortOrder (ENUM)
5. PubStatus (ENUM)

10. WeblogEntryTag

Attributes

- log : Log
- id : String
- website: Weblog
- userName : String
- name : String
- time : Timestamp

Methods

+ getId() : String
+ setId(String id)
+ getWeblog() : Weblog
+ setWeblog(Weblog website)
+ getWeblogEntry() : WeblogEntry
+ setWeblogEntry(WeblogEntry data)
+ getUser() : User
+ getCreatorUserName() : String
+ setCreatorUserName(String userName)
+ getName() : String
+ setName(String name)
+ getTime() : Timestamp
+ setTime(Timestamp tagTime)

Role:

- WeblogEntryTag represents a *tag (keyword)* assigned to a specific weblog entry.
- It links a WeblogEntry, the Weblog, and the creator user, and stores when the tag was created.
- Its main role is to support tag-based organization, searching, and filtering of blog entries.

Interaction with other parts of the subsystem

1. Weblog
2. WeblogEntry (One weblog entry can have many tags and each tag is associated with exactly one weblog).

3. User (One tag is created by one user) {Multiplicity - 1} (**INDIRECT**).

11. **WeblogEntryTagAggregate**

Attributes

- Id : String
- name : String
- Website : Weblog
- lastUsed : Timestamp
- total : int

Methods

+ getId() : String
+ setId(String id)
+ getName() : String
+ setName(String name)
+ getLastUsed() : Timestamp
+ setLastUsed(Timestamp lastUsed)
+ getTotal() : int
+ setTotal(int total)

Role

- It represents aggregated tag statistics at the weblog level—for each tag, it stores how many times it is used (total) and when it was last used.
- This is mainly used for tag efficient tag listing without scanning individual entries.

Interaction with other parts of the subsystem

1. WeblogEntryTag
2. Weblog - (Each aggregate belongs to one weblog)

12. **WeblogEntryComment**

Attributes

- id : String
- name : String
- email : String
- url : String
- content : String
- postTime : Timestamp
- status : ApprovalStatus
- notify : Boolean
- remoteHost : String
- referrer : String
- userAgent : STring
- plugins : String

- contentType : String
- weblogEntry : WeblogEntry

Methods

+ getId() : String
+ setId(String id)
+ getName() : String
+ setName(String name)
+ getEmail() : String
+ setEmail(String email)
+ getWeblogEntry() : WeblogEntry
+ setWeblogEntry(WeblogEntry entry)
+ getUrl() : String
+ setUrl(String url)
+ getContent() : String
+ setContent(String content)
+ getPostTime() : Timestamp
+ setPostTime(Timestamp postTime)
+ getStatus() : ApprovalStatus
+ setStatus(ApprovalStatus status)
+ getNotify() : Boolean
+ setNotify(Boolean notify)
+ getRemoteHost() : String
+ setRemoteHost(STring remoteHost)
+ getReferrer() : String
+ setReferrer(String referrer)
+ getUserAgent() : String
+ setUserAgent(String userAgent)
+ getPlugins() : String
+ setPlugins(String plugins)
+ getContentType() : String
+ setContentType(String ContentType)
+ getPending() : Boolean
+ getApproved() : Boolean
+ getTimestamp() : String

Role

- It represents a user comment on a weblog entry, storing the comment's content, author details, posting time, and moderation status (approved, pending, spam).

Interaction with other parts of the subsystem

1. WeblogEntry - (Each weblogEntry has many comments and each comment belongs to exactly one webLogEntry).

2. ApprovalStatus (**ENUM**) - {APPROVED, DISAPPROVED, SPAM, PENDING}

13. **WeblogCategory**

Attributes

- id : String
- name : String
- description : String
- image : String
- position : int
- weblog : Weblog

Methods

+ calculatePosition() : void
+ getId() : String
+ setId(String id)
+ getName() : String
+ setName(String name)
+ getDescription() : String
+ setDescription(String description)
+ getPosition() : int
+ setPosition(int position)
+ setImage(String image)
+ getImage() : String
+ getWeblog() : Weblog
+ setWeblog(Weblog weblog)
+ retrieveWeblogEntries(boolean publishedOnly) : List<WeblogEntry>
+ isInUse() : boolean

Role

- It represents a classification/grouping mechanism for weblog entries, allowing blog posts to be organized, ordered, and displayed under meaningful categories.

Interactions with other parts of the subsystem

1. Weblog : Each category belongs to exactly one weblog; a weblog can have many categories.
2. WeblogEntry : Entries are associated with categories; categories retrieve entries using WeblogEntryManager and WeblogEntrySearchCriteria.

14. **MailProvider**

Attributes

- session : Session
- type : ConfigurationType
- mailHostName : String
- mailPort : int
- mailUsername : String
- mailPassword : String

Methods

+ getSession() : Session
+ getTransport() : Transport

Role

- MailProvider centralizes and manages email configuration and connectivity for Roller, creating and validating JavaMail Session and Transport objects used to send emails.

Interaction with other parts of the subsystem (NEED TO CONFIRM)

1.

15. **RendererManager**

Attributes

- rendererFactories : List<RendererFactory>

Methods

+ getRenderer(Template template, MobileDeviceRepository.DeviceType deviceType) : Renderer

Role

- RendererManager acts as a central dispatcher that selects the appropriate Renderer for a given Template and device type, enabling rendering mechanism.

Interaction with other parts of the subsystem

1. Renderer (**INTERFACE**)
2. Template
3. MobileDeviceRepository

16. WeblogManager (Interface)

Attributes
None. This is an interface.

Methods
addWeblog(newWebsite : Weblog) : void

saveWeblog(data : Weblog) : void

removeWeblog(website : Weblog) : void

getWeblog(id : String) : Weblog
getWeblogByHandle(handle : String) : Weblog
getWeblogByHandle(handle : String, enabled : Boolean) : Weblog
getWeblogs(enabled : Boolean, active : Boolean, startDate : Date, endDate : Date, offset : int, length : int) : List<Weblog>
getUserWeblogs(user : User, enabledOnly : boolean) : List<Weblog>
getWeblogUsers(weblog : Weblog, enabledOnly : boolean) : List<User>
getMostCommentedWeblogs(startDate : Date, endDate : Date, offset : int, length : int) : List<StatCount>
getWeblogHandleLetterMap() : Map<String, Long>
getWeblogsByLetter(letter : char, offset : int, length : int) : List<Weblog>
saveTemplate(data : WeblogTemplate) : void
removeTemplate(template : WeblogTemplate) : void
getTemplate(id : String) : WeblogTemplate
getTemplateByAction(w : Weblog, a : ComponentType) : WeblogTemplate
getTemplateByName(w : Weblog, p : String) : WeblogTemplate
getTemplateByLink(w : Weblog, p : String) : WeblogTemplate
saveTemplateRendition(templateCode : CustomTemplateRendition) : void
getTemplates(w : Weblog) : List<WeblogTemplate>

getWeblogCount() : long
release() : void

Role
The WeblogManager interface acts as the business service layer for weblog and template management. It defines the contract for creating, retrieving, updating, and deleting Weblog entities. It also supports querying weblogs by different criteria such as user ownership, activity state, and comment statistics. In addition, it manages the persistence and retrieval of WeblogTemplate objects and their custom renditions.

Interactions with other parts of the subsystem

1. **Weblog** - Dependency (WeblogManager 1 : N Weblog) - WeblogManager handles many Weblog instances
2. **User** - Dependency (WeblogManager 1 : N User) - WeblogManager interacts with many User instances
3. **WeblogTemplate** - Dependency (WeblogManager 1 : N WeblogTemplate) - WeblogManager handles many WeblogTemplate instances
4. **CustomTemplateRendition** - Dependency (WeblogManager 1 : N CustomTemplateRendition) - WeblogManager handles many CustomTemplateRendition instances
5. **StatCount** - Dependency (WeblogManager 1 : N StatCount) - WeblogManager produces many StatCount objects
6. **ComponentType** - Dependency (WeblogManager 1 : 1 ComponentType) - WeblogManager uses one ComponentType per method call

17. UserManager (interface)

Attributes
None (This is an Interface).

Methods
+ addUser(newUser : User) : void
+ saveUser(user : User) : void
+ removeUser(user : User) : void
+ getUserCount() : long
+ getUserByActivationCode(activationCode : String) : User
+ getUser(id : String) : User
+ getUserByUserName(userName : String) : User
+ getUserByUserName(userName : String, enabled : Boolean) : User
+ getUserByOpenIdUrl(openIdUrl : String) : User
+ getUsers(enabled : Boolean, startDate : Date, endDate : Date, offset : int, length : int) : List<User>
+ getUsersStartingWith(startsWith : String, enabled : Boolean, offset : int, length : int) : List<User>
+ getUserNameLetterMap() : Map<String, Long>
+ getUsersByLetter(letter : char, offset : int, length : int) : List<User>
+ checkPermission(perm : RollerPermission, user : User) : boolean
+ grantWeblogPermission(weblog : Weblog, user : User, actions : List<String>) : void
+ grantWeblogPermissionPending(weblog : Weblog, user : User, actions : List<String>) : void
+ confirmWeblogPermission(weblog : Weblog, user : User) : void
+ declineWeblogPermission(weblog : Weblog, user : User) : void

+ revokeWeblogPermission(weblog : Weblog, user : User, actions : List<String>) : void
+ getWeblogPermissions(user : User) : List<WeblogPermission>
+ getPendingWeblogPermissions(user : User) : List<WeblogPermission>
+ getWeblogPermissions(weblog : Weblog) : List<WeblogPermission>
+ getPendingWeblogPermissions(weblog : Weblog) : List<WeblogPermission>
+ getWeblogPermissionsIncludingPending(weblog : Weblog) : List<WeblogPermission>
+ getWeblogPermission(weblog : Weblog, user : User) : WeblogPermission
+ getWeblogPermissionIncludingPending(weblog : Weblog, user : User) : WeblogPermission
+ grantRole(roleName : String, user : User) : void
+ revokeRole(roleName : String, user : User) : void
+ hasRole(roleName : String, user : User) : boolean
+ getRoles(user : User) : List<String>
+ release() : void

Role in Subsystem
The UserManager interface defines the contract for managing the lifecycle of system users and their authorization. In the context of the Weblog and Content Subsystem, it acts as the gatekeeper and relationship manager. It handles the creation and retrieval of User entities (authors) and manages WeblogPermission objects, which link a User to a Weblog with specific rights (e.g., ability to post entries, manage comments). Without this class, the system cannot determine ownership or authorship of content.

Interactions with Other Classes

1. **User** - Dependency (UserManager 1 : N User) - UserManager creates, saves, retrieves, and manages many User instances
2. **Weblog** - Dependency (UserManager 1 : N Weblog) - UserManager manages permissions associating users with many Weblogs
3. **WeblogPermission** - Dependency (UserManager 1 : N WeblogPermission) - UserManager creates and retrieves many WeblogPermission objects linking users to weblogs
4. **RollerPermission** - Dependency (UserManager 1 : 1 RollerPermission) - UserManager uses one RollerPermission per permission check operation

18. **WeblogEntryManager (interface)**
No attributes

Methods

saveWeblogEntry(entry : WeblogEntry) : void

removeWeblogEntry(entry : WeblogEntry) : void

getWeblogEntry(id : String) : WeblogEntry

getWeblogEntryByAnchor(website : Weblog, anchor : String) : WeblogEntry

getWeblogEntries(wesc : WeblogEntrySearchCriteria) : List<WeblogEntry>

getWeblogEntryObjectMap(wesc : WeblogEntrySearchCriteria) : Map<Date, List<WeblogEntry>>

getWeblogEntryStringMap(wesc : WeblogEntrySearchCriteria) : Map<Date, String>

getMostCommentedWeblogEntries(website : Weblog, startDate : Date, endDate : Date, offset : int, length : int) : List<StatCount>

getNextEntry(current : WeblogEntry, catName : String, locale : String) : WeblogEntry

getPreviousEntry(current : WeblogEntry, catName : String, locale : String) : WeblogEntry

getWeblogEntriesPinnedToMain(max : Integer) : List<WeblogEntry>

removeWeblogEntryAttribute(name : String, entry : WeblogEntry) : void

saveWeblogCategory(cat : WeblogCategory) : void

removeWeblogCategory(cat : WeblogCategory) : void

getWeblogCategory(id : String) : WeblogCategory

moveWeblogCategoryContents(srcCat : WeblogCategory, destCat : WeblogCategory) : void

getWeblogCategoryByName(website : Weblog, categoryName : String) : WeblogCategory

getWeblogCategories(website : Weblog) : List<WeblogCategory>

saveComment(comment : WeblogEntryComment) : void

removeComment(comment : WeblogEntryComment) : void

getComment(id : String) : WeblogEntryComment

getComments(csc : CommentSearchCriteria) : List<WeblogEntryComment>

removeMatchingComments(website : Weblog, entry : WeblogEntry, searchString : String, startDate : Date, endDate : Date, status : ApprovalStatus) : int

createAnchor(data : WeblogEntry) : String

isDuplicateWeblogCategoryName(data : WeblogCategory) : boolean

isWeblogCategoryInUse(data : WeblogCategory) : boolean

applyCommentDefaultsToEntries(website : Weblog) : void

release() : void

getPopularTags(website : Weblog, startDate : Date, offset : int, limit : int) : List<TagStat>

getTags(website : Weblog, sortBy : String, startsWith : String, offset : int, limit : int) : List<TagStat>

getTagComboExists(tags : List<String>, weblog : Weblog) : boolean

getHitCount(id : String) : WeblogHitCount

getHitCountByWeblog(weblog : Weblog) : WeblogHitCount

getHotWeblogs(sinceDays : int, offset : int, length : int) : List<WeblogHitCount>

saveHitCount(hitCount : WeblogHitCount) : void

removeHitCount(hitCount : WeblogHitCount) : void

incrementHitCount(weblog : Weblog, amount : int) : void

resetAllHitCounts() : void

resetHitCount(weblog : Weblog) : void

getCommentCount() : long

getCommentCount(websiteData : Weblog) : long

getEntryCount() : long

getEntryCount(websiteData : Weblog) : long

Role

The WeblogEntryManager acts as the business service interface for the content subsystem. It abstracts the persistence layer logic required to manage the core artifacts of a blog: entries (posts), categories (organization),

comments (user interaction), tags (metadata), and hit counts (analytics). It ensures data integrity (e.g., checking for duplicate categories) and provides complex retrieval methods (e.g., searching entries by criteria, getting popular tags).

Interactions with other parts of the subsystem

1. **WeblogEntry** - Dependency (WeblogEntryManager 1 : N WeblogEntry) - WeblogEntryManager performs CRUD operations on many WeblogEntry objects
2. **Weblog** - Dependency (WeblogEntryManager 1 : N Weblog) - WeblogEntryManager uses many Weblogs to scope and filter query operations
3. **WeblogCategory** - Dependency (WeblogEntryManager 1 : N WeblogCategory) - WeblogEntryManager creates, removes, retrieves, and moves content between many WeblogCategory objects
4. **WeblogEntryComment** - Dependency (WeblogEntryManager 1 : N WeblogEntryComment) - WeblogEntryManager persists and moderates many WeblogEntryComment objects
5. **WeblogEntrySearchCriteria** - Dependency (WeblogEntryManager 1 : N WeblogEntrySearchCriteria) - WeblogEntryManager uses many criteria objects to encapsulate complex filtering options for entry queries
6. **CommentSearchCriteria** - Dependency (WeblogEntryManager 1 : N CommentSearchCriteria) - WeblogEntryManager uses many criteria objects to filter comment retrieval operations
7. **WeblogHitCount** (not in this subsystem in any part) - Dependency (WeblogEntryManager 1 : N WeblogHitCount) - WeblogEntryManager updates and retrieves many hit count statistics for weblogs
8. **TagStat** - Dependency (WeblogEntryManager 1 : N TagStat) - WeblogEntryManager calculates and returns many tag usage statistics as TagStat objects
9. **StatCount** (not in this subsystem in any part) - Dependency (WeblogEntryManager 1 : N StatCount) - WeblogEntryManager calculates and returns many statistical objects for metrics like most commented entries

19. Weblogger

(Interface)

Methods

+ getUserManager() : UserManager

+ getBookmarkManager() : BookmarkManager

+ getOAuthManager() : OAuthManager

+ getWeblogManager() : WeblogManager

+ getWeblogEntryManager() : WeblogEntryManager

+ getAutopingManager() : AutoPingManager

+ getPingTargetManager() : PingTargetManager

+ getPingQueueManager() : PingQueueManager

+ getPropertiesManager() : PropertiesManager

+ getThreadManager() : ThreadManager

+ getIndexManager() : IndexManager

+ getThemeManager() : ThemeManager

+ getPluginManager() : PluginManager

+ getMediaFileManager() : MediaFileManager

+ getFileContentManager() : FileContentManager

+ getUrlStrategy() : URLStrategy

+ flush() : void

+ release() : void

+ initialize() : void

+ shutdown() : void

+ getVersion() : String

+ getRevision() : String

+ getBuildTime() : String

+ getBuildUser() : String

+ getFeedFetcher() : FeedFetcher

+ getPlanetManager() : PlanetManager

+ getPlanetURLStrategy() : PlanetURLStrategy

Role

The Weblogger interface acts as the Facade and Service Locator for the entire Weblogger business tier. It aggregates all specific domain managers (such as WeblogManager, WeblogEntryManager, MediaFileManager) and provides a single, unified entry point for the application to access these services. It is responsible for the initialization, lifecycle management, and resource release of the business layer. In the context of the "Weblog and Content Subsystem," it is the primary interface through which the subsystem's components are retrieved and utilized.

Interactions

1. **WeblogManager** - Association/Dependency (Weblogger 1 : 1 WeblogManager) - Weblogger provides access to WeblogManager for weblog creation and management
2. **WeblogEntryManager** - Association/Dependency (Weblogger 1 : 1 WeblogEntryManager) - Weblogger provides access to WeblogEntryManager for managing entries, comments, and categories
3. **ThemeManager** - Association/Dependency (Weblogger 1 : 1 ThemeManager) - Weblogger provides access to ThemeManager for rendering engine themes and templates
4. **FeedFetcher** (not in this subsystem) - Association/Dependency (Weblogger 1 : 1 FeedFetcher) - Weblogger provides access to FeedFetcher for Planet aggregator content fetching
5. **MediaFileManager** - Association/Dependency (Weblogger 1 : 1 MediaFileManager) - Weblogger provides access to MediaFileManager for managing media uploads and directories
6. **UserManager** (not in this subsystem) - Association/Dependency (Weblogger 1 : 1 UserManager) - Weblogger provides access to UserManager for user accounts and permissions
7. **BookmarkManager** (not in this subsystem) - Association/Dependency (Weblogger 1 : 1 BookmarkManager) - Weblogger provides access to BookmarkManager for managing blogrolls and bookmarks
8. **PluginManager** - Association/Dependency (Weblogger 1 : 1 PluginManager) - Weblogger provides access to PluginManager for managing content plugins
9. **IndexManager** (not in this subsystem) - Association/Dependency (Weblogger 1 : 1 IndexManager) - Weblogger provides access to IndexManager for search indexing operations
10. **FileContentManager** (not in this subsystem) - Association/Dependency (Weblogger 1 : 1 FileContentManager) - Weblogger provides access to FileContentManager for low-level file storage operations

20. Tagstat

Attributes

- serialVersionUID : long

- name : String

- count : int

- intensity : int

Methods

+ TagStat()

+ getName() : String

+ setName(String name) : void

+ getCount() : int

+ setCount(int count) : void

+ toString() : String

+ getIntensity() : int

+ setIntensity(int intensity) : void

Role

The TagStat class acts as a Data Transfer Object (DTO) or a value object representing the usage statistics of a specific tag within a weblog or across the entire site.

- It encapsulates the tag's name.
- It holds the count (frequency) of how many times the tag has been used on blog entries.
- It holds an intensity value, which is typically calculated by the business logic (Manager) to determine the visual weight (e.g., font size) of the tag when rendered in a UI "Tag Cloud."

Interactions with other parts of the subsystem

1. **WeblogEntryManager** - Dependency (WeblogEntryManager 1 : N TagStat) - WeblogEntryManager calculates tag statistics and creates/returns TagStat objects via methods like getPopularTags() and getTags()
2. **java.io.Serializable** - Realization (TagStat 1 : 1 Serializable) - TagStat implements Serializable interface for object serialization in caching and network transmission

22. PluginManager (interface)

Methods

hasPagePlugins() : boolean

getWeblogEntryPlugins(website : Weblog) : Map<String, WeblogEntryPlugin>

applyWeblogEntryPlugins(pagePlugins : Map<String, WeblogEntryPlugin>, entry : WeblogEntry, str : String) : String

getCommentPlugins() : List

applyCommentPlugins(comment : WeblogEntryComment, text : String) : String

release() : void

Role

The PluginManager interface defines the contract for managing the lifecycle and application of content plugins within the Weblogger system. It acts as a bridge between the core content objects (Weblogs, Entries, Comments) and the extensible plugin architecture. Its primary role is to retrieve configured plugins and apply them to text content, transforming raw input into render-ready HTML (e.g., converting line breaks, processing smileys, or sanitizing malicious code).

Interactions with other parts of the subsystem

1. **Weblog** - Dependency (PluginManager 1 : 1 Weblog) - PluginManager uses Weblog parameter in getWeblogEntryPlugins to identify website context for site-specific plugin retrieval
2. **WeblogEntryPlugin** - Dependency (PluginManager 1 : N WeblogEntryPlugin) - PluginManager returns and orchestrates multiple WeblogEntryPlugin interfaces in getWeblogEntryPlugins and applyWeblogEntryPlugins to transform entry text
3. **WeblogEntry** - Dependency (PluginManager 1 : 1 WeblogEntry) - PluginManager uses WeblogEntry parameter in applyWeblogEntryPlugins to provide context for plugin rendering process
4. **WeblogEntryCommentPlugin** - Dependency (PluginManager 1 : N WeblogEntryCommentPlugin) - PluginManager returns multiple WeblogEntryCommentPlugin interfaces in getCommentPlugins for processing comment text
5. **WeblogEntryComment** - Dependency (PluginManager 1 : 1 WeblogEntryComment) - PluginManager uses WeblogEntryComment

parameter in applyCommentPlugins to determine applicable plugins and provide transformation context

23. PluginManagerImpl

Attributes (private)

-log : Log

-mPagePlugins : Map<String, Class<? extends WeblogEntryPlugin>>

-commentPlugins : List<WeblogEntryCommentPlugin>

Methods

+PluginManagerImpl()

+hasPagePlugins() : boolean

+getWeblogEntryPlugins(website : Weblog) : Map<String, WeblogEntryPlugin>

+applyWeblogEntryPlugins(pagePlugins : Map<String, WeblogEntryPlugin>, entry : WeblogEntry, str : String) : String

+getCommentPlugins() : List<WeblogEntryCommentPlugin>

+applyCommentPlugins(comment : WeblogEntryComment, text : String) : String

-loadPagePluginClasses() : void

-loadCommentPlugins() : void

+release() : void

Role
PluginManagerImpl is the concrete implementation of the PluginManager interface within the Weblog and Content Subsystem. Its primary role is to act as the registry and execution engine for content transformers. It loads plugin configurations (both for blog entries and comments) from the system properties, instantiates them using reflection, and orchestrates their application to text content. It ensures that raw text from entries or comments is processed through the configured pipeline of plugins (e.g., formatting, syntax highlighting) and sanitized before being rendered to the user.

Interactions with other parts of the subsystem

1. **PluginManager** - Realization (PluginManagerImpl 1 : 1 PluginManager) - PluginManagerImpl implements PluginManager interface providing concrete plugin management logic
2. **WeblogEntryPlugin** - Association (PluginManagerImpl 1 : N WeblogEntryPlugin) - PluginManagerImpl maintains static map of entry plugin classes, instantiates them via reflection in getWeblogEntryPlugins, and iterates through instances in applyWeblogEntryPlugins
3. **WeblogEntryCommentPlugin** - Aggregation (PluginManagerImpl 1 : N WeblogEntryCommentPlugin) - PluginManagerImpl maintains list of instantiated comment plugin objects and iterates over them in applyCommentPlugins
4. **Weblog** - Dependency (PluginManagerImpl 1 : 1 Weblog) - PluginManagerImpl passes Weblog to each WeblogEntryPlugin's init() method in getWeblogEntryPlugins for blog-specific initialization
5. **WeblogEntry** - Dependency (PluginManagerImpl 1 : 1 WeblogEntry) - PluginManagerImpl passes WeblogEntry to plugins in applyWeblogEntryPlugins to provide rendering context
6. **WeblogEntryComment** - Dependency (PluginManagerImpl 1 : 1 WeblogEntryComment) - PluginManagerImpl checks WeblogEntryComment for enabled plugins and provides execution context in applyCommentPlugins
7. **WebloggerConfig** - Dependency (PluginManagerImpl 1 : 1 WebloggerConfig) - PluginManagerImpl uses WebloggerConfig in loadPagePluginClasses to retrieve plugins.page property containing plugin class names
8. **HTMLSanitizer** - Dependency (PluginManagerImpl 1 : 1 HTMLSanitizer) - PluginManagerImpl calls HTMLSanitizer.conditionallySanitize in applyWeblogEntryPlugins to sanitize final output after plugin processing

24. WeblogEntryPlugin (interface)

Methods

+ getName() : String

+ getDescription() : String

+ init(weblog : Weblog) : void

+ render(entry : WeblogEntry, str : String) : String

Role

The WeblogEntryPlugin interface defines the contract for plugins that perform text transformations on weblog entries. It acts as a hook in the rendering pipeline, allowing the system to modify the content of a blog post (summary or body) dynamically. This is used for features like converting line breaks to

HTML, parsing wiki syntax, handling emoticons, or applying other formatting rules before the content is served to the reader.

Interactions with other parts of the subsystem

1. **Weblog** - Dependency (WeblogEntryPlugin 1 : 1 Weblog) - WeblogEntryPlugin receives Weblog parameter in init() method for weblog-specific context and settings initialization
2. **WeblogEntry** - Dependency (WeblogEntryPlugin 1 : 1 WeblogEntry) - WeblogEntryPlugin receives WeblogEntry parameter in render() method to access entry details during text transformation
3. **WebloggerException** (doubt if to be added in this subsystem) - Dependency (WeblogEntryPlugin 1 : 1 WebloggerException) - WeblogEntryPlugin throws WebloggerException from init() method when initialization fails

25. WeblogEntryCommentPlugin (interface)

Methods

getId() : String

getName() : String

getDescription() : String

render(comment : WeblogEntryComment, str : String) : String

Role
This interface defines the contract for plugins that manipulate, transform, or validate the text of a weblog entry comment. It serves as an abstraction layer allowing the system to apply various processors (such as spam filters, HTML sanitizers, or text formatters) to comments dynamically without modifying the core comment logic.

Interactions with other parts of the subsystem

1. **WeblogEntryComment** - Dependency (WeblogEntryCommentPlugin 1 : 1 WeblogEntryComment) - WeblogEntryCommentPlugin receives WeblogEntryComment parameter in render() method to inspect comment metadata for text transformation decisions

26. CommentSearchCriteria

Attributes

- weblog : Weblog

- entry : WeblogEntry

- searchText : String

- startDate : Date

- endDate : Date

- status : ApprovalStatus

- reverseChrono : boolean

- offset : int

- maxResults : int

Methods

+ getWeblog() : Weblog

+ setWeblog(weblog : Weblog) : void

+ getEntry() : WeblogEntry

+ setEntry(entry : WeblogEntry) : void

+ getSearchText() : String

+ setSearchText(searchText : String) : void

+ getStartDate() : Date

+ setStartDate(startDate : Date) : void

+ getEndDate() : Date

+ setEndDate(endDate : Date) : void

+ getStatus() : ApprovalStatus

+ setStatus(status : ApprovalStatus) : void

+ isReverseChrono() : boolean

+ setReverseChrono(reverseChrono : boolean) : void

+ getOffset() : int

+ setOffset(offset : int) : void

+ getMaxResults() : int

+ setMaxResults(maxResults : int) : void

Role
This class functions as a Search Specification or Parameter Object. Its primary role is to decouple the method signatures of the business layer from the specific criteria used to query comments. Instead of passing 9 different arguments to a search method (which would be messy and hard to maintain), the system passes a single instance of CommentSearchCriteria. It allows the application to filter comments based on the blog they belong to, the specific entry, the content text, date ranges, or approval status, while also handling pagination (offset/limit) and sorting.

Interactions with other parts of the subsystem

1. **Weblog** - Association (CommentSearchCriteria 1 : 1 Weblog) - CommentSearchCriteria holds optional Weblog reference to restrict search results to specific weblog; null applies search to all weblogs
2. **WeblogEntry** - Association (CommentSearchCriteria 1 : 1 WeblogEntry) - CommentSearchCriteria holds optional WeblogEntry reference to narrow search to comments on specific blog post
3. **WeblogEntryComment.ApprovalStatus** - Dependency (CommentSearchCriteria 1 : 1 ApprovalStatus) - CommentSearchCriteria uses optional ApprovalStatus enum to filter comments by workflow state (PENDING, APPROVED, SPAM)
4. **Date** - Dependency (CommentSearchCriteria 1 : 2 Date) - CommentSearchCriteria uses optional Date objects (startDate and endDate) to define temporal boundaries for search query

27. GlobalCommentManagement

Attributes

- log : Log

- COUNT : int = 30

- bean : GlobalCommentManagementBean

- pager : CommentsPager

- firstComment : WeblogEntryComment

- lastComment : WeblogEntryComment

- bulkDeleteCount : int = 0

- httpMethod : String = "GET"

Methods

+ GlobalCommentManagement()

+ requiredGlobalPermissionActions() : List<String>

+ isWeblogRequired() : boolean

+ loadComments() : void

- buildBaseUrl() : String

+ execute() : String

+ query() : String

+ delete() : String

+ update() : String

+ getCommentStatusOptions() : List<KeyValueObject>

+ getBean() : GlobalCommentManagementBean

+ setBean(bean : GlobalCommentManagementBean) : void

+ getBulkDeleteCount() : int

+ setBulkDeleteCount(bulkDeleteCount : int) : void

+ getFirstComment() : WeblogEntryComment

+ setFirstComment(firstComment : WeblogEntryComment) : void

+ getLastComment() : WeblogEntryComment

+ setLastComment(lastComment : WeblogEntryComment) : void

+ getPager() : CommentsPager

+ setPager(pager : CommentsPager) : void

+ setServletRequest(req : HttpServletRequest) : void

Role
This class serves as the Controller (Struts2 Action) for the global administration of weblog comments. It allows administrators to search, view, approve, mark as spam, or delete comments across the entire system. It acts as the bridge between the administrative user interface and the backend business logic (WeblogEntryManager) that persists comment data.

Interactions with other parts of the subsystem

1. **UIAction** - Inheritance (GlobalCommentManagement 1 : 1 UIAction) - GlobalCommentManagement extends UIAction to inherit Struts2 action capabilities like menu handling and title setting
2. **ServletRequestAware** - Realization (GlobalCommentManagement 1 : 1 ServletRequestAware) - GlobalCommentManagement implements ServletRequestAware to access HttpServletRequest for HTTP method determination
3. **GlobalCommentManagementBean** - Association/Composition (GlobalCommentManagement 1 : 1 GlobalCommentManagementBean) - GlobalCommentManagement uses GlobalCommentManagementBean as DTO to bind UI form data and search criteria
4. **WeblogEntryManager** - Dependency (GlobalCommentManagement 1 : 1 WeblogEntryManager) - GlobalCommentManagement uses WeblogEntryManager for querying, removing, and saving comments
5. **WeblogEntryComment** - Association (GlobalCommentManagement 1 : N WeblogEntryComment) - GlobalCommentManagement holds references to comment entities for display logic and processes lists during updates
6. **CommentsPager** - Association (GlobalCommentManagement 1 : 1 CommentsPager) - GlobalCommentManagement instantiates CommentsPager to handle pagination logic for comment list display
7. **WebloggerFactory** - Dependency (GlobalCommentManagement 1 : 1 WebloggerFactory) - GlobalCommentManagement uses WebloggerFactory statically to retrieve WeblogEntryManager and UrlStrategy instances
8. **CommentSearchCriteria** - Dependency (GlobalCommentManagement 1 : 1 CommentSearchCriteria) - GlobalCommentManagement instantiates CommentSearchCriteria to encapsulate search filters when calling WeblogEntryManager
9. **Weblog** - Dependency (GlobalCommentManagement 1 : N Weblog) - GlobalCommentManagement accesses Weblogs via comments to identify which weblog caches need invalidation after updates
10. **CacheManager** - Dependency (GlobalCommentManagement 1 : 1 CacheManager) - GlobalCommentManagement uses CacheManager statically to invalidate weblog caches when comments are modified

28. CommentAuthenticator (interface)

Methods

+ getHtml(request : HttpServletRequest) : String

+ authenticate(request : HttpServletRequest) : boolean

Role
This interface defines the contract for comment authentication plugins within the subsystem. It acts as a strategy pattern, allowing the weblog engine to

plug in different mechanisms (such as Math CAPTCHA, LDAP, or simple pass-through) to verify a user's identity or humanity before a comment is accepted. It is responsible for generating the necessary HTML for the authentication challenge and for verifying the user's response during the comment submission process.

Interactions with other parts of the subsystem

1. **HttpServletRequest** - Dependency (CommentAuthenticator 1 : 1 HttpServletRequest) - CommentAuthenticator uses HttpServletRequest parameter to access form parameters, session attributes, and locale for authentication widget generation and credential verification
2. **DefaultCommentAuthenticator** - Realization (DefaultCommentAuthenticator 1 : 1 CommentAuthenticator) - DefaultCommentAuthenticator implements CommentAuthenticator to provide default no-op or pass-through authentication behavior
3. **MathCommentAuthenticator** - Realization (MathCommentAuthenticator 1 : 1 CommentAuthenticator) - MathCommentAuthenticator implements CommentAuthenticator to challenge users with math problems for spam prevention
4. **LdapCommentAuthenticator** - Realization (LdapCommentAuthenticator 1 : 1 CommentAuthenticator) - LdapCommentAuthenticator implements CommentAuthenticator to authenticate commenters against external LDAP directory service

29. CommentValidator (interface)

Methods
getName() : String

validate(WeblogEntryComment comment, RollerMessages messages) : int

Role
It defines the contract for comment validation plugins within the subsystem. Implementations of this interface are used to inspect WeblogEntryComment objects for validity (e.g., checking for spam, banned words, or excessive links) and report errors via RollerMessages. It allows the comment validation logic to be extensible and pluggable.

Interactions with other parts of the subsystem

1. **WeblogEntryComment** - Dependency (CommentValidator 1 : 1 WeblogEntryComment) - CommentValidator receives WeblogEntryComment parameter in validate() method to inspect content, author, and URL for validation criteria

2. **RollerMessages** - Dependency (CommentValidator 1 : 1 RollerMessages) - CommentValidator receives RollerMessages parameter in validate() method to add error messages when comment fails validation
3. **CommentValidationManager** - Association/Aggregation (CommentValidationManager 1 : N CommentValidator) - CommentValidationManager manages CommentValidator lifecycle, iterates through configured validators, and invokes their validate methods

30. CommentValidationManager

Attributes

- log : Log

- validators : List<CommentValidator>

Methods

+ CommentValidationManager()

+ addCommentValidator(val : CommentValidator) : void

+ validateComment(comment : WeblogEntryComment, messages : RollerMessages) : int

Role in Subsystem
The CommentValidationManager acts as the central orchestrator for comment validation. Its primary role is to manage a collection of pluggable CommentValidator implementations (such as spam checkers, size limiters, or banned word checkers). When a new comment is submitted, this manager iterates through all configured validators to assess the comment's validity, aggregating confidence scores and collecting error messages.

Interactions with Other Classes

1. **CommentValidator** - Aggregation (CommentValidationManager 1 : N CommentValidator) - CommentValidationManager maintains list of CommentValidator instances and iterates through them invoking validate() method in validateComment
2. **WeblogEntryComment** - Dependency (CommentValidationManager 1 : N WeblogEntryComment) - CommentValidationManager receives WeblogEntryComment parameter in validateComment for manager and validators to inspect state for validity
3. **RollerMessages** - Dependency (CommentValidationManager 1 : N RollerMessages) - CommentValidationManager receives RollerMessages

parameter in validateComment where validators add error messages for UI feedback

4. **Reflection** - Dependency (CommentValidationManager 1 : 1 Reflection) - CommentValidationManager uses Reflection utility in constructor to dynamically instantiate CommentValidator objects from configuration property class names

31.Renderer

Methods

+ render(model : Map<String, Object>, writer : Writer) : void

Role

The Renderer interface acts as an abstraction layer for the rendering mechanism in the Weblogger subsystem. Its primary role is to take a data model (a map of objects) and a template, and generate the final output (typically writing to a Writer). By defining this interface, the system allows for pluggable rendering technologies (such as Velocity or other template engines) without tightly coupling the core logic to a specific implementation.

Interactions with other parts of the subsystem

1. **PageServlet** - Dependency (PageServlet 1 : 1 Renderer) - PageServlet retrieves Renderer instance per request and calls render() method to generate HTTP response content for weblog page
2. **RendererManager** - Dependency (RendererManager 1 : 1 Renderer) - RendererManager locates and returns appropriate Renderer implementation for given template and device type
3. **RendererFactory** - Dependency (RendererFactory 1 : N Renderer) - RendererFactory implementations instantiate specific Renderer implementations
4. **RenderingException** - Dependency (Renderer 1 : 1 RenderingException) - Renderer throws RenderingException from render() method when rendering errors occur

32. VelocityRenderer

Attributes

- log : Log {static, readOnly}

- renderTemplate : Template {readOnly}

- deviceType : MobileDeviceRepository.DeviceType {readOnly}

- velocityTemplate : org.apache.velocity.Template

- velocityDecorator : org.apache.velocity.Template

- velocityException : Exception

Methods

+ VelocityRenderer(template : Template, deviceType : MobileDeviceRepository.DeviceType)

+ render(model : Map<String, Object>, out : Writer) : void

- renderException(model : Map<String, Object>, out : Writer, template : String) : void

Role
The VelocityRenderer class acts as the concrete implementation of the rendering engine specifically for Velocity templates within the Weblog and Content Subsystem. Its primary role is to take a generic Template object and a data model, and use the Apache Velocity engine to merge them, producing the final textual output (typically HTML) to a Writer. It handles the complexity of looking up the correct template rendition based on the device type, managing decorators, and rendering specific error pages if the Velocity engine encounters exceptions.

Interactions with other parts of the subsystem (DOUBT)

- Renderer (Interface)
  - Exact Relationship: Realization
  - Cardinality: 1
  - Reason: VelocityRenderer implements the Renderer interface, fulfilling the contract for rendering content in the system.
- Template (Interface)
  - Exact Relationship: Association
  - Cardinality: 1
  - Reason: The renderer holds a direct reference to the Template object (renderTemplate) it is responsible for rendering.
- RollerVelocity

33. TemplateRendition

Attributes
(Note: As an interface, these are properties implied by the accessor methods. The inner Enums defined within the file also contain attributes.)

TemplateRendition (Interface Properties)

- template : String

- templateLanguage : TemplateLanguage
- type : RenditionType

TemplateLanguage (Inner Enum)

- readableName : String

Methods
TemplateRendition

+ getTemplate() : String

+ getTemplateLanguage() : TemplateLanguage

+ getType() : RenditionType

+ setTemplate(template : String) : void

+ setTemplateLanguage(templateLanguage : TemplateLanguage) : void

+ setType(type : RenditionType) : void

TemplateLanguage (Inner Enum)

+ getReadableName() : String

Role
The TemplateRendition interface represents a specific version of a weblog template tailored for a particular context. Its primary role is to encapsulate the actual template code (source) along with metadata describing its language (e.g., Velocity) and its intended target device or output format (defined by RenditionType, such as STANDARD or MOBILE). This allows the rendering engine to dynamically select and render the appropriate template based on the user's device or request context.

Interactions:

Only enums relations, no specific relationships from this

34. CustomTemplateRendition

Attributes
Identification

- {static} serialVersionUID : long

- id : String

Associations

- weblogTemplate : WeblogTemplate

Content & Metadata

- template : String

- type : RenditionType

- templateLanguage : TemplateLanguage

Methods

+ CustomTemplateRendition(template : WeblogTemplate, type : RenditionType)

+ CustomTemplateRendition()

+ getWeblogTemplate() : WeblogTemplate

+ setWeblogTemplate(weblogTemplate : WeblogTemplate) : void

+ getId() : String

+ setId(id : String) : void

+ getTemplate() : String

+ setTemplate(template : String) : void

+ getType() : RenditionType

+ setType(type : RenditionType) : void

+ toString() : String

+ equals(other : Object) : boolean

+ hashCode() : int

+ getTemplateLanguage() : TemplateLanguage

+ setTemplateLanguage(templateLanguage : TemplateLanguage) : void

Role in Subsystem
The CustomTemplateRendition class represents a specific implementation (rendition) of a weblog template. In the rendering engine, a single "Template" (like "Weblog Standard") might need different code depending on the device accessing it (e.g., a Standard version vs. a Mobile version) or the language used. This class stores that specific template code string and links it to the parent WeblogTemplate. It interacts with the rest of the subsystem by

providing the raw content that the template engine parses to generate HTML for the user.

Interactions with other parts of the subsystem

1. **WeblogTemplate** - Association/Aggregation (WeblogTemplate 1 : N CustomTemplateRendition) - CustomTemplateRendition receives WeblogTemplate in constructor, calls addTemplateRendition(this) to establish bidirectional link, and stores reference in weblogTemplate field
2. **RenditionType** - Association (RenditionType 1 : N CustomTemplateRendition) - CustomTemplateRendition receives RenditionType in constructor and stores in type field for rendering engine to determine device/context-specific rendition
3. **TemplateLanguage** - Association (TemplateLanguage 1 : N CustomTemplateRendition) - CustomTemplateRendition stores TemplateLanguage in templateLanguage field to identify template code syntax for rendering engine parser selection
4. **TemplateRendition** - Realization (CustomTemplateRendition 1 : 1 TemplateRendition) - CustomTemplateRendition implements TemplateRendition interface providing concrete implementations of interface methods

35. PageModel

Attributes

-log: Log (static)

-pageRequest: WeblogPageRequest

-urlStrategy: URLStrategy

-commentForm: WeblogEntryCommentForm

-requestParameters: Map<String, String[]>

-weblog: Weblog

-deviceType: DeviceType

Methods

+PageModel()

+getModelName(): String

+init(initData: Map<String, Object>): void

+getLocale(): String

+getWeblog(): WeblogWrapper

+isPermalink(): boolean

+isSearchResults(): boolean

+getWeblogEntry(): WeblogEntryWrapper

+getWeblogPage(): ThemeTemplateWrapper

+getWeblogCategory(): WeblogCategoryWrapper

+getTags(): List<String>

+getDeviceType(): String

+getWeblogEntriesPager(): WeblogEntriesPager

+getWeblogEntriesPager(catArgument: String): WeblogEntriesPager

+getWeblogEntriesPagerByTag(tagArgument: String): WeblogEntriesPager

+getCommentForm(): WeblogEntryCommentForm

+getRequestParameter(paramName: String): String

-getWeblogEntriesPager(catArgument: String, tagArgument: String): WeblogEntriesPager

Role in Subsystem
The PageModel class is a crucial component in the content rendering process. Its primary role is to act as a data bridge between the incoming web request and the view template (e.g., a JSP or Velocity template). It gathers all necessary data for a specific page request-such as the weblog itself, the specific entry or category being viewed, date information, and request parameters and exposes this data to the rendering engine through a clean and organized API. It encapsulates the logic for determining what content to display (e.g., a single post, a page of posts for a specific day, or the latest posts).

Interactions with Other Classes

1. **Model** - Realization (PageModel 1 : 1 Model) - PageModel implements Model interface providing getModelName() and init() methods for polymorphic handling by rendering framework
2. **WeblogPageRequest** - Association (PageModel 1 : 1 WeblogPageRequest) - PageModel receives and stores WeblogPageRequest in init() containing parsed URL details for fetching correct content

3. **URLStrategy** - Association (PageModel 1 : 1 URLStrategy) - PageModel uses URLStrategy to generate site-specific URLs for entities when wrapping them for rendered page links
4. **Weblog** - Association (PageModel 1 : 1 Weblog) - PageModel holds reference to specific Weblog for which page is rendered to access properties and associated content
5. **WeblogEntriesPager** - Dependency (PageModel 1 : 1 WeblogEntriesPager) - PageModel acts as factory in getWeblogEntriesPager methods, instantiating specific pager implementation based on request details
6. **WeblogWrapper, WeblogEntryWrapper, WeblogCategoryWrapper, ThemeTemplateWrapper** - Dependency (PageModel 1 : N Wrappers) - PageModel wraps raw POJOs using Decorator pattern before returning from methods to add URL generation functionality
7. **WeblogEntryCommentForm** - Association (PageModel 1 : 1 WeblogEntryCommentForm) - PageModel receives optional WeblogEntryCommentForm in init() containing submitted comment data and validation errors for template re-display

36. WeblogPageRequest

Attributes
Static Attributes

- log : Log

- PAGE_SERVLET : String

Lightweight Attributes (URL Parsing Results)

- context : String

- weblogAnchor : String

- weblogPageName : String

- weblogCategoryName : String

- weblogDate : String

- tags : List<String>

- pageNum : int

- customParams : Map<String, String[]>

Heavyweight Attributes (Domain Objects)

- weblogEntry : WeblogEntry

- weblogPage : ThemeTemplate

- weblogCategory : WeblogCategory

Page Hits (State Flags)

- websitePageHit : boolean

- otherPageHit : boolean

Methods

+ WeblogPageRequest()

+ WeblogPageRequest(request : HttpServletRequest)

~ isValidDestination(servlet : String) : boolean

- isValidDateString(dateString : String) : boolean

+ getContext() : String

+ setContext(context : String) : void

+ getWeblogAnchor() : String

+ setWeblogAnchor(weblogAnchor : String) : void

+ getWeblogPageName() : String

+ setWeblogPageName(weblogPage : String) : void

+ getWeblogCategoryName() : String

+ setWeblogCategoryName(weblogCategory : String) : void

+ getWeblogDate() : String

+ setWeblogDate(weblogDate : String) : void

+ getPageNum() : int

+ setPageNum(pageNum : int) : void

+ getCustomParams() : Map<String, String[]>

+ setCustomParams(customParams : Map<String, String[]>) : void

+ getTags() : List<String>

+ setTags(tags : List<String>) : void

+ getWeblogEntry() : WeblogEntry

+ setWeblogEntry(weblogEntry : WeblogEntry) : void

+ getWeblogPage() : ThemeTemplate

+ setWeblogPage(weblogPage : WeblogTemplate) : void

+ getWeblogCategory() : WeblogCategory

+ setWeblogCategory(weblogCategory : WeblogCategory) : void

+ isWebsitePageHit() : boolean

+ setWebsitePageHit(websitePageHit : boolean) : void

+ isOtherPageHit() : boolean

+ setOtherPageHit(otherPageHit : boolean) : void

Role
This class acts as a Request Parser and Context Provider for the rendering engine. It translates a raw HttpServletRequest and its URL path (e.g., /entry/my-post, /date/20230101) into a structured object containing the specific domain objects (Entry, Category, Page) required to render the view. It encapsulates the logic for decoding URL structures and lazy-loading the corresponding database entities.

Interactions with other parts of the subsystem

1. **WeblogRequest** (doubts) - Inheritance (WeblogPageRequest 1 : 1 WeblogRequest) - WeblogPageRequest extends WeblogRequest to inherit weblog handle and locale parsing before performing page-specific parsing
2. **HttpServletRequest** - Dependency (WeblogPageRequest 1 : 1 HttpServletRequest) - WeblogPageRequest receives HttpServletRequest in constructor to extract servlet path, path info, and query parameters
3. **WeblogEntry** - Association/Aggregation (WeblogPageRequest 1 : 1 WeblogEntry) - WeblogPageRequest lazy-loads optional WeblogEntry using WeblogEntryManager when getWeblogEntry() is called
4. **ThemeTemplate** (doubt) - Association/Aggregation (WeblogPageRequest 1 : 1 ThemeTemplate) - WeblogPageRequest retrieves optional ThemeTemplate from Weblog's theme when getWeblogPage() is called
5. **WeblogCategory** - Association/Aggregation (WeblogPageRequest 1 : 1 WeblogCategory) - WeblogPageRequest lazy-loads optional WeblogCategory using WeblogEntryManager when getWeblogCategory() is called
6. **WeblogEntryManager** - Dependency (WeblogPageRequest 1 : 1 WeblogEntryManager) - WeblogPageRequest uses WeblogEntryManager in getWeblogEntry() and getWeblogCategory() to fetch data objects from backend

7. **WebloggerFactory** - Dependency (WeblogPageRequest 1 : 1 WebloggerFactory) - WeblogPageRequest uses WebloggerFactory statically to obtain WeblogEntryManager instance

37. WeblogFeedRequest

Attributes

Static Attributes

- {static} log : Log

- {static} FEED_SERVLET : String

Lightweight Attributes

- type : String

- format : String

- weblogCategoryName : String

- tags : List<String>

- page : int

- excerpts : boolean

- term : String

Heavyweight Attributes

- weblogCategory : WeblogCategory

Methods

+ WeblogFeedRequest()

+ WeblogFeedRequest(request : HttpServletRequest)

+ getType() : String

+ setType(type : String) : void

+ getFormat() : String

+ setFormat(format : String) : void

+ getWeblogCategoryName() : String

+ setWeblogCategoryName(weblogCategory : String) : void

+ getTags() : List<String>

+ setTags(tags : List<String>) : void

+ isExcerpts() : boolean

+ setExcerpts(excerpts : boolean) : void

+ getWeblogCategory() : WeblogCategory

+ setWeblogCategory(weblogCategory : WeblogCategory) : void

+ getPage() : int

+ setPage(page : int) : void

+ getTerm() : String

+ setTerm(query : String) : void

Role

The WeblogFeedRequest class acts as a specialized request parser and context holder for Roller weblog feeds (e.g., RSS, Atom). Its primary role is to interpret the URL structure (e.g., /roller-ui/rendering/feed/handle/type/format) and query parameters (like categories, tags, or search terms) to determine exactly what subset of weblog data is being requested. It encapsulates this criteria so the rendering engine can fetch the appropriate entries.

Interactions with other parts of the subsystem

1. **WeblogRequest** (doubts) - Inheritance (WeblogFeedRequest 1 : 1 WeblogRequest) - WeblogFeedRequest extends WeblogRequest to inherit weblog handle and locale identification, adding feed-specific parameter logic
2. **WeblogCategory** - Association (WeblogFeedRequest 1 : 1 WeblogCategory) - WeblogFeedRequest holds optional WeblogCategory reference in weblogCategory attribute for feed filtering based on parsed weblogCategoryName
3. **WeblogEntryManager** - Dependency (WeblogFeedRequest 1 : 1 WeblogEntryManager) - WeblogFeedRequest uses WeblogEntryManager in getWeblogCategory() to retrieve WeblogCategory object from database
4. **HttpServletRequest** - Dependency (WeblogFeedRequest 1 : 1 HttpServletRequest) - WeblogFeedRequest receives HttpServletRequest in constructor to parse ServletPath, PathInfo, and request parameters
5. **WebloggerFactory** - Dependency (WeblogFeedRequest 1 : 1 WebloggerFactory) - WeblogFeedRequest uses WebloggerFactory statically in getWeblogCategory() to obtain WeblogEntryManager instance

38.WeblogFeedRequest

Attributes

log : Log -> private

SEARCH_SERVLET : String-> public

query : String-> private

pageNum : int-> private

weblogCategoryName : String-> private

weblogCategory : WeblogCategory-> private

Methods (all public)

WeblogSearchRequest()

WeblogSearchRequest(request : HttpServletRequest)

getQuery() : String

setQuery(query : String) : void

getPageNum() : int

setPageNum(pageNum : int) : void

getWeblogCategoryName() : String

setWeblogCategoryName(weblogCategory : String) : void

getWeblogCategory() : WeblogCategory

setWeblogCategory(weblogCategory : WeblogCategory) : void

Role
The WeblogSearchRequest class acts as a specialized data transfer object and request parser for the weblog search functionality. Its primary role is to validate that an incoming HTTP request is intended for the search servlet and to extract relevant search parameters (such as the search query q, the page number page, and the category filter cat). It encapsulates this logic so that the rendering engine can easily access the search criteria without parsing the raw HttpServletRequest again.

Interactions with other parts of the subsystem

1. **WeblogRequest** (doubt) - Inheritance (WeblogSearchRequest 1 : 1 WeblogRequest) - WeblogSearchRequest extends WeblogRequest via super(request) to inherit weblog handle and locale parsing

2. **HttpServletRequest** - Dependency (WeblogSearchRequest 1 : 1 HttpServletRequest) - WeblogSearchRequest receives HttpServletRequest in constructor to retrieve servlet path and extract query parameters (q, page, cat)
3. **WeblogCategory** - Association (WeblogSearchRequest 1 : 1 WeblogCategory) - WeblogSearchRequest maintains optional WeblogCategory reference for search category context, lazy-loaded in getWeblogCategory()
4. **WeblogEntryManager** - Dependency (WeblogSearchRequest 1 : 1 WeblogEntryManager) - WeblogSearchRequest uses WeblogEntryManager in getWeblogCategory() to retrieve WeblogCategory entity from database by category name
5. **WebloggerFactory** - Dependency (WeblogSearchRequest 1 : 1 WebloggerFactory) - WeblogSearchRequest uses WebloggerFactory.getWeblogger() in getWeblogCategory() to obtain Weblogger instance for accessing WeblogEntryManager
6. **URLUtilities** - Dependency (WeblogSearchRequest 1 : 1 URLUtilities) - WeblogSearchRequest uses URLUtilities.decode() statically in constructor to decode cat parameter from URL
7. **Log** - Association (WeblogSearchRequest 1 : 1 Log) - WeblogSearchRequest uses static Log instance for debug messages during parsing and error messages on category lookup failures
8. **InvalidRequestException** - Dependency (WeblogSearchRequest 1 : N InvalidRequestException) - WeblogSearchRequest throws InvalidRequestException from constructor when request path is invalid or not bound to search servlet

39. Pager (interface)

Methods:

+ getHomeLink(): String
+ getHomeName(): String
+ getNextLink(): String
+ getNextName(): String
+ getPrevLink(): String
+ getPrevName(): String
+ getItems(): List<T>

Role in Subsystem

The Pager<T> interface establishes a generic and reusable contract for pagination within the Roller Weblogger application. Its primary role is to provide a standardized way for the rendering engine to handle collections of objects (like weblog entries, comments, users, etc.) that are too large to be

displayed on a single page. It decouples the presentation layer (the templates) from the business logic required to fetch subsets of data and create navigation links, ensuring a consistent user experience for pagination across different types of content.

Interactions with other parts of the subsystem

1. **AbstractPager** - Realization (AbstractPager 1 : 1 Pager) - AbstractPager implements Pager interface providing foundational logic for creating Home, Next, and Previous links based on page number
2. **Concrete Pager Classes (e.g., CommentsPager, UsersPager, WeblogsPager, WeblogEntriesListPager)** - Realization (Concrete Pagers N : 1 Pager via AbstractPager) - Concrete pager classes extend AbstractPager to implement getItems() for specific data types and inherit link generation methods
3. **Rendering Engine (e.g., PageServlet, Velocity/JSP Templates)** - Association/Dependency (Rendering Engine 1 : 1 Pager) - Rendering components use Pager instance per request to display items via getItems() and navigation controls via getNextLink()/getPrevLink()

40. WeblogEntriesPager (interface)

Methods

getEntries() : Map<Date, ? extends Collection>

getHomeLink() : String

getHomeName() : String

getNextLink() : String

getNextName() : String

getPrevLink() : String

getPrevName() : String

getNextCollectionLink() : String

getNextCollectionName() : String

getPrevCollectionLink() : String

getPrevCollectionName() : String

Role
The WeblogEntriesPager interface acts as a contract for the rendering mechanism to navigate and display lists of weblog entries. It abstracts the

logic required to paginate through content (e.g., "Next Page", "Previous Page") and navigate between logical collections (e.g., "Next Month", "Previous Day"). It ensures that the rendering layer receives entries grouped by date, facilitating the display of chronological blog posts regardless of the underlying data source (search results, category archives, or main weblog feeds).

Interactions with other parts of the subsystem

1. **WeblogEntryWrapper** - Dependency (WeblogEntriesListPager 1 : N WeblogEntryWrapper) - WeblogEntriesListPager manages and returns collections of WeblogEntryWrapper objects via getEntries() method for template display
2. **Date** - Dependency (WeblogEntriesListPager 1 : N Date) - WeblogEntriesListPager uses Date as map key in getEntries() to organize wrapped entries chronologically by day

41. WelogEntriesPermalinkPager

Attributes

- static final Log log

- WeblogEntry currEntry

- WeblogEntry nextEntry

- WeblogEntry prevEntry

- Map<Date, List<WeblogEntryWrapper>> entries

Methods

+ WeblogEntriesPermalinkPager(URLStrategy strat, Weblog weblog, String locale, String pageLink, String entryAnchor, String dateString, String catName, List<String> tags, int page)

+ getEntries() : Map<Date, List<WeblogEntryWrapper>>

+ getHomeLink() : String

+ getHomeName() : String

+ getNextLink() : String

+ getNextName() : String

+ getPrevLink() : String

+ getPrevName() : String

- getNextEntry() : WeblogEntry

- getPrevEntry() : WeblogEntry

Role

The WeblogEntriesPermalinkPager class is responsible for managing the display of a single weblog entry, typically when a user accesses an entry directly via its permalink. It fetches the specific entry identified by its anchor and provides navigation links to the chronologically next and previous published entries within the same weblog. This class is a specialized pager that focuses on individual entry views, contrasting with pagers that might display lists of entries (e.g., by month, day, or latest). It ensures that the content for a permalink view is correctly retrieved, wrapped for presentation, and that contextual navigation is available.

Interactions with Other Classes

1. **AbstractWeblogEntriesPager** - Inheritance (WeblogEntriesPermalinkPager 1 : 1 AbstractWeblogEntriesPager) - WeblogEntriesPermalinkPager extends AbstractWeblogEntriesPager to inherit URL creation logic, internationalization utilities, and common constructor parameters
2. **WeblogEntry** - Association (WeblogEntriesPermalinkPager 1 : 3 WeblogEntry) - WeblogEntriesPermalinkPager holds references to currEntry (1), nextEntry (0..1), and prevEntry (0..1) for display and navigation
3. **WeblogEntryWrapper** - Association (WeblogEntriesPermalinkPager 1 : N WeblogEntryWrapper) - WeblogEntriesPermalinkPager stores List<WeblogEntryWrapper> in entries map, wrapping raw WeblogEntry objects for presentation-specific methods
4. **Weblog** - Association (WeblogEntriesPermalinkPager 1 : 1 Weblog) - WeblogEntriesPermalinkPager receives and stores Weblog in constructor to retrieve entries, determine locale/timezone, and construct weblog-specific URLs
5. **URLStrategy** - Association (WeblogEntriesPermalinkPager 1 : 1 URLStrategy) - WeblogEntriesPermalinkPager receives and stores URLStrategy in constructor to generate home link and next/previous entry URLs
6. **WebloggerFactory** - Dependency (WeblogEntriesPermalinkPager 1 : 1 WebloggerFactory) - WeblogEntriesPermalinkPager uses WebloggerFactory statically to obtain central Weblogger instance for accessing business managers
7. **Weblogger** - Dependency (WeblogEntriesPermalinkPager 1 : 1 Weblogger) - WeblogEntriesPermalinkPager obtains Weblogger from WebloggerFactory to access WeblogEntryManager

8. **WeblogEntryManager** - Dependency (WeblogEntriesPermalinkPager 1 : 1 WeblogEntryManager) - WeblogEntriesPermalinkPager obtains WeblogEntryManager from Weblogger to fetch entries by anchor and retrieve next/previous entries chronologically
9. **Log** - Dependency (WeblogEntriesPermalinkPager 1 : 1 Log) - WeblogEntriesPermalinkPager uses static Log instance for logging errors during entry fetching operations
10. **Utilities** - Dependency (WeblogEntriesPermalinkPager 1 : 1 Utilities) - WeblogEntriesPermalinkPager uses Utilities statically for truncateNicely to format next/previous entry titles for navigation display

42. WeblogEntriesListPager

Attributes

- locale : String
- sinceDays : int
- length : int
- queryWeblog : Weblog
- queryUser : User
- queryCat : String
- queryTags : List<String>
- entries : List<WeblogEntryWrapper>
- more : boolean
- lastUpdated : Date

Methods

+ getItems() : List<WeblogEntryWrapper>
+ hasMoreItems() : boolean
+ getLastUpdated() : Date

Role

- WeblogEntriesListPager is a UI-layer pagination component responsible for fetching, filtering, and paginating published weblog entries and preparing them for rendering in views (pages, feeds, archives).

Interaction with other parts of the subsystem (NEED TO CONFIRM)

1. WeblogEntryManager
2. WeblogEntrySearchCriteria
3. WeblogEntry
4. WeblogEntryWrapper
5. **Weblog**
6. **User**

43. RollerVelocity

Attributes

- VELOCITY_CONFIG : String
- velocityEngine : VelocityEngine

Methods

+ getEngine() : VelocityEngine
+ getTemplate(String name) : Template
+ getTemplate(String name, MobileDeviceRepository.DeviceType deviceType) : Template
+ getTemplate(String name, String encoding) : Template
+ getTemplate(String name, MobileDeviceRepository.DeviceType deviceType, String encoding) : Template

Role

- The RollerVelocity class is a Velocity template engine provider and configuration manager for Apache Roller.
- It acts as a singleton wrapper around Apache Velocity, providing centralized initialization, configuration, and access to the Velocity rendering engine.
- This is a critical infrastructure component within the Rendering Engine subsystem that enables template-based content rendering.

Interaction with other parts of the subsystem

1. Template
2. WebloggerConfig
3. RollerContext
4. MobileDeviceRepository

44. ThemeManager (INTERFACE)

Methods

+ getTheme(String id) : SharedTheme
+ getTheme(Weblog weblog) : WeblogTheme
+ getEnabledThemesList() : List<SharedTheme>
+ importTheme(Weblog website, SharedTheme theme,boolean skipStylesheet)
+ reLoadThemeFromDisk(String reloadTheme) : boolean

Role

- The ThemeManager interface defines the business layer contract for managing weblog themes in Apache Roller.
- It provides operations for retrieving, importing, and managing both shared and custom themes, acting as the central theme management authority within the Weblog and Content Subsystem.

- This interface abstracts theme operations from their implementation details.

Interaction with other parts of the subsystem

1. WeblogTheme
2. SharedTheme
3. Weblog

45. WeblogCategory

Attributes:

Database/ID
- id: String

Content/Display Info
- name: String
- description: String
- image: String
- position: int

Associations
- weblog: Weblog

Static/Constant
+ serialVersionUID: long

Methods:

Constructors
+ WeblogCategory()
+ WeblogCategory(weblog: Weblog, name: String, description: String, image: String)

Business/Utility Methods
+ calculatePosition(): void
+ retrieveWeblogEntries(publishedOnly: boolean): List<WeblogEntry> throws WebloggerException
+ isInUse(): boolean

Object Overrides
+ toString(): String
+ equals(other: Object): boolean
+ hashCode(): int

+ compareTo(other: WeblogCategory): int

Getters / Setters
+ getId(): String
+ setId(id: String): void
+ getName(): String
+ setName(name: String): void
+ getDescription(): String
+ setDescription(description: String): void
+ getPosition(): int
+ setPosition(position: int): void
+ getImage(): String
+ setImage(image: String): void
+ getWeblog(): Weblog
+ setWeblog(weblog: Weblog): void

Role:
WeblogCategory represents a category for blog entries. It helps organize content within a weblog, allows retrieval of weblog entries by category, and tracks display order and visual representation (image). It is key for content classification and rendering in the weblog.

Classes it interacts with:

1. **Weblog** - Association/Composition (WeblogCategory 1 : 1 Weblog) - WeblogCategory belongs to exactly one Weblog to calculate position, add itself to weblog's category list, and retrieve entries
2. **WeblogEntry** - Dependency (WeblogCategory 1 : N WeblogEntry) - WeblogCategory uses List<WeblogEntry> for retrieving entries in category through retrieveWeblogEntries()
3. **WebloggerFactory** - Dependency (WeblogCategory 1 : 1 WebloggerFactory) - WeblogCategory calls WebloggerFactory methods to fetch entries and check if category is in use
4. **Utilities** - Dependency (WeblogCategory 1 : 1 Utilities) - WeblogCategory uses Utilities in setName() and setDescription() to remove HTML from input strings for safe storage/display

46. WeblogEntryTag

Attributes:

Static / Logging

- log: Log

Static / Constant

+ serialVersionUID: long

Database / Identification

- id: String

Associations / Links

- website: Weblog

- weblogEntry: WeblogEntry

User / Creator Info

- userName: String

Tag Info / Content

- name: String

- time: Timestamp

Methods:

Constructors

+ WeblogEntryTag()

+ WeblogEntryTag(id: String, website: Weblog, weblogEntry: WeblogEntry, user: User, name: String, time: Timestamp)

Getters / Setters

+ getId(): String

+ setId(id: String): void

+ getWeblog(): Weblog

+ setWeblog(website: Weblog): void

+ getWeblogEntry(): WeblogEntry

+ setWeblogEntry(data: WeblogEntry): void

+ getUser(): User

+ getCreatorUserName(): String

+ setCreatorUserName(userName: String): void

+ getName(): String

+ setName(name: String): void

+ getTime(): Timestamp

+ setTime(tagTime: Timestamp): void

Object Overrides / Utility

+ toString(): String

+ equals(other: Object): boolean

+ hashCode(): int

Role:
WeblogEntryTag represents a tag for a weblog entry, allowing entries to be classified, filtered, and searched by keywords. It supports features like content organization, tag-based navigation, and analytics. Each tag belongs to a single blog entry and records the user who created it and the time it was added.

Classes it interacts with:

1. **Weblog** - Association/Composition (WeblogEntryTag 1 : 1 Weblog) - WeblogEntryTag links to specific Weblog to provide blog context for tag fetching and filtering
2. **WeblogEntry** - Association/Composition (WeblogEntryTag 1 : 1 WeblogEntry) - WeblogEntryTag belongs to single WeblogEntry for retrieval, display, and filtering of tagged content
3. **User** - Dependency (WeblogEntryTag 1 : 1 User) - WeblogEntryTag stores userName and can fetch User object to record tag creator
4. **WebloggerFactory** - Dependency (WeblogEntryTag 1 : 1 WebloggerFactory) - WeblogEntryTag uses WebloggerFactory internally to resolve userName to full User object
5. **Log/LogFactory** - Dependency (WeblogEntryTag 1 : 1 Log) - WeblogEntryTag uses static Log instance to log errors such as User fetch failure

47. WeblogEntryTagAggregate

Attributes:

Static / Constant

+ serialVersionUID: long

Database / Identification

- id: String

Content / Tag Info

- name: String

- total: int

- lastUsed: Timestamp

Associations / Links

- website: Weblog

Methods:

Constructors
+ WeblogEntryTagAggregate()
+ WeblogEntryTagAggregate(id: String, website: Weblog, name: String, total: int)

Getters / Setters
+ getId(): String
+ setId(id: String): void
+ getWeblog(): Weblog
+ setWeblog(website: Weblog): void
+ getName(): String
+ setName(name: String): void
+ getTotal(): int
+ setTotal(total: int): void
+ getLastUsed(): Timestamp
+ setLastUsed(lastUsed: Timestamp): void

Object Overrides / Utility
+ toString(): String
+ equals(other: Object): boolean
+ hashCode(): int

Role:
WeblogEntryTagAggregate is an aggregate class for tags, storing summary statistics about tag usage in a weblog. It helps in generating analytics, tag clouds, or popular tags lists. It does not manage individual entries directly but complements WeblogEntryTag by providing aggregated information for the blog.

Classes it interacts with and relationships:

1. **Weblog** - Association (WeblogEntryTagAggregate 1 : 1 Weblog) - WeblogEntryTagAggregate is scoped to single Weblog and stores reference for filtering and aggregation purposes

7. Media management

48. MediaFile

Attributes:

// Identifiers / basic metadata

- id: String

- name: String

- description: String

- copyrightText: String

// Flags / booleans

- isSharedForGallery: Boolean

// Size / dimensions

- length: long

- width: int

- height: int

- thumbnailHeight: int

- thumbnailWidth: int

// Content / type / path

- contentType: String

- originalPath: String

// Timestamps / audit

- dateUploaded: Timestamp

- lastUpdated: Timestamp

// Ownership / creator / associations

- creatorUserName: String

- weblog: Weblog

- directory: MediaFileDirectory

// Content storage / streams

- is: InputStream

- content: FileContent

- thumbnail: FileContent

// Tag management

- tagSet: Set

- removedTags: Set

- addedTags: Set

Methods:

+ MediaFile()
+ getId(): String
+ setId(String): void
+ getName(): String
+ setName(String): void
+ getDescription(): String
+ setDescription(String): void
+ getCopyrightText(): String
+ setCopyrightText(String): void
+ getSharedForGallery(): Boolean
+ setSharedForGallery(Boolean): void
+ getLength(): long

+ setLength(long): void
+ getDateUploaded(): Timestamp
+ setDateUploaded(Timestamp): void
+ getLastModified(): long
+ getLastUpdated(): Timestamp
+ setLastUpdated(Timestamp): void
+ getDirectory(): MediaFileDirectory
+ setDirectory(MediaFileDirectory): void
+ getTags(): Set
- setTags(Set): void
+ addTag(String): void
+ onRemoveTag(String): void
+ getAddedTags(): Set
+ getRemovedTags(): Set
+ updateTags(List): void
+ getTagsAsString(): String
+ setTagsAsString(String): void
+ getContentType(): String
+ setContentType(String): void
+ getPath(): String
+ getInputStream(): InputStream
+ setInputStream(InputStream): void
+ setContent(FileContent): void
+ isImageFile(): boolean
+ getPermalink(): String
+ getThumbnailURL(): String
+ getCreatorUserName(): String
+ setCreatorUserName(String): void
+ getCreator(): User
+ getOriginalPath(): String
+ setOriginalPath(String): void
+ getWeblog(): Weblog
+ setWeblog(Weblog): void
+ getWidth(): int
+ setWidth(int): void
+ getHeight(): int
+ setHeight(int): void
+ getThumbnailInputStream(): InputStream
+ setThumbnailContent(FileContent): void
+ getThumbnailHeight(): int
+ getThumbnailWidth(): int
- figureThumbnailSize(): void
+ toString(): String
+ equals(Object): boolean

+ hashCode(): int

Role

MediaFile represents an uploaded media asset (image, document, theme resource, or migrated file) associated with a weblog. It encapsulates metadata such as name, description, creator, timestamps, content type, size, and dimensions, along with storage references (directory and FileContent objects for primary and thumbnail data) and tag-related state required by the media subsystem. As a lightweight domain object, it serves managers, rendering logic, and URL strategies by providing a consistent reference to media without containing storage implementation details.

How it interacts with other classes:

1. MediaFileDirectory - Aggregation (MediaFile 1 : 1 MediaFileDirectory) - MediaFile references standalone MediaFileDirectory via directory field and accesses via getDirectory(), setDirectory(), getPath() for logical namespace
2. FileContent - Aggregation (MediaFile 1 : 2 FileContent) - MediaFile holds optional content and thumbnail FileContent references, delegates getInputStream() and getThumbnailInputStream() to content objects
3. MediaFileTag - Composition (MediaFile 1 : N MediaFileTag) - MediaFile owns tagSet, creates tags via addTag(), manages via getTags(), setTagsAsString(), updateTags(), getTagsAsString(), setTags()
4. MediaFileManager - Dependency (MediaFile 1 : 1 MediaFileManager) - MediaFile uses MediaFileManager service via WebloggerFactory in updateTags() for tag removal and accesses MAX_WIDTH/MAX_HEIGHT constants in figureThumbnailSize()
5. Weblog - Aggregation/Association (MediaFile 1 : 1 Weblog) - MediaFile references optional Weblog via weblog field, uses in getWeblog(), setWeblog(), addTag() for locale, and getPermalink()/getThumbnailURL() for URL context
6. User - Dependency (MediaFile 1 : 1 User) - MediaFile resolves optional User lazily via UserManager in getCreator() using stored creatorUserName
7. WebloggerFactory - Dependency (MediaFile 1 : 1 WebloggerFactory) - MediaFile uses singleton WebloggerFactory in getPermalink(), getThumbnailURL(), getCreator(), updateTags() to obtain managers and UrlStrategy
8. UrlStrategy - Dependency (MediaFile 1 : 1 UrlStrategy) - MediaFile uses UrlStrategy service obtained via WebloggerFactory in getPermalink() and getThumbnailURL() to build public URLs

9. Utilities - Dependency (MediaFile 1 : 1 Utilities) - MediaFile uses static Utilities for normalizeTag() in addTag()/updateTags() and splitStringAsTags() in setTagsAsString()
10. MediaFileType - Dependency (MediaFile 1 : 1 MediaFileType) - MediaFile compares contentType with MediaFileType.IMAGE.getContentTypePrefix() in isImageFile() for type checking
11. UUIDGenerator - Dependency (MediaFile 1 : 1 UUIDGenerator) - MediaFile uses UUIDGenerator at construction for default id generation via generateUUID()

49. MediaFileComparator

Attributes:

enum MediaFileComparatorType { NAME, TYPE, DATE_UPLOADED }

- MediaFileComparatorType type

Methods:

- MediaFileComparator(MediaFileComparatorType type)
- int compare(MediaFile file1, MediaFile file2)

Role:
MediaFileComparator is a lightweight utility comparator that orders or presents media files according to a chosen attribute (name, content type, or upload date). It is part of media management and is used wherever collections of org.apache.roller.weblogger.pojos.MediaFile must be sorted before being returned to callers or rendered.

Interactions with other classes:

1. **MediaFile** - Dependency (MediaFileComparator 1 : 2 MediaFile) - MediaFileComparator accesses MediaFile getters (getName(), getContentType(), getDateUploaded()) in compare() to compute ordering between two MediaFile objects
2. **MediaFileManager** - Dependency (MediaFileManager 1 : N MediaFileComparator) - MediaFileManager implementations call MediaFileComparator in sorting routines when returning lists in methods like listMediaFiles() for consistent ordering
3. **FileContentManager** - Dependency (FileContentManager 1 : N MediaFileComparator) - FileContentManager uses MediaFileComparator to sort MediaFile results by content type or upload date when building presentation lists

4. **MediaFileDirectory** - Dependency (MediaFileDirectory 1 : N MediaFileComparator) - MediaFileDirectory uses MediaFileComparator to sort contained MediaFile collections when returning or rendering directory contents
5. **JPAMediaFileManagerImpl** - Dependency (JPAMediaFileManagerImpl 1 : N MediaFileComparator) - JPAMediaFileManagerImpl uses MediaFileComparator to apply uniform comparison logic on in-memory results after fetching from persistence

50. MediaFileDirectory

Attributes:

- id: String

- name: String

- description: String

- weblog: Weblog

- mediaFiles: Set<MediaFile>

Methods:

+ MediaFileDirectory()

+ MediaFileDirectory(weblog: Weblog, name: String, desc: String)

+ isEmpty(): boolean

+ getId(): String

+ setId(id: String): void

+ getName(): String

+ setName(name: String): void

+ getDescription(): String

+ setDescription(description: String): void

+ getWeblog(): Weblog

+ setWeblog(weblog: Weblog): void

+ getMediaFiles(): Set<MediaFile>

+ setMediaFiles(mediaFiles: Set<MediaFile>): void

+ hasMediaFile(name: String): boolean

+ getMediaFile(name: String): MediaFile

+ equals(other: Object): boolean

+ hashCode(): int

Role:

Role: represents a named folder/bucket of media files belonging to a single weblog. It organizes media content (images, documents, etc.) so the rendering layer, upload handlers, and management APIs can find, list, and validate files associated with a weblog.

Responsibilities: stores metadata (id, name, description), maintains a reference to the owning weblog, and holds the collection of contained MediaFile objects; provides simple helper operations such as checking whether a file exists, looking up a file by name, and determining whether the folder is empty.

Where used: media upload and download workflows, media listing in the admin UI, template rendering (resolving media paths through Weblog/MediaFile), and persistence layers or managers responsible for creating, updating, and organizing media files and folders.

How it interacts with other classes:

1. **Weblog** - Composition (Weblog 1 : N MediaFileDirectory) - MediaFileDirectory stores reference to owning Weblog, constructor adds itself to weblog.getMediaFileDirectories() for enumeration, admin UI listing, and MediaFileManager operations
2. **MediaFile** - Composition (MediaFileDirectory 1 : N MediaFile) - MediaFileDirectory holds Set of MediaFile objects, provides hasMediaFile()/getMediaFile() for validation; MediaFile references directory via setDirectory()/getDirectory() and uses directory.getName() in getPath()
3. **MediaFileManager** - Association (MediaFileManager 1 : N MediaFileDirectory) - MediaFileManager implementations create/update directories and media files, use directory metadata and methods like getMediaFiles()/hasMediaFile() for upload validation, persistence, and URL generation

51. MediaFileDirectoryComparator

Attributes:

```
@startuml
package org.apache.roller.weblogger.pojos {
  class MediaFileDirectoryComparator {
    +enum DirectoryComparatorType { NAME }
    ~DirectoryComparatorType type
    +MediaFileDirectoryComparator(DirectoryComparatorType type)
    +int compare(MediaFileDirectory dir1, MediaFileDirectory dir2)
  }
  MediaFileDirectoryComparator ..|> java.util.Comparator
}
@enduml
```

Role:

Acts as a sorting utility that provides a name-based comparator for media file directories, allowing collections of MediaFileDirectory objects to be ordered predictably. It does not store any domain state and exists purely to supply comparison behavior when directory lists need to be displayed, ordered, or processed by managers or the user interface.

How it interacts with other classes:

1. **MediaFileDirectory** - Dependency (MediaFileDirectoryComparator 1 : N MediaFileDirectory) - MediaFileDirectoryComparator uses MediaFileDirectory in compare() method signature and calls getName() to determine ordering when sorting directory collections
2. **Weblog** - Dependency (Weblog 1 : 1 MediaFileDirectoryComparator) - Weblog optionally uses MediaFileDirectoryComparator to sort getMediaFileDirectories() collection for admin UI, API output, or export/import flows
3. **MediaFile** - Indirect Association (MediaFileDirectoryComparator 1 : N MediaFile via MediaFileDirectory) - MediaFileDirectoryComparator indirectly affects MediaFile presentation order by sorting parent MediaFileDirectory objects in UI listings or feed generation

52. MediaFileFilter.java

```
@startuml
```

```
class MediaFileFilter {
    ' enums
    +enum SizeFilterType { GT, GTE, EQ, LT, LTE }
    +enum MediaFileOrder { NAME, DATE_UPLOADED, TYPE }

    ' attributes (package visibility)
    ~name : String
    ~type : MediaFileType
    ~size : long
    ~sizeFilterType : SizeFilterType
    ~tags : List<String>
    ~order : MediaFileOrder
    ~startIndex : int = -1
    ~length : int

    ' methods (public)
    +getName() : String
    +setName(name : String) : void
    +getType() : MediaFileType
    +setType(type : MediaFileType) : void
    +getTags() : List<String>
    +setTags(tags : List<String>) : void
    +getSize() : long
    +setSize(size : long) : void
    +getSizeFilterType() : SizeFilterType
```

```
    +setSizeFilterType(sizeFilterType : SizeFilterType) : void

    +getStartIndex() : int

    +setStartIndex(startIndex : int) : void

    +getLength() : int

    +setLength(length : int) : void

    +getOrder() : MediaFileOrder

    +setOrder(order : MediaFileOrder) : void

}

@enduml
```

Role

MediaFileFilter is a simple search-criteria DTO used to express filtering, sorting, and paging options when querying media files (uploads) belonging to weblogs. It does not implement any logic itself. Instead, it carries parameters that are consumed by media-management services or managers, which perform the actual search and retrieval.

Classes it interacts with

1. MediaFileManager - Dependency/Association (MediaFileManager 1 : 1 MediaFileFilter) - MediaFileManager receives optional MediaFileFilter parameter in search/list methods to constrain results by type, tags, size, ordering, and paging
2. JPAMediaFileManagerImpl - Dependency (JPAMediaFileManagerImpl 1 : 1 MediaFileFilter) - JPAMediaFileManagerImpl reads optional MediaFileFilter values to build JPA/SQL query criteria, apply ordering/pagination, and return matching MediaFile entities
3. MediaFile - Association (MediaFileFilter 1 : N MediaFile) - MediaFileFilter defines constraints corresponding to MediaFile attributes (type, tags, size, metadata) to retrieve zero or more MediaFile instances
4. MediaFileType - Association (MediaFileFilter 1 : 1 MediaFileType) - MediaFileFilter has optional MediaFileType attribute to restrict searches to specific media categories (images, audio, video)
5. MediaFileDirectory - Indirect Association (MediaFileFilter 1 : N MediaFileDirectory) - MediaFileFilter results relate to MediaFileDirectory through returned MediaFile objects containing directory/path details; query implementations may apply directory-based constraints

6. FileContentManager - Indirect Dependency (MediaFileManager 1 : 1 FileContentManager) - MediaFileManager implementations invoke FileContentManager to load file bytes after MediaFileFilter-based queries resolve matching records
7. Weblog - Contextual Association (Weblog 1 : N MediaFileFilter) - Weblog scopes MediaFileFilter usage; search APIs accept both Weblog and optional MediaFileFilter to filter within weblog's media space

53. MediaFileTag.java

class MediaFileTag <<Serializable>> {

- serialVersionUID : long {static, final}

- id : String

- name : String

- mediaFile : MediaFile

+ MediaFileTag()

+ MediaFileTag(name : String, mediaFile : MediaFile)

+ getId() : String

+ setId(id : String) : void

+ getName() : String

+ setName(name : String) : void

+ getMediaFile() : MediaFile

+ setMediaFile(mediaFile : MediaFile) : void

+ toString() : String

+ equals(other : Object) : boolean

+ hashCode() : int

}

Role:

Represents a tag applied to a media file as metadata. It provides a unique identity (id), the tag value (name), and a reference back to the owning MediaFile. It is used for searching and querying media files, displaying tags in pages and feeds, and enabling tag based filtering of media during rendering and feed generation.

Classes it interacts with

1. **MediaFile** - Composition (MediaFile 1 : N MediaFileTag) - MediaFile owns Set of MediaFileTag, manages via updateTags(), setTags(), getTags(); MediaFileTag stores back reference to owning MediaFile
2. **UUIDGenerator** - Dependency (MediaFileTag 1 : N UUIDGenerator) - MediaFileTag uses UUIDGenerator.generateUUID() for identifier initialization as utility dependency
3. **MediaFileManager** - Dependency (MediaFileManager 1 : N MediaFileTag via MediaFile) - MediaFileManager creates, updates, deletes media files; interacts with tags indirectly through MediaFile lifecycle operations
4. **JPAMediaFileManagerImpl** - Dependency (JPAMediaFileManagerImpl 1 : N MediaFileTag via MediaFile) - JPAMediaFileManagerImpl JPA persistence layer loads/stores MediaFile entities with associated tag set as part of entity state
5. **MediaFileDirectory** - Indirect Association (MediaFileDirectory 1 : N MediaFileTag via MediaFile) - MediaFileDirectory contains MediaFile objects which own tags; forms containment chain used in upload interfaces, media browsing, feeds, and resource generation

54. MediaFileType

enum MediaFileType {

// enum constants (public static final)

+ AUDIO

+ VIDEO

+ IMAGE

+ OTHERS

-- **Attributes** --

~ contentTypePrefix : String

~ id : String

~ description : String

-- **Methods** --

- MediaFileType(id: String, desc: String, prefix: String) // private constructor

+ getContentTypePrefix() : String

+ getId() : String

+ getDesc() : String

}

Role:

simple domain enum that categorizes media files into Audio, Video, Image, or Others and stores a content type prefix used for type classification and routing, such as deciding image specific processing or thumbnail logic. It is a value or domain type used by media POJOs and managers, not a manager or service itself.

How it interacts with other classes

1. **MediaFile** - Association/Aggregation (MediaFile N : 1 MediaFileType) - MediaFile references MediaFileType for UI display, validation, and processing decisions; uses MediaFileType metadata (prefix, identifier, description) for classification logic
2. **MediaFileDirectory** - Dependency/Indirect (MediaFileDirectory 1 : N MediaFileType via MediaFile) - MediaFileDirectory contains MediaFile instances classified by MediaFileType; used when listing or serving resources to filter file types
3. **MediaFileManager** - Dependency (MediaFileManager 1 : N MediaFileType) - MediaFileManager uses MediaFileType to choose processing paths (thumbnail generation, upload validation, content handling) for many categorized files
4. **JPAMediaFileManagerImpl** - Dependency (JPAMediaFileManagerImpl 1 : N MediaFileType) - JPAMediaFileManagerImpl branches based on MediaFileType when loading MediaFile entities for thumbnailing, content loading, or saving behavior
5. **FileContentManager** - Dependency (FileContentManager 1 : N MediaFileType) - FileContentManager uses MediaFileType or content type prefix when saving/validating uploaded content to decide storage location or processing for different media types

6. **RollerAtomService** - Dependency (RollerAtomService 1 : N MediaFileType) - RollerAtomService uses MediaFileType to advertise acceptable content types and categorize collections; provides content type prefix for accept headers or collection metadata

55. MediaFileManager (interface)

interface MediaFileManager {

+ static final int MAX_WIDTH = 120

+ static final int MAX_HEIGHT = 120

+ void initialize()

+ void release()

+ void createMediaFile(Weblog weblog, MediaFile mediaFile, RollerMessages errors) throws WebloggerException

+ void createThemeMediaFile(Weblog weblog, MediaFile mediaFile, RollerMessages errors) throws WebloggerException

+ void updateMediaFile(Weblog weblog, MediaFile mediaFile) throws WebloggerException

+ void updateMediaFile(Weblog website, MediaFile mf, java.io.InputStream fis) throws WebloggerException

+ MediaFile getMediaFile(String id) throws WebloggerException

+ MediaFile getMediaFile(String id, boolean includeContent) throws WebloggerException

+ void removeMediaFile(Weblog weblog, MediaFile mediaFile) throws WebloggerException

+ java.util.List<MediaFile> searchMediaFiles(Weblog weblog, MediaFileFilter filter) throws WebloggerException

+ MediaFileDirectory createDefaultMediaFileDirectory(Weblog weblog) throws WebloggerException

+ void createMediaFileDirectory(MediaFileDirectory directory) throws WebloggerException

+ MediaFileDirectory createMediaFileDirectory(Weblog weblog, String name) throws WebloggerException

+ MediaFileDirectory getMediaFileDirectory(String id) throws WebloggerException

+ MediaFileDirectory getMediaFileDirectoryByName(Weblog weblog, String name) throws WebloggerException

+ MediaFile getMediaFileByPath(Weblog weblog, String path) throws WebloggerException

+ MediaFile getMediaFileByOriginalPath(Weblog weblog, String origpath) throws WebloggerException

+ java.util.List<MediaFileDirectory> getMediaFileDirectories(Weblog weblog) throws WebloggerException

+ MediaFileDirectory getDefaultMediaFileDirectory(Weblog weblog) throws WebloggerException

+ void moveMediaFiles(java.util.Collection<MediaFile> mediaFiles, MediaFileDirectory directory) throws WebloggerException

+ void moveMediaFile(MediaFile mediaFile, MediaFileDirectory directory) throws WebloggerException

+ java.util.List<MediaFile> fetchRecentPublicMediaFiles(int length) throws WebloggerException

+ void removeAllFiles(Weblog website) throws WebloggerException

+ void removeMediaFileDirectory(MediaFileDirectory mediaFileDir) throws WebloggerException

+ void removeMediaFileTag(String name, MediaFile entry) throws WebloggerException

}

Role:

Interface that defines the media management API used by the content subsystem. Its responsibilities include lifecycle operations such as initialize and release, creating, updating, and deleting media metadata and content, managing directories, searching and retrieving media files, moving files between directories, cleaning up all media for a weblog, and returning recent public media files. It separates higher level rendering and weblog entry logic from storage and binary content concerns.

How it integrates: rendering code and feed or page models request media metadata and content through this interface. Concrete implementations, such as JPA backed managers, handle persistence and delegate binary storage responsibilities to a FileContentManager.

Interactions:

1. **MediaFile** - Association (MediaFileManager 1 : N MediaFile) - MediaFileManager uses MediaFile as parameter and return type in methods for creating, updating, retrieving, and returning search results
2. **MediaFileDirectory** - Association (MediaFileManager 1 : N MediaFileDirectory) - MediaFileManager uses MediaFileDirectory as parameter and return type for creating, listing, retrieving, deleting directories, and moving files between them
3. **Weblog** - Dependency (MediaFileManager 1 : 1 Weblog) - MediaFileManager receives one Weblog parameter per operation to scope media operations like creating files, listing directories, or removing all files

4. **MediaFileFilter** - Dependency (MediaFileManager 1 : 1 MediaFileFilter) - MediaFileManager receives optional MediaFileFilter parameter in searchMediaFiles for filtering, sorting, and paging criteria
5. **RollerMessages** - Dependency (MediaFileManager 1 : 1 RollerMessages) - MediaFileManager receives optional RollerMessages parameter in create/update methods to collect validation messages and errors for user presentation
6. **FileContentManager** - Association/Aggregation (MediaFileManager 1 : 1 FileContentManager) - MediaFileManager implementations delegate binary content handling (reading/writing uploaded files and thumbnails) to FileContentManager while coordinating metadata and lifecycle
7. **JPAMediaFileManagerImpl** - Realization (JPAMediaFileManagerImpl 1 : 1 MediaFileManager) - JPAMediaFileManagerImpl implements MediaFileManager interface providing persistence, database interaction, transaction handling, and FileContentManager integration
8. **WebloggerFactory/Weblogger** - Association (Weblogger 1 : 1 MediaFileManager) - MediaFileManager obtained via WebloggerFactory.getWeblogger().getMediaFileManager() by rendering code, actions, and other consumers

56. JPAMediaFileManagerImpl.java

Attributes

- roller : org.apache.roller.weblogger.business.Weblogger

- strategy : JPAPersistenceStrategy

- static final log : org.apache.commons.logging.Log

- public static final MIGRATION_STATUS_FILENAME : String

Methods

+ <<constructor>> protected JPAMediaFileManagerImpl(roller : org.apache.roller.weblogger.business.Weblogger, persistenceStrategy : JPAPersistenceStrategy)

+ initialize() : void

+ release() : void

+ moveMediaFiles(mediaFiles : java.util.Collection<org.apache.roller.weblogger.pojos.MediaFile>, targetDirectory : org.apache.roller.weblogger.pojos.MediaFileDirectory) : void throws org.apache.roller.weblogger.WebloggerException

+ moveMediaFile(mediaFile : org.apache.roller.weblogger.pojos.MediaFile, targetDirectory : org.apache.roller.weblogger.pojos.MediaFileDirectory) : void throws org.apache.roller.weblogger.WebloggerException

+ createMediaFileDirectory(directory : org.apache.roller.weblogger.pojos.MediaFileDirectory) : void throws org.apache.roller.weblogger.WebloggerException

+ createMediaFileDirectory(weblog : org.apache.roller.weblogger.pojos.Weblog, requestedName : String) : org.apache.roller.weblogger.pojos.MediaFileDirectory throws org.apache.roller.weblogger.WebloggerException

+ createDefaultMediaFileDirectory(weblog : org.apache.roller.weblogger.pojos.Weblog) : org.apache.roller.weblogger.pojos.MediaFileDirectory throws org.apache.roller.weblogger.WebloggerException

+ createMediaFile(weblog : org.apache.roller.weblogger.pojos.Weblog, mediaFile : org.apache.roller.weblogger.pojos.MediaFile, errors : org.apache.roller.weblogger.util.RollerMessages) : void throws org.apache.roller.weblogger.WebloggerException

+ createThemeMediaFile(weblog : org.apache.roller.weblogger.pojos.Weblog, mediaFile : org.apache.roller.weblogger.pojos.MediaFile, errors : org.apache.roller.weblogger.util.RollerMessages) : void throws org.apache.roller.weblogger.WebloggerException

- updateThumbnail(mediaFile : org.apache.roller.weblogger.pojos.MediaFile) : void

+ updateMediaFile(weblog : org.apache.roller.weblogger.pojos.Weblog, mediaFile : org.apache.roller.weblogger.pojos.MediaFile) : void throws org.apache.roller.weblogger.WebloggerException

+ updateMediaFile(weblog : org.apache.roller.weblogger.pojos.Weblog, mediaFile : org.apache.roller.weblogger.pojos.MediaFile, is : java.io.InputStream) : void throws org.apache.roller.weblogger.WebloggerException

+ getMediaFile(id : String) : org.apache.roller.weblogger.pojos.MediaFile throws org.apache.roller.weblogger.WebloggerException

+ getMediaFile(id : String, includeContent : boolean) : org.apache.roller.weblogger.pojos.MediaFile throws org.apache.roller.weblogger.WebloggerException

+ getMediaFileDirectoryByName(weblog : org.apache.roller.weblogger.pojos.Weblog, name : String) : org.apache.roller.weblogger.pojos.MediaFileDirectory throws org.apache.roller.weblogger.WebloggerException

+ getMediaFileByPath(weblog : org.apache.roller.weblogger.pojos.Weblog, path : String) : org.apache.roller.weblogger.pojos.MediaFile throws org.apache.roller.weblogger.WebloggerException

+ getMediaFileByOriginalPath(weblog : org.apache.roller.weblogger.pojos.Weblog, origpath : String) : org.apache.roller.weblogger.pojos.MediaFile throws org.apache.roller.weblogger.WebloggerException

+ getMediaFileDirectory(id : String) : org.apache.roller.weblogger.pojos.MediaFileDirectory throws org.apache.roller.weblogger.WebloggerException

+ getDefaultMediaFileDirectory(weblog : org.apache.roller.weblogger.pojos.Weblog) : org.apache.roller.weblogger.pojos.MediaFileDirectory throws org.apache.roller.weblogger.WebloggerException

+ getMediaFileDirectories(weblog : org.apache.roller.weblogger.pojos.Weblog) : java.util.List<org.apache.roller.weblogger.pojos.MediaFileDirectory> throws org.apache.roller.weblogger.WebloggerException

+ removeMediaFile(weblog : org.apache.roller.weblogger.pojos.Weblog, mediaFile : org.apache.roller.weblogger.pojos.MediaFile) : void throws org.apache.roller.weblogger.WebloggerException

+ fetchRecentPublicMediaFiles(length : int) : java.util.List<org.apache.roller.weblogger.pojos.MediaFile> throws org.apache.roller.weblogger.WebloggerException

+ searchMediaFiles(weblog : org.apache.roller.weblogger.pojos.Weblog, filter : org.apache.roller.weblogger.pojos.MediaFileFilter) : java.util.List<org.apache.roller.weblogger.pojos.MediaFile> throws org.apache.roller.weblogger.WebloggerException

+ isFileStorageUpgradeRequired() : boolean

+ upgradeFileStorage() : java.util.List<String>

- upgradeUploadsDir(weblog : org.apache.roller.weblogger.pojos.Weblog, user : org.apache.roller.weblogger.pojos.User, oldDir : java.io.File, newDir : org.apache.roller.weblogger.pojos.MediaFileDirectory) : void

+ removeAllFiles(website : org.apache.roller.weblogger.pojos.Weblog) : void throws org.apache.roller.weblogger.WebloggerException

+ removeMediaFileDirectory(dir : org.apache.roller.weblogger.pojos.MediaFileDirectory) : void throws org.apache.roller.weblogger.WebloggerException

+ removeMediaFileTag(name : String, entry : org.apache.roller.weblogger.pojos.MediaFile) :

Role:

JPAMediaFileManagerImpl implements media storage and lifecycle management for uploaded media. It creates and manages media database records, delegates binary content storage to FileContentManager, generates thumbnails, migrates legacy filesystem uploads into the new storage system, and removes media when required. It acts as an adapter and service in the content subsystem, connecting persistent entities such as MediaFile, MediaFileDirectory, and tags to the low level file store and the weblog domain represented by Weblog.

Classes it interacts with

1. Weblogger - Association (JPAMediaFileManagerImpl 1 : 1 Weblogger) - JPAMediaFileManagerImpl receives constructor-injected Weblogger as service locator to access WeblogManager and FileContentManager for runtime services
2. JPAPersistenceStrategy - Association/Composition (JPAMediaFileManagerImpl 1 : 1 JPAPersistenceStrategy) - JPAMediaFileManagerImpl receives injected JPAPersistenceStrategy for database operations (loading, storing, removing, querying persistent entities)
3. FileContentManager - Dependency/Association (JPAMediaFileManagerImpl 1 : 1 FileContentManager) - JPAMediaFileManagerImpl delegates binary file content operations (saving, loading, deleting) to FileContentManager
4. WeblogManager - Dependency (JPAMediaFileManagerImpl 1 : 1 WeblogManager) - JPAMediaFileManagerImpl obtains WeblogManager from Weblogger to update weblog last modified timestamps, retrieve weblog users during migration, and persist weblog changes
5. MediaFile - Association (JPAMediaFileManagerImpl 1 : N MediaFile) - JPAMediaFileManagerImpl manages MediaFile entities through creation, update, retrieval, and deletion operations

6. MediaFileDirectory - Association (JPAMediaFileManagerImpl 1 : N MediaFileDirectory) - JPAMediaFileManagerImpl manages directory entities as containers for organizing media file uploads through creation and modification
7. MediaFileTag - Association/Aggregation (JPAMediaFileManagerImpl 1 : N MediaFileTag) - JPAMediaFileManagerImpl iterates and removes tags associated with media files during media or tag management operations
8. MediaFileFilter - Dependency (JPAMediaFileManagerImpl 1 : 1 MediaFileFilter) - JPAMediaFileManagerImpl receives optional MediaFileFilter parameter per search operation to build dynamic media queries
9. Weblog - Association (JPAMediaFileManagerImpl 1 : 1 Weblog) - JPAMediaFileManagerImpl receives one Weblog per operation as context since media files belong to specific weblog instance
10. RollerMessages - Dependency (JPAMediaFileManagerImpl 1 : N RollerMessages) - JPAMediaFileManagerImpl uses RollerMessages to collect validation and feedback messages during media creation or migration operations
11. WebloggerFactory - Dependency (JPAMediaFileManagerImpl 1 : 1 WebloggerFactory) - JPAMediaFileManagerImpl uses WebloggerFactory statically to obtain FileContentManager when injected Weblogger reference unavailable
12. MediaFileType - Dependency (JPAMediaFileManagerImpl 1 : 1 MediaFileType) - JPAMediaFileManagerImpl uses optional MediaFileType per operation to map media file types to content type prefixes in queries

57. FileContentManager

interface FileContentManager {

-- **Attributes** --

// none

-- **Methods** --

+ FileContent getFileContent(Weblog weblog, String fileId)

throws FileNotFoundException, FilePathException

+ void saveFileContent(Weblog weblog, String fileId, InputStream is)

throws FileNotFoundException, FilePathException, FileIOException

+ void deleteFile(Weblog weblog, String fileId)

throws FileNotFoundException, FilePathException, FileIOException

+ void deleteAllFiles(Weblog weblog)

throws FileIOException

+ boolean overQuota(Weblog weblog)

+ boolean canSave(Weblog weblog, String fileName, String contentType, long size, RollerMessages messages)

+ void release()

}

Role:

Defines the contract for saving, reading, and deleting raw file contents stored for a weblog uploads area. It encapsulates filesystem or storage related concerns such as reading, writing, deleting files, quota checks, and validation.

How it interacts with other classes:

1. **Weblog** - Association (FileContentManager 1 : 1 Weblog) - FileContentManager receives one Weblog parameter per operation to locate correct uploads area or tenant
2. **FileContent** - Association (FileContentManager 1 : 1 FileContent) - FileContentManager returns one FileContent instance per getFileContent() call representing stored binary content and metadata
3. **RollerMessages** - Dependency (FileContentManager 1 : 1 RollerMessages) - FileContentManager receives optional RollerMessages parameter in canSave() to populate validation errors (filename, content type, quota violations)
4. **MediaFileManager** - Association (MediaFileManager 1 : 1 FileContentManager) - MediaFileManager implementations depend on FileContentManager instance to delegate reading/writing file bytes in create, update, or retrieve operations

5. **MediaFile** - Association (FileContentManager 1 : N MediaFile) - FileContentManager interacts with MediaFile objects which receive FileContent from getFileContent() and provide context for saveFileContent()
6. **FileContentManagerImpl** - Realization (FileContentManagerImpl 1 : 1 FileContentManager) - FileContentManagerImpl implements FileContentManager interface performing actual I/O operations (filesystem access, path resolution, thumbnail handling)
7. **WeblogEntryManager** - Indirect Association (WeblogEntryManager 1 : 1 FileContentManager) - WeblogEntryManager indirectly relies on FileContentManager through MediaFileManager when rendering entries or handling embedded media content

58. JPAFileContentManagerImpl

class JPAMediaFileManagerImpl {

- final Weblogger roller

- final JPAPersistenceStrategy strategy

- static final Log log

+ static final String MIGRATION_STATUS_FILENAME

+ <<constructor>> protected JPAMediaFileManagerImpl(Weblogger roller, JPAPersistenceStrategy persistenceStrategy)

+ void initialize()

+ void release()

+ void moveMediaFiles(Collection<MediaFile> mediaFiles, MediaFileDirectory targetDirectory) throws WebloggerException

+ void moveMediaFile(MediaFile mediaFile, MediaFileDirectory targetDirectory) throws WebloggerException

+ void createMediaFileDirectory(MediaFileDirectory directory) throws WebloggerException

+ MediaFileDirectory createMediaFileDirectory(Weblog weblog, String requestedName) throws WebloggerException

+ MediaFileDirectory createDefaultMediaFileDirectory(Weblog weblog) throws WebloggerException

+ void createMediaFile(Weblog weblog, MediaFile mediaFile, RollerMessages errors) throws WebloggerException

+ void createThemeMediaFile(Weblog weblog, MediaFile mediaFile, RollerMessages errors) throws WebloggerException

- void updateThumbnail(MediaFile mediaFile)

+ void updateMediaFile(Weblog weblog, MediaFile mediaFile) throws WebloggerException

+ void updateMediaFile(Weblog weblog, MediaFile mediaFile, InputStream is) throws WebloggerException

+ MediaFile getMediaFile(String id) throws WebloggerException

+ MediaFile getMediaFile(String id, boolean includeContent) throws WebloggerException

+ MediaFileDirectory getMediaFileDirectoryByName(Weblog weblog, String name) throws WebloggerException

+ MediaFile getMediaFileByPath(Weblog weblog, String path) throws WebloggerException

+ MediaFile getMediaFileByOriginalPath(Weblog weblog, String origpath) throws WebloggerException

+ MediaFileDirectory getMediaFileDirectory(String id) throws WebloggerException

+ MediaFileDirectory getDefaultMediaFileDirectory(Weblog weblog) throws WebloggerException

+ List<MediaFileDirectory> getMediaFileDirectories(Weblog weblog) throws WebloggerException

+ void removeMediaFile(Weblog weblog, MediaFile mediaFile) throws WebloggerException

+ List<MediaFile> fetchRecentPublicMediaFiles(int length) throws WebloggerException

+ List<MediaFile> searchMediaFiles(Weblog weblog, MediaFileFilter filter) throws WebloggerException

+ boolean isFileStorageUpgradeRequired()

+ List<String> upgradeFileStorage()

- void upgradeUploadsDir(Weblog weblog, User user, File oldDir, MediaFileDirectory newDir)

+ void removeAllFiles(Weblog website) throws WebloggerException

+ void removeMediaFileDirectory(MediaFileDirectory dir) throws WebloggerException

+ void removeMediaFileTag(String name, MediaFile entry) throws WebloggerException

}

Role:

Implements media storage and management for weblogs, including creating, reading, updating, and deleting media files and directories, generating thumbnails, migrating legacy filesystem uploads, and searching and retrieving media files. It provides services used by the content rendering layer for media referenced by posts and by the administrative user interface for managing uploads.

How it interacts with other classes:

1. **Weblogger** - Association (JPAMediaFileManagerImpl 1 : 1 Weblogger) - JPAMediaFileManagerImpl holds field reference to Weblogger to obtain FileContentManager and WeblogManager, call flush(), and access configuration as central application context
2. **JPAPersistenceStrategy** - Association (JPAMediaFileManagerImpl 1 : 1 JPAPersistenceStrategy) - JPAMediaFileManagerImpl holds field reference to JPAPersistenceStrategy for all database operations (store, load, remove, queries, refresh)
3. **MediaFileManager** - Realization (JPAMediaFileManagerImpl 1 : 1 MediaFileManager) - JPAMediaFileManagerImpl implements MediaFileManager interface as concrete implementation exposed to application
4. **FileContentManager** - Dependency (JPAMediaFileManagerImpl 1 : 1 FileContentManager) - JPAMediaFileManagerImpl obtains FileContentManager from Weblogger/WebloggerFactory to store, retrieve, and delete binary file contents
5. **WeblogManager** - Dependency (JPAMediaFileManagerImpl 1 : 1 WeblogManager) - JPAMediaFileManagerImpl obtains WeblogManager from Weblogger to update weblog last modified state and lookup weblogs/users during migration

6. **MediaFile** - Association (JPAMediaFileManagerImpl 1 : N MediaFile) - JPAMediaFileManagerImpl manipulates MediaFile entities through creation, update, move, removal, search, and thumbnail/content association
7. **MediaFileDirectory** - Association (JPAMediaFileManagerImpl 1 : N MediaFileDirectory) - JPAMediaFileManagerImpl manipulates MediaFileDirectory entities for organizing files through directory lookup, creation, and migration
8. **FileContent** - Dependency (JPAMediaFileManagerImpl 1 : 1 FileContent) - JPAMediaFileManagerImpl uses optional FileContent per media file to attach binary content when loading with includeContent enabled
9. **MediaFileFilter, MediaFileTag, MediaFileType** - Dependency/Association (JPAMediaFileManagerImpl 1 : N) - JPAMediaFileManagerImpl uses optional MediaFileFilter for search input, manipulates MediaFileTag instances, uses optional MediaFileType for classification/filtering
10. **User and Weblog** - Association (JPAMediaFileManagerImpl 1 : N User, 1 : 1 Weblog) - JPAMediaFileManagerImpl uses User entities during migration for creator information and associates operations with one Weblog per operation
11. **RollerMessages** - Dependency (JPAMediaFileManagerImpl 1 : 1 RollerMessages) - JPAMediaFileManagerImpl uses optional RollerMessages as output parameter to collect validation messages during creation/migration
12. **Utilities** - Dependency (JPAMediaFileManagerImpl 1 : 1 Utilities) - JPAMediaFileManagerImpl uses Utilities statically to derive content type information from filenames
13. **java.io.File, java.util.Properties, java.io.InputStream, java.sql.Timestamp** - Dependency (JPAMediaFileManagerImpl 1 : N JDK classes) - JPAMediaFileManagerImpl uses JDK classes for filesystem migration, I/O streams, configuration handling, and timestamping

59. FileContent

class FileContent {

- resourceFile: java.io.File

- fileId: java.lang.String

- weblog: org.apache.roller.weblogger.pojos.Weblog

+ FileContent(weblog: org.apache.roller.weblogger.pojos.Weblog, fileId: java.lang.String, file: java.io.File)

+ getWeblog(): org.apache.roller.weblogger.pojos.Weblog

+ getName(): java.lang.String

+ getFileId(): java.lang.String

+ getLastModified(): long

+ getLength(): long

+ getInputStream(): java.io.InputStream

}

Role:

FileContent encapsulates a filesystem-backed media/resource attached to a weblog. It provides metadata (name, length, lastModified), the owning weblog reference (for scoping/permission/path resolution) and an InputStream factory to read the binary content. It is used by resource managers/DAOs, rendering or download controllers, and any content-fetching logic that serves media for weblog entries.

Classes it interacts with

1. Weblog - Aggregation (Weblog 1 : N FileContent) - FileContent references one owner Weblog via getWeblog() for scoping, permissions, URL/path resolution, and media association
2. java.io.File - Composition (FileContent 1 : 1 File) - FileContent owns backing File instance in resourceFile field; all metadata and stream creation derive from it
3. java.io.InputStream - Dependency (FileContent 1 : 1 InputStream) - FileContent returns InputStream via getInputStream() to consumers for reading file content
4. java.io.FileInputStream - Dependency (FileContent 1 : 1 FileInputStream) - FileContent uses FileInputStream in getInputStream() implementation via new FileInputStream(resourceFile)
5. java.io.FileNotFoundException / java.lang.RuntimeException - Dependency (FileContent 1 : N Exceptions) - FileContent catches FileNotFoundException in getInputStream() and rethrows as RuntimeException to signal stream creation error

Uml code with attributes and methods:

```
@startuml
skinparam classAttributeIconSize 0
skinparam linetype ortho

' ==================== MAIN CLASSES ====================

class Weblog {
  - id : String
  - handle : String
  - name : String
  - tagline : String
  - about : String
  - creator : String
  - dateCreated : Date
  - lastModified : Date
  - enableBloggerApi : Boolean
  - allowComments : Boolean
  - emailComments : Boolean
  - moderateComments : Boolean
  - defaultAllowComments : Boolean
  - defaultCommentDays : int
  - entryDisplayCount : int
  - visible : Boolean
```

- active : Boolean

- enableMultiLang : boolean

- showAllLangs : boolean

- analyticsCode : String

- locale : String

- timeZone : String

- editorTheme : String

- editorPage : String

- iconPath : String

- bloggerCategory : WeblogCategory

- weblogCategories : List<WeblogCategory>

- bookmarkFolders : List<WeblogBookmarkFolder>

- mediaFileDirectories : List<MediaFileDirectory>

- initializedPlugins : Map<String, WeblogEntryPlugin>

+ getTheme() : weblogTheme

+ getId() : String

+ setId(String id)

+ getHandle() : String

+ setHandle(String handle)

+ getName() : String

+ setName(String Name)

+ getTagline() : String

+ setTagline(string tagline)

+ getCreator() : User

+ getCreatorUserName() : String

+ setCreatorUserName(String creatorUserName)

+ getEnableBloggerApi() : Boolean

+ setEnableBloggerApi(Boolean enableBloggerApi)

+ getBloggerCategory() : WeblogCategory

+ setBloggerCategory(WeblogCategory bloggerCategory)

+ getEditorPage() : String

+ setEditorPage(String editorPage)

+ getBannedwordslits() : String

+ setBannedwordslist(String bannedwordslist)

+ getAllowComments() : Boolean

+ setAllowComments(Boolean allowComments)

+ getDefaultAllowComments() : Boolean

+ setDefaultAllowComments(Boolean defaultAllowComments)

+ getDefaultCommentDays() : int

+ setDefaultCommentDays(int defaultCommentDays)

+ getModerateComments() : Boolean

+ setModerateComments(Boolean moderateComments)

+ getEmailComments() : Boolean

+ setEmailComments(Boolean emailComments)

+ getEmailAddress() : String

+ setEmailAddress(String emailAddress)

+ getEditorTheme() : String

+ setEditorTheme(String editorTheme)

+ getLocale() : String

+ setLocale(String locale)

+ getTimeZone() : String

+ setTimeZone(String timeZone)

+ getDateCreated() : Date

+ setDateCreated(final Date date)

+ getDefaultPlugins : String

+ setDefaultPlugins(String string)

+ setData(Weblog other)

+ getLocaleInstance() : Locale

+ getTimeZoneInstance() : TimeZone

+ hasUserPermission(User user, String action) : boolean

+ hasUserPermissions(User user, List<String> actions) : boolean

+ getEntryDisplayCount() : int

+ setEntryDisplayCount(int entryDisplayCount)

+ getVisible() : Boolean

+ setVisible(Boolean visible)

+ getActive() : Boolean

+ setActive(Boolean Active)

+ getCommentModerationRequired() : boolean

+ setCommentModerationRequired(boolean modRequired)

+ getLastModified() : Date

+ setLastModified(Date lastModified)

+ isEnableMultiLang() : boolean

+ setEnableMultiLang(boolean enableMultiLang)

+ isShowAllLangs() : boolean

+ setShowAllLangs(boolean showAllLangs)

+ getURL() : String

+ getAbsoluteURL() : String

+ getIconPath() : String

+ setIconPath(String iconPath)

+ getAnalyticsCode() : String

+ setAnalyticsCode(String analyticsCode)

+ getAbout() : String

+ setAbout(String about)

+ getInitializedPlugins() : Map<String, WeblogEntryPlugin>

+ getWeblogEntry(String anchor) : WeblogEntry

+ getWeblogCategory(String categoryName) : weblogCategory

+ getRecentWeblogEntries(String cat, int length) : List<WeblogEntry>

+ getRecentWeblogEntriesByTag(String tag, int length) : List<WeblogEntry>

+ getRecentComments(int length) : List<WeblogEntryComment>

+ getBookmarkFolder(String folderName) : weblogBookmarkFolder

+ getTodaysHits() : int

+ getPopularTags(int sinceDays, int length) : List<TagStat>

+ getCommentCount() : long

+ getEntryCount() : long

+ addCategory(weblogCategory category)

+ getWeblogCategories() : List<WeblogCategory>

+ setWeblogCategories(List<WeblogCategory> cats)

+ hasCategory(String name) : boolean

+ getBookmarkFolders() : List<WeblogBookmarkFolder>

```
  + setBookmarkFolders(List<WeblogBookmarkFolder> bookmarkFolders)
  + getMediaFileDirectories :List<MediaFileDirectory>
  + setMediaFileDirectories(List<MediaFileDirectory> mediaFileDirectories)
  + addBookmarkFolder(WeblogBookmarkFolder folder)
  + hasBookmarkFolder(String name) : boolean
  + hasMediaFileDirectory(String name) : boolean
  + getMediaFileDirectory(String name) : MediaFileDirectory
}

class User {
  - id : String
  - userName : String
  - password : String
  - openIdUrl : String
  - screenName : String
  - fullName : String
  - emailAddress : String
  - dateCreated : Date
  - locale : String
  - timeZone : String
  - enabled : Boolean
  - activationCode : String

  + getId() : String
  + setId(id : String) : void
```

+ getUserName() : String

+ setUserName(userName : String) : void

+ getPassword() : String

+ setPassword(password : String) : void

+ resetPassword(newPassword : String) : void

+ getOpenIdUrl() : String

+ setOpenIdUrl(openIdUrl : String) : void

+ getScreenName() : String

+ setScreenName(screenName : String) : void

+ getFullName() : String

+ setFullName(fullName : String) : void

+ getEmailAddress() : String

+ setEmailAddress(email : String) : void

+ getDateCreated() : Date

+ setDateCreated(final Date date)

+ getLocale() : String

+ setLocale(locale : String) : void

+ getTimeZone() : String

+ setTimeZone(timeZone : String) : void

+ getEnabled() : Boolean

+ setEnabled(enabled : Boolean) : void

+ getActivationCode() : String

+ setActivationCode(code : String) : void

+ hasGlobalPermission(action : String) : boolean

+ hasGlobalPermissions(actions : List<String>) : boolean

```
}

class WeblogTheme {
  # weblog : Weblog

  + getWeblog() : Weblog
}

class WeblogTemplate {
  - id : String
  - action : ComponentType
  - name : String
  - description : String
  - link : String
  - lastModified : Date
  - hidden : boolean
  - navbar : boolean
  - outputContentType : String
  - weblog : Weblog
  - templateRenditions : List<CustomTemplateRendition>

  + getId() : String
  + setId(String id)
  + getWeblog() : Weblog
  + setWeblog(Weblog website)
```

+ getAction() : ComponentType

+ setAction(ComponentType action)

+ getName() : String

+ setName(String name)

+ getDescription() : String

+ setDescription(String description)

+ getLink() : Strong

+ setLink(STring link)

+ getLastModified() : Date

+ setLastModified(final Date newtime)

+ isNavbar() : boolean

+ setNavbar(boolean navbar)

+ isHiddent() : boolean

+ setHidden(boolean isHidden)

+ getOutputContentType() : String

+ setOutputContentType(String outputContentType)

+ isRequired() : boolean

+ isCustom() : boolean

+ getTemplateRenditions() : List<CustomTemplateRendition>

+ setTemplateRenditions(List<CustomTemplateRendition> templateRenditions)

+ getTemplateRendition(CustomTemplateRendition.RenditionType desiredType) : CustomTemplateRendition

+ addTemplateRendition(CustomTemplateRendition newRendition)

+ hasTemplateRendition(customTemplateRendition proposed) : boolean

}

```
class WeblogBookmark {
  - id : String
  - name : String
  - description : String
  - url : String
  - priority : Integer
  - image : String
  - feedUrl : String
  - folder : WeblogBookmarkFolder

  + getId() : String
  + setId(id : String)
  + getName() : String
  + setName(name : String)
  + getDescription() : String
  + setDescription(description : String)
  + getUrl() : String
  + setUrl(url : String)
  + getPriority() : Integer
  + setPriority(priority : Integer)
  + getImage() : String
  + setImage(image : String)
  + getFeedUrl() : String
  + setFeedUrl(feedUrl : String)
```

```
  + calculatePriority() : void
  + getFolder() : WeblogBookmarkFolder
  + setFolder(folder : WeblogBookmarkFolder)
  + getWebsite() : Weblog
  + compareTo(o : WeblogBookmark) : int
}

class WeblogBookmarkFolder {
  - id : String
  - name : String
  - weblog : Weblog
  - bookmarks : List<WeblogBookmark>

  + getId() : String
  + setId(String id)
  + getName() : String
  + setName(String name)
  + getWeblog() : Weblog
  + setWeblog(Weblog website)
  + getBookmarks() : List<WeblogBookmark>
  + setBookmarks(List<WeblogBookmark> bookmarks)
  + addBookmark(WeblogBookmark bookmark)
  + hasBookmarkOfName(String bookmarkName) : boolean
  + retrieveBookmarks() : List<WeblogBookmark>
  + compareTo(WeblogBookmarkFolder other) : int
```

}

class WeblogEntry {

- id : String
- title : String
- link : String
- summary : String
- text : String
- contentType : String
- contentSrc : String
- anchor : String
- pubTime : Timestamp
- updateTime : Timestamp
- plugins : String
- allowComments : Boolean
- commentDays : Integer
- rightToLeft : Boolean
- pinnedToMain : Boolean
- status : PubStatus
- locale : String
- creatorUserName : String
- searchDescription : String
- refreshAggregates : Boolean
- website : Weblog
- category : WeblogCategory

- attest : Set<WeblogEntryAttribute>

- tagSet : Set<WeblogEntryTag>

- removedTags : Set<WeblogEntryTag>

- addedTags : Set<WeblogEntryTag>

+ setData(WeblogEntry other)

+ getId() : String

+ setId(String id)

+ getCategory() : WeblogCategory

+ setCategory(WeblogCategory cat)

+ getCategories() : List<WeblogCategory>

+ getWebsite() : Weblog

+ setWebsite(Weblog website)

+ getCreator() : User

+ getCreatorUserName() : String

+ setCreatorUserName(String creatorUserName)

+ getTitle() : String

+ setTitle(String title)

+ getSummary() : String

+ setSummary(String summary)

+ getSearchDescription() : String

+ setSearchDescription(String searchDescription)

+ getText() : String

+ setText(String text)

+ getContentType() : String

+ setContentType(String contentType)

+ getContentSrc() : String

+ setContentSrc(String contentSrc)

+ getAnchor() : String

+ setAnchor(String anchor)

+ getEntryAttributes() : Set<WeblogEntryAttribute>

+ setEntryAttributes(Set<WeblogEntryAttribute> atts)

+ findEntryAttribute(String name) : String

+ putEntryAttribute(String name, String value)

+ getPubTime() : Timestamp

+ setPubTime(Timestamp pubTime)

+ getUpdateTime() : Timestamp

+ setUpdateTime(Timestamp updateTime)

+ getStatus() : PubStatus

+ setStatus(PubStatus status)

+ getLink() : String

+ setLink(String link)

+ getPlugins() : String

+ setPlugins(String string)

+ getAllowComments() : Boolean

+ setAllowComments(Boolean allowComments)

+ getCommentDays() : Integer

+ setCommentDays(Integer commentDays)

+ getRightToLeft() : Boolean

+ setRightToLeft(Boolean rightToLeft)

+ getPinnedToMain() : Boolean

+ setPinnedToMain(Boolean pinnedToMain)

+ getLocale() : String

+ setLocale(String locale)

+ getTags() : Set<WeblogEntryTag>

+ setTags(Set<WeblogEntryTag> tagSet)

+ addTag(String name)

+ getAddedTags() : Set<WeblogEntryTag>

+ getRemovedTags() : Set<WeblogEntryTag>

+ getTagsAsString() : String

+ setTagsAsString(String tags)

+ getCommentsStillAllowed() : boolean

+ setCommentsStillAllowed(boolean ignored)

+ formatPubTime(String pattern) : String

+ formatUpdateTime(String pattern) : String

+ getComments() : List<WeblogEntryComment>

+ getComments(boolean ignoreSpam, boolean approvedOnly) : List<WeblogEntryComment>

+ getCommentCount() : int

+ getPermalink() : String

+ getPermaLink() : String

+ getCommentsLink() : String

+ getDisplayTitle() : String

+ createAnchor() : String

+ createAnchorBase() : String

```
  + setPermalink(String string)
  + setPermaLink(String string)
  + setDisplayTitle(String string)
  + getPluginsList() : List<String>
  + isDraft() : boolean
  + isPending() : boolean
  + isPublished() : boolean
  + getTransformedText() : String
  + getTransformedSummary() : String
  + hasWritePermissions(User user) : boolean
  + render(String str) : String
  + displayContent(String readMoreLink) : String
  + getDisplayContent() : String
  + getRefreshAggregates() : Boolean
  + setRefreshAggregates(Boolean refreshAggregates)
}

class WeblogEntryAttribute {
  - id : String
  - entry : WeblogEntry
  - name : String
  - value : String

  + getId() : String
  + setId(String Id)
```

```
  + getEntry() : WeblogEntry
  + setEntry(WeblogEntry entry)
  + getName() : String
  + setName(String name)
  + getValue() : String
  + setValue(String value)
  + compareTo(WeblogEntryAttribute att)
}

class WeblogEntrySearchCriteria {
  - weblog : Weblog
  - user : User
  - startDate : Date
  - endDate : Date
  - catName : String
  - tags : List<String>
  - status : PubStatus
  - text : String
  - sortBy : SortBy
  - sortOrder : SortOrder
  - locale : String

  + getWeblog() : Weblog
  + setWeblog(Weblog weblog)
  + getUser() : User
```

```
  + setUser(User user)
  + getStartDate() : Date
  + setStartDate(Date startDate)
  + getEndDate() : Date
  + setEndDate(Date endDate)
  + getCatName() : String
  + setCatName(String catName)
  + getTags() : List<String>
  + setTags(List<String> tags)
  + getStatus() : PubStatus
  + setStatus(PubStatus status)
  + getText() : String
  + setText(String text)
  + getSortBy() : SortBy
  + setSortBy(SortBy sortBy)
  + getSortOrder() : SortOrder
  + setSortOrder(SortOrder sortOrder)
  + getLocale() : String
  + setLocale(String locale)
}

class WeblogEntryTag {
  - log : Log
  - id : String
  - website: Weblog
```

```
  - userName : String
  - name : String
  - time : Timestamp

  + getId() : String
  + setId(String id)
  + getWeblog() : Weblog
  + setWeblog(Weblog website)
  + getWeblogEntry() : WeblogEntry
  + setWeblogEntry(WeblogEntry data)
  + getUser() : User
  + getCreatorUserName() : String
  + setCreatorUserName(String userName)
  + getName() : String
  + setName(String name)
  + getTime() : Timestamp
  + setTime(Timestamp tagTime)
}

class WeblogEntryTagAggregate {
  - Id : String
  - name : String
  - Website : Weblog
  - lastUsed : Timestamp
  - total : int
```

```
  + getId() : String
  + setId(String id)
  + getName() : String
  + setName(String name)
  + getLastUsed() : Timestamp
  + setLastUsed(Timestamp lastUsed)
  + getTotal() : int
  + setTotal(int total)
}

class WeblogEntryComment {
  - id : String
  - name : String
  - email : String
  - url : String
  - content : String
  - postTime : Timestamp
  - status : ApprovalStatus
  - notify : Boolean
  - remoteHost : String
  - referrer : String
  - userAgent : STring
  - plugins : String
  - contentType : String
```

- weblogEntry : WeblogEntry

+ getId() : String

+ setId(String id)

+ getName() : String

+ setName(String name)

+ getEmail() : String

+ setEmail(String email)

+ getWeblogEntry() : WeblogEntry

+ setWeblogEntry(WeblogEntry entry)

+ getUrl() : String

+ setUrl(String url)

+ getContent() : String

+ setContent(String content)

+ getPostTime() : Timestamp

+ setPostTime(Timestamp postTime)

+ getStatus() : ApprovalStatus

+ setStatus(ApprovalStatus status)

+ getNotify() : Boolean

+ setNotify(Boolean notify)

+ getRemoteHost() : String

+ setRemoteHost(STring remoteHost)

+ getReferrer() : String

+ setReferrer(String referrer)

+ getUserAgent() : String

```
  + setUserAgent(String userAgent)
  + getPlugins() : String
  + setPlugins(String plugins)
  + getContentType() : String
  + setContentType(String ContentType)
  + getPending() : Boolean
  + getApproved() : Boolean
  + getTimestamp() : String
}

class WeblogCategory {
  - id : String
  - name : String
  - description : String
  - image : String
  - position : int
  - weblog : Weblog

  + calculatePosition() : void
  + getId() : String
  + setId(String id)
  + getName() : String
  + setName(String name)
  + getDescription() : String
  + setDescription(String description)
```

```
  + getPosition() : int
  + setPosition(int position)
  + setImage(String image)
  + getImage() : String
  + getWeblog() : Weblog
  + setWeblog(Weblog weblog)
  + retrieveWeblogEntries(boolean publishedOnly) : List<WeblogEntry>
  + isInUse() : boolean
}

class MailProvider {
  - session : Session
  - type : ConfigurationType
  - mailHostName : String
  - mailPort : int
  - mailUsername : String
  - mailPassword : String

  + getSession() : Session
  + getTransport() : Transport
}

class RendererManager {
  - rendererFactories : List<RendererFactory>
```

```
  + getRenderer(Template template, MobileDeviceRepository.DeviceType
deviceType) : Renderer

}


interface WeblogManager {

  + addWeblog(newWebsite : Weblog) : void

  + saveWeblog(data : Weblog) : void

  + removeWeblog(website : Weblog) : void

  + getWeblog(id : String) : Weblog

  + getWeblogByHandle(handle : String) : Weblog

  + getWeblogByHandle(handle : String, enabled : Boolean) : Weblog

  + getWeblogs(enabled : Boolean, active : Boolean, startDate : Date, endDate :
Date, offset : int, length : int) : List<Weblog>

  + getUserWeblogs(user : User, enabledOnly : boolean) : List<Weblog>

  + getWeblogUsers(weblog : Weblog, enabledOnly : boolean) : List<User>

  + getMostCommentedWeblogs(startDate : Date, endDate : Date, offset : int, length
: int) : List<StatCount>

  + getWeblogHandleLetterMap() : Map<String, Long>

  + getWeblogsByLetter(letter : char, offset : int, length : int) : List<Weblog>

  + saveTemplate(data : WeblogTemplate) : void

  + removeTemplate(template : WeblogTemplate) : void

  + getTemplate(id : String) : WeblogTemplate

  + getTemplateByAction(w : Weblog, a : ComponentType) : WeblogTemplate

  + getTemplateByName(w : Weblog, p : String) : WeblogTemplate

  + getTemplateByLink(w : Weblog, p : String) : WeblogTemplate

  + saveTemplateRendition(templateCode : CustomTemplateRendition) : void
```

```
  + getTemplates(w : Weblog) : List<WeblogTemplate>

  + getWeblogCount() : long

  + release() : void

}


interface UserManager {

  + addUser(newUser : User) : void

  + saveUser(user : User) : void

  + removeUser(user : User) : void

  + getUserCount() : long

  + getUserByActivationCode(activationCode : String) : User

  + getUser(id : String) : User

  + getUserByUserName(userName : String) : User

  + getUserByUserName(userName : String, enabled : Boolean) : User

  + getUserByOpenIdUrl(openIdUrl : String) : User

  + getUsers(enabled : Boolean, startDate : Date, endDate : Date, offset : int, length :
int) : List<User>

  + getUsersStartingWith(startsWith : String, enabled : Boolean, offset : int, length :
int) : List<User>

  + getUserNameLetterMap() : Map<String, Long>

  + getUsersByLetter(letter : char, offset : int, length : int) : List<User>

  + checkPermission(perm : RollerPermission, user : User) : boolean

  + grantWeblogPermission(weblog : Weblog, user : User, actions : List<String>) :
void

  + grantWeblogPermissionPending(weblog : Weblog, user : User, actions :
List<String>) : void

  + confirmWeblogPermission(weblog : Weblog, user : User) : void
```

```
  + declineWeblogPermission(weblog : Weblog, user : User) : void

  + revokeWeblogPermission(weblog : Weblog, user : User, actions : List<String>) :
void

  + getWeblogPermissions(user : User) : List<WeblogPermission>

  + getPendingWeblogPermissions(user : User) : List<WeblogPermission>

  + getWeblogPermissions(weblog : Weblog) : List<WeblogPermission>

  + getPendingWeblogPermissions(weblog : Weblog) : List<WeblogPermission>

  + getWeblogPermissionsIncludingPending(weblog : Weblog) :
List<WeblogPermission>

  + getWeblogPermission(weblog : Weblog, user : User) : WeblogPermission

  + getWeblogPermissionIncludingPending(weblog : Weblog, user : User) :
WeblogPermission

  + grantRole(roleName : String, user : User) : void

  + revokeRole(roleName : String, user : User) : void

  + hasRole(roleName : String, user : User) : boolean

  + getRoles(user : User) : List<String>

  + release() : void

}


interface WeblogEntryManager {

  + saveWeblogEntry(entry : WeblogEntry) : void

  + removeWeblogEntry(entry : WeblogEntry) : void

  + getWeblogEntry(id : String) : WeblogEntry

  + getWeblogEntryByAnchor(website : Weblog, anchor : String) : WeblogEntry

  + getWeblogEntries(wesc : WeblogEntrySearchCriteria) : List<WeblogEntry>

  + getWeblogEntryObjectMap(wesc : WeblogEntrySearchCriteria) : Map<Date,
List<WeblogEntry>>
```

+ getWeblogEntryStringMap(wesc : WeblogEntrySearchCriteria) : Map<Date, String>

+ getMostCommentedWeblogEntries(website : Weblog, startDate : Date, endDate : Date, offset : int, length : int) : List<StatCount>

+ getNextEntry(current : WeblogEntry, catName : String, locale : String) : WeblogEntry

+ getPreviousEntry(current : WeblogEntry, catName : String, locale : String) : WeblogEntry

+ getWeblogEntriesPinnedToMain(max : Integer) : List<WeblogEntry>

+ removeWeblogEntryAttribute(name : String, entry : WeblogEntry) : void

+ saveWeblogCategory(cat : WeblogCategory) : void

+ removeWeblogCategory(cat : WeblogCategory) : void

+ getWeblogCategory(id : String) : WeblogCategory

+ moveWeblogCategoryContents(srcCat : WeblogCategory, destCat : WeblogCategory) : void

+ getWeblogCategoryByName(website : Weblog, categoryName : String) : WeblogCategory

+ getWeblogCategories(website : Weblog) : List<WeblogCategory>

+ saveComment(comment : WeblogEntryComment) : void

+ removeComment(comment : WeblogEntryComment) : void

+ getComment(id : String) : WeblogEntryComment

+ getComments(csc : CommentSearchCriteria) : List<WeblogEntryComment>

+ removeMatchingComments(website : Weblog, entry : WeblogEntry, searchString : String, startDate : Date, endDate : Date, status : ApprovalStatus) : int

+ createAnchor(data : WeblogEntry) : String

+ isDuplicateWeblogCategoryName(data : WeblogCategory) : boolean

+ isWeblogCategoryInUse(data : WeblogCategory) : boolean

+ applyCommentDefaultsToEntries(website : Weblog) : void

```
  + release() : void

  + getPopularTags(website : Weblog, startDate : Date, offset : int, limit : int) :
List<TagStat>

  + getTags(website : Weblog, sortBy : String, startsWith : String, offset : int, limit : int)
: List<TagStat>

  + getTagComboExists(tags : List<String>, weblog : Weblog) : boolean

  + getHitCount(id : String) : WeblogHitCount

  + getHitCountByWeblog(weblog : Weblog) : WeblogHitCount

  + getHotWeblogs(sinceDays : int, offset : int, length : int) : List<WeblogHitCount>

  + saveHitCount(hitCount : WeblogHitCount) : void

  + removeHitCount(hitCount : WeblogHitCount) : void

  + incrementHitCount(weblog : Weblog, amount : int) : void

  + resetAllHitCounts() : void

  + resetHitCount(weblog : Weblog) : void

  + getCommentCount() : long

  + getCommentCount(websiteData : Weblog) : long

  + getEntryCount() : long

  + getEntryCount(websiteData : Weblog) : long

}


interface Weblogger {

  + getUserManager() : UserManager

  + getBookmarkManager() : BookmarkManager

  + getOAuthManager() : OAuthManager

  + getWeblogManager() : WeblogManager

  + getWeblogEntryManager() : WeblogEntryManager
```

```
  + getAutopingManager() : AutoPingManager
  + getPingTargetManager() : PingTargetManager
  + getPingQueueManager() : PingQueueManager
  + getPropertiesManager() : PropertiesManager
  + getThreadManager() : ThreadManager
  + getIndexManager() : IndexManager
  + getThemeManager() : ThemeManager
  + getPluginManager() : PluginManager
  + getMediaFileManager() : MediaFileManager
  + getFileContentManager() : FileContentManager
  + getUrlStrategy() : URLStrategy
  + flush() : void
  + release() : void
  + initialize() : void
  + shutdown() : void
  + getVersion() : String
  + getRevision() : String
  + getBuildTime() : String
  + getBuildUser() : String
  + getFeedFetcher() : FeedFetcher
  + getPlanetManager() : PlanetManager
  + getPlanetURLStrategy() : PlanetURLStrategy
}

class TagStat {
```

```
  - serialVersionUID : long
  - name : String
  - count : int
  - intensity : int

  + TagStat()
  + getName() : String
  + setName(String name) : void
  + getCount() : int
  + setCount(int count) : void
  + toString() : String
  + getIntensity() : int
  + setIntensity(int intensity) : void
}

interface PluginManager {
  + hasPagePlugins() : boolean
  + getWeblogEntryPlugins(website : Weblog) : Map<String, WeblogEntryPlugin>
  + applyWeblogEntryPlugins(pagePlugins : Map<String, WeblogEntryPlugin>, entry :
WeblogEntry, str : String) : String
  + getCommentPlugins() : List
  + applyCommentPlugins(comment : WeblogEntryComment, text : String) : String
  + release() : void
}
```

```
class PluginManagerImpl {
  -log : Log
  -mPagePlugins : Map<String, Class<? extends WeblogEntryPlugin>>
  -commentPlugins : List<WeblogEntryCommentPlugin>

  +PluginManagerImpl()
  +hasPagePlugins() : boolean
  +getWeblogEntryPlugins(website : Weblog) : Map<String, WeblogEntryPlugin>
  +applyWeblogEntryPlugins(pagePlugins : Map<String, WeblogEntryPlugin>, entry :
WeblogEntry, str : String) : String
  +getCommentPlugins() : List<WeblogEntryCommentPlugin>
  +applyCommentPlugins(comment : WeblogEntryComment, text : String) : String
  -loadPagePluginClasses() : void
  -loadCommentPlugins() : void
  +release() : void
}

interface WeblogEntryPlugin {
  + getName() : String
  + getDescription() : String
  + init(weblog : Weblog) : void
  + render(entry : WeblogEntry, str : String) : String
}

interface WeblogEntryCommentPlugin {
```

```
  + getId() : String
  + getName() : String
  + getDescription() : String
  + render(comment : WeblogEntryComment, str : String) : String
}

class CommentSearchCriteria {
  - weblog : Weblog
  - entry : WeblogEntry
  - searchText : String
  - startDate : Date
  - endDate : Date
  - status : ApprovalStatus
  - reverseChrono : boolean
  - offset : int
  - maxResults : int

  + getWeblog() : Weblog
  + setWeblog(weblog : Weblog) : void
  + getEntry() : WeblogEntry
  + setEntry(entry : WeblogEntry) : void
  + getSearchText() : String
  + setSearchText(searchText : String) : void
  + getStartDate() : Date
  + setStartDate(startDate : Date) : void
```

```
  + getEndDate() : Date
  + setEndDate(endDate : Date) : void
  + getStatus() : ApprovalStatus
  + setStatus(status : ApprovalStatus) : void
  + isReverseChrono() : boolean
  + setReverseChrono(reverseChrono : boolean) : void
  + getOffset() : int
  + setOffset(offset : int) : void
  + getMaxResults() : int
  + setMaxResults(maxResults : int) : void
}

class GlobalCommentManagement {
  - log : Log
  - COUNT : int = 30
  - bean : GlobalCommentManagementBean
  - pager : CommentsPager
  - firstComment : WeblogEntryComment
  - lastComment : WeblogEntryComment
  - bulkDeleteCount : int = 0
  - httpMethod : String = "GET"

  + GlobalCommentManagement()
  + requiredGlobalPermissionActions() : List<String>
  + isWeblogRequired() : boolean
```

```
  + loadComments() : void
  - buildBaseUrl() : String
  + execute() : String
  + query() : String
  + delete() : String
  + update() : String
  + getCommentStatusOptions() : List<KeyValueObject>
  + getBean() : GlobalCommentManagementBean
  + setBean(bean : GlobalCommentManagementBean) : void
  + getBulkDeleteCount() : int
  + setBulkDeleteCount(bulkDeleteCount : int) : void
  + getFirstComment() : WeblogEntryComment
  + setFirstComment(firstComment : WeblogEntryComment) : void
  + getLastComment() : WeblogEntryComment
  + setLastComment(lastComment : WeblogEntryComment) : void
  + getPager() : CommentsPager
  + setPager(pager : CommentsPager) : void
  + setServletRequest(req : HttpServletRequest) : void
}

interface CommentAuthenticator {
  + getHtml(request : HttpServletRequest) : String
  + authenticate(request : HttpServletRequest) : boolean
}
```

```
interface CommentValidator {
  + getName() : String
  + validate(WeblogEntryComment comment, RollerMessages messages) : int
}

class CommentValidationManager {
  - log : Log
  - validators : List<CommentValidator>

  + CommentValidationManager()
  + addCommentValidator(val : CommentValidator) : void
  + validateComment(comment : WeblogEntryComment, messages :
RollerMessages) : int
}

interface Renderer {
  + render(model : Map<String, Object>, writer : Writer) : void
}

class VelocityRenderer {
  - log : Log
  - renderTemplate : Template
  - deviceType : MobileDeviceRepository.DeviceType
  - velocityTemplate : org.apache.velocity.Template
  - velocityDecorator : org.apache.velocity.Template
```

```
  - velocityException : Exception


  + VelocityRenderer(template : Template, deviceType :
MobileDeviceRepository.DeviceType)

  + render(model : Map<String, Object>, out : Writer) : void

  - renderException(model : Map<String, Object>, out : Writer, template : String) :
void

}


interface TemplateRendition {

  + getTemplate() : String

  + getTemplateLanguage() : TemplateLanguage

  + getType() : RenditionType

  + setTemplate(template : String) : void

  + setTemplateLanguage(templateLanguage : TemplateLanguage) : void

  + setType(type : RenditionType) : void

}


class CustomTemplateRendition {

  - serialVersionUID : long

  - id : String

  - weblogTemplate : WeblogTemplate

  - template : String

  - type : RenditionType

  - templateLanguage : TemplateLanguage
```

```
  + CustomTemplateRendition(template : WeblogTemplate, type : RenditionType)
  + CustomTemplateRendition()
  + getWeblogTemplate() : WeblogTemplate
  + setWeblogTemplate(weblogTemplate : WeblogTemplate) : void
  + getId() : String
  + setId(id : String) : void
  + getTemplate() : String
  + setTemplate(template : String) : void
  + getType() : RenditionType
  + setType(type : RenditionType) : void
  + toString() : String
  + equals(other : Object) : boolean
  + hashCode() : int
  + getTemplateLanguage() : TemplateLanguage
  + setTemplateLanguage(templateLanguage : TemplateLanguage) : void
}

class PageModel {
  -log: Log
  -pageRequest: WeblogPageRequest
  -urlStrategy: URLStrategy
  -commentForm: WeblogEntryCommentForm
  -requestParameters: Map<String, String[]>
  -weblog: Weblog
  -deviceType: DeviceType
```

```
  +PageModel()
  +getModelName(): String
  +init(initData: Map<String, Object>): void
  +getLocale(): String
  +getWeblog(): WeblogWrapper
  +isPermalink(): boolean
  +isSearchResults(): boolean
  +getWeblogEntry(): WeblogEntryWrapper
  +getWeblogPage(): ThemeTemplateWrapper
  +getWeblogCategory(): WeblogCategoryWrapper
  +getTags(): List<String>
  +getDeviceType(): String
  +getWeblogEntriesPager(): WeblogEntriesPager
  +getWeblogEntriesPager(catArgument: String): WeblogEntriesPager
  +getWeblogEntriesPagerByTag(tagArgument: String): WeblogEntriesPager
  +getCommentForm(): WeblogEntryCommentForm
  +getRequestParameter(paramName: String): String
  -getWeblogEntriesPager(catArgument: String, tagArgument: String):
WeblogEntriesPager
}

class WeblogPageRequest {
  - log : Log
  - PAGE_SERVLET : String
```

- context : String

- weblogAnchor : String

- weblogPageName : String

- weblogCategoryName : String

- weblogDate : String

- tags : List<String>

- pageNum : int

- customParams : Map<String, String[]>

- weblogEntry : WeblogEntry

- weblogPage : ThemeTemplate

- weblogCategory : WeblogCategory

- websitePageHit : boolean

- otherPageHit : boolean

+ WeblogPageRequest()

+ WeblogPageRequest(request : HttpServletRequest)

~ isValidDestination(servlet : String) : boolean

- isValidDateString(dateString : String) : boolean

+ getContext() : String

+ setContext(context : String) : void

+ getWeblogAnchor() : String

+ setWeblogAnchor(weblogAnchor : String) : void

+ getWeblogPageName() : String

+ setWeblogPageName(weblogPage : String) : void

+ getWeblogCategoryName() : String

```
  + setWeblogCategoryName(weblogCategory : String) : void
  + getWeblogDate() : String
  + setWeblogDate(weblogDate : String) : void
  + getPageNum() : int
  + setPageNum(pageNum : int) : void
  + getCustomParams() : Map<String, String[]>
  + setCustomParams(customParams : Map<String, String[]>) : void
  + getTags() : List<String>
  + setTags(tags : List<String>) : void
  + getWeblogEntry() : WeblogEntry
  + setWeblogEntry(weblogEntry : WeblogEntry) : void
  + getWeblogPage() : ThemeTemplate
  + setWeblogPage(weblogPage : WeblogTemplate) : void
  + getWeblogCategory() : WeblogCategory
  + setWeblogCategory(weblogCategory : WeblogCategory) : void
  + isWebsitePageHit() : boolean
  + setWebsitePageHit(websitePageHit : boolean) : void
  + isOtherPageHit() : boolean
  + setOtherPageHit(otherPageHit : boolean) : void
}

class WeblogFeedRequest {
  - log : Log
  - FEED_SERVLET : String
  - type : String
```

- format : String

- weblogCategoryName : String

- tags : List<String>

- page : int

- excerpts : boolean

- term : String

- weblogCategory : WeblogCategory

+ WeblogFeedRequest()

+ WeblogFeedRequest(request : HttpServletRequest)

+ getType() : String

+ setType(type : String) : void

+ getFormat() : String

+ setFormat(format : String) : void

+ getWeblogCategoryName() : String

+ setWeblogCategoryName(weblogCategory : String) : void

+ getTags() : List<String>

+ setTags(tags : List<String>) : void

+ isExcerpts() : boolean

+ setExcerpts(excerpts : boolean) : void

+ getWeblogCategory() : WeblogCategory

+ setWeblogCategory(weblogCategory : WeblogCategory) : void

+ getPage() : int

+ setPage(page : int) : void

+ getTerm() : String

```
  + setTerm(query : String) : void
}

class WeblogSearchRequest {
  - log : Log
  - SEARCH_SERVLET : String
  - query : String
  - pageNum : int
  - weblogCategoryName : String
  - weblogCategory : WeblogCategory

  + WeblogSearchRequest()
  + WeblogSearchRequest(request : HttpServletRequest)
  + getQuery() : String
  + setQuery(query : String) : void
  + getPageNum() : int
  + setPageNum(pageNum : int) : void
  + getWeblogCategoryName() : String
  + setWeblogCategoryName(weblogCategory : String) : void
  + getWeblogCategory() : WeblogCategory
  + setWeblogCategory(weblogCategory : WeblogCategory) : void
}

interface Pager {
  + getHomeLink(): String
```

```
  + getHomeName(): String
  + getNextLink(): String
  + getNextName(): String
  + getPrevLink(): String
  + getPrevName(): String
  + getItems(): List<T>
}

interface WeblogEntriesPager {
  + getEntries() : Map<Date, ? extends Collection>
  + getHomeLink() : String
  + getHomeName() : String
  + getNextLink() : String
  + getNextName() : String
  + getPrevLink() : String
  + getPrevName() : String
  + getNextCollectionLink() : String
  + getNextCollectionName() : String
  + getPrevCollectionLink() : String
  + getPrevCollectionName() : String
}

class WeblogEntriesPermalinkPager {
  - log : Log
  - currEntry : WeblogEntry
```

```
  - nextEntry : WeblogEntry

  - prevEntry : WeblogEntry

  - entries : Map<Date, List<WeblogEntryWrapper>>


  + WeblogEntriesPermalinkPager(URLStrategy strat, Weblog weblog, String locale,
String pageLink, String entryAnchor, String dateString, String catName, List<String>
tags, int page)

  + getEntries() : Map<Date, List<WeblogEntryWrapper>>

  + getHomeLink() : String

  + getHomeName() : String

  + getNextLink() : String

  + getNextName() : String

  + getPrevLink() : String

  + getPrevName() : String

  - getNextEntry() : WeblogEntry

  - getPrevEntry() : WeblogEntry

}


class WeblogEntriesListPager {

  - locale : String

  - sinceDays : int

  - length : int

  - queryWeblog : Weblog

  - queryUser : User

  - queryCat : String

  - queryTags : List<String>
```

```
  - entries : List<WeblogEntryWrapper>

  - more : boolean

  - lastUpdated : Date


  + getItems() : List<WeblogEntryWrapper>

  + hasMoreItems() : boolean

  + getLastUpdated() : Date

}


class RollerVelocity {

  - VELOCITY_CONFIG : String

  - velocityEngine : VelocityEngine


  + getEngine() : VelocityEngine

  + getTemplate(String name) : Template

  + getTemplate(String name, MobileDeviceRepository.DeviceType deviceType) :
Template

  + getTemplate(String name, String encoding) : Template

  + getTemplate(String name, MobileDeviceRepository.DeviceType deviceType,
String encoding) : Template

}


interface ThemeManager {

  + getTheme(String id) : SharedTheme

  + getTheme(Weblog weblog) : WeblogTheme

  + getEnabledThemesList() : List<SharedTheme>
```

```
  + importTheme(Weblog website, SharedTheme theme,boolean skipStylesheet)
  + reLoadThemeFromDisk(String reloadTheme) : boolean
}

class MediaFile {
  - id: String
  - name: String
  - description: String
  - copyrightText: String
  - isSharedForGallery: Boolean
  - length: long
  - width: int
  - height: int
  - thumbnailHeight: int
  - thumbnailWidth: int
  - contentType: String
  - originalPath: String
  - dateUploaded: Timestamp
  - lastUpdated: Timestamp
  - creatorUserName: String
  - weblog: Weblog
  - directory: MediaFileDirectory
  - is: InputStream
  - content: FileContent
  - thumbnail: FileContent
```

- tagSet: Set

- removedTags: Set

- addedTags: Set

+ MediaFile()

+ getId(): String

+ setId(String): void

+ getName(): String

+ setName(String): void

+ getDescription(): String

+ setDescription(String): void

+ getCopyrightText(): String

+ setCopyrightText(String): void

+ getSharedForGallery(): Boolean

+ setSharedForGallery(Boolean): void

+ getLength(): long

+ setLength(long): void

+ getDateUploaded(): Timestamp

+ setDateUploaded(Timestamp): void

+ getLastModified(): long

+ getLastUpdated(): Timestamp

+ setLastUpdated(Timestamp): void

+ getDirectory(): MediaFileDirectory

+ setDirectory(MediaFileDirectory): void

+ getTags(): Set

- setTags(Set): void

+ addTag(String): void

+ onRemoveTag(String): void

+ getAddedTags(): Set

+ getRemovedTags(): Set

+ updateTags(List): void

+ getTagsAsString(): String

+ setTagsAsString(String): void

+ getContentType(): String

+ setContentType(String): void

+ getPath(): String

+ getInputStream(): InputStream

+ setInputStream(InputStream): void

+ setContent(FileContent): void

+ isImageFile(): boolean

+ getPermalink(): String

+ getThumbnailURL(): String

+ getCreatorUserName(): String

+ setCreatorUserName(String): void

+ getCreator(): User

+ getOriginalPath(): String

+ setOriginalPath(String): void

+ getWeblog(): Weblog

+ setWeblog(Weblog): void

+ getWidth(): int

```
  + setWidth(int): void
  + getHeight(): int
  + setHeight(int): void
  + getThumbnailInputStream(): InputStream
  + setThumbnailContent(FileContent): void
  + getThumbnailHeight(): int
  + getThumbnailWidth(): int
  - figureThumbnailSize(): void
  + toString(): String
  + equals(Object): boolean
  + hashCode(): int
}

class MediaFileComparator {
  - type : MediaFileComparatorType

  + MediaFileComparator(MediaFileComparatorType type)
  + compare(MediaFile file1, MediaFile file2) : int
}

class MediaFileDirectory {
  - id: String
  - name: String
  - description: String
  - weblog: Weblog
```

```
  - mediaFiles: Set<MediaFile>

  + MediaFileDirectory()
  + MediaFileDirectory(weblog: Weblog, name: String, desc: String)
  + isEmpty(): boolean
  + getId(): String
  + setId(id: String): void
  + getName(): String
  + setName(name: String): void
  + getDescription(): String
  + setDescription(description: String): void
  + getWeblog(): Weblog
  + setWeblog(weblog: Weblog): void
  + getMediaFiles(): Set<MediaFile>
  + setMediaFiles(mediaFiles: Set<MediaFile>): void
  + hasMediaFile(name: String): boolean
  + getMediaFile(name: String): MediaFile
  + equals(other: Object): boolean
  + hashCode(): int
}

class MediaFileDirectoryComparator {
  - type : DirectoryComparatorType

  + MediaFileDirectoryComparator(DirectoryComparatorType type)
```

```
  + compare(MediaFileDirectory dir1, MediaFileDirectory dir2) : int
}

class MediaFileFilter {
  - name : String
  - type : MediaFileType
  - size : long
  - sizeFilterType : SizeFilterType
  - tags : List<String>
  - order : MediaFileOrder
  - startIndex : int = -1
  - length : int

  + getName() : String
  + setName(name : String) : void
  + getType() : MediaFileType
  + setType(type : MediaFileType) : void
  + getTags() : List<String>
  + setTags(tags : List<String>) : void
  + getSize() : long
  + setSize(size : long) : void
  + getSizeFilterType() : SizeFilterType
  + setSizeFilterType(sizeFilterType : SizeFilterType) : void
  + getStartIndex() : int
  + setStartIndex(startIndex : int) : void
```

```
  + getLength() : int
  + setLength(length : int) : void
  + getOrder() : MediaFileOrder
  + setOrder(order : MediaFileOrder) : void
}

class MediaFileTag {
  - serialVersionUID : long
  - id : String
  - name : String
  - mediaFile : MediaFile

  + MediaFileTag()
  + MediaFileTag(name : String, mediaFile : MediaFile)
  + getId() : String
  + setId(id : String) : void
  + getName() : String
  + setName(name : String) : void
  + getMediaFile() : MediaFile
  + setMediaFile(mediaFile : MediaFile) : void
  + toString() : String
  + equals(other : Object) : boolean
  + hashCode() : int
}
```

enum MediaFileType {

AUDIO

VIDEO

IMAGE

OTHERS

- contentTypePrefix : String

- id : String

- description : String

- MediaFileType(id: String, desc: String, prefix: String)

+ getContentTypePrefix() : String

+ getId() : String

+ getDesc() : String

}

interface MediaFileManager {

+ {static} MAX_WIDTH : int = 120

+ {static} MAX_HEIGHT : int = 120

+ initialize() : void

+ release() : void

+ createMediaFile(Weblog weblog, MediaFile mediaFile, RollerMessages errors) : void

+ createThemeMediaFile(Weblog weblog, MediaFile mediaFile, RollerMessages errors) : void

```
  + updateMediaFile(Weblog weblog, MediaFile mediaFile) : void
  + updateMediaFile(Weblog website, MediaFile mf, java.io.InputStream fis) : void
  + getMediaFile(String id) : MediaFile
  + getMediaFile(String id, boolean includeContent) : MediaFile
  + removeMediaFile(Weblog weblog, MediaFile mediaFile) : void
  + searchMediaFiles(Weblog weblog, MediaFileFilter filter) : java.util.List<MediaFile>
  + createDefaultMediaFileDirectory(Weblog weblog) : MediaFileDirectory
  + createMediaFileDirectory(MediaFileDirectory directory) : void
  + createMediaFileDirectory(Weblog weblog, String name) : MediaFileDirectory
  + getMediaFileDirectory(String id) : MediaFileDirectory
  + getMediaFileDirectoryByName(Weblog weblog, String name) : MediaFileDirectory
  + getMediaFileByPath(Weblog weblog, String path) : MediaFile
  + getMediaFileByOriginalPath(Weblog weblog, String origpath) : MediaFile
  + getMediaFileDirectories(Weblog weblog) : java.util.List<MediaFileDirectory>
  + getDefaultMediaFileDirectory(Weblog weblog) : MediaFileDirectory
  + moveMediaFiles(java.util.Collection<MediaFile> mediaFiles, MediaFileDirectory
directory) : void
  + moveMediaFile(MediaFile mediaFile, MediaFileDirectory directory) : void
  + fetchRecentPublicMediaFiles(int length) : java.util.List<MediaFile>
  + removeAllFiles(Weblog website) : void
  + removeMediaFileDirectory(MediaFileDirectory mediaFileDir) : void
  + removeMediaFileTag(String name, MediaFile entry) : void
}


class JPAMediaFileManagerImpl {
```

- roller : Weblogger

- strategy : JPAPersistenceStrategy

- log : Log

+ {static} MIGRATION_STATUS_FILENAME : String

JPAMediaFileManagerImpl(roller : Weblogger, persistenceStrategy : JPAPersistenceStrategy)

+ initialize() : void

+ release() : void

+ moveMediaFiles(mediaFiles : Collection<MediaFile>, targetDirectory : MediaFileDirectory) : void

+ moveMediaFile(mediaFile : MediaFile, targetDirectory : MediaFileDirectory) : void

+ createMediaFileDirectory(directory : MediaFileDirectory) : void

+ createMediaFileDirectory(weblog : Weblog, requestedName : String) : MediaFileDirectory

+ createDefaultMediaFileDirectory(weblog : Weblog) : MediaFileDirectory

+ createMediaFile(weblog : Weblog, mediaFile : MediaFile, errors : RollerMessages) : void

+ createThemeMediaFile(weblog : Weblog, mediaFile : MediaFile, errors : RollerMessages) : void

- updateThumbnail(mediaFile : MediaFile) : void

+ updateMediaFile(weblog : Weblog, mediaFile : MediaFile) : void

+ updateMediaFile(weblog : Weblog, mediaFile : MediaFile, is : InputStream) : void

+ getMediaFile(id : String) : MediaFile

+ getMediaFile(id : String, includeContent : boolean) : MediaFile

+ getMediaFileDirectoryByName(weblog : Weblog, name : String) : MediaFileDirectory

+ getMediaFileByPath(weblog : Weblog, path : String) : MediaFile

```
  + getMediaFileByOriginalPath(weblog : Weblog, origpath : String) : MediaFile

  + getMediaFileDirectory(id : String) : MediaFileDirectory

  + getDefaultMediaFileDirectory(weblog : Weblog) : MediaFileDirectory

  + getMediaFileDirectories(weblog : Weblog) : List<MediaFileDirectory>

  + removeMediaFile(weblog : Weblog, mediaFile : MediaFile) : void

  + fetchRecentPublicMediaFiles(length : int) : List<MediaFile>

  + searchMediaFiles(weblog : Weblog, filter : MediaFileFilter) : List<MediaFile>

  + isFileStorageUpgradeRequired() : boolean

  + upgradeFileStorage() : List<String>

  - upgradeUploadsDir(weblog : Weblog, user : User, oldDir : File, newDir :
MediaFileDirectory) : void

  + removeAllFiles(website : Weblog) : void

  + removeMediaFileDirectory(dir : MediaFileDirectory) : void

  + removeMediaFileTag(name : String, entry : MediaFile) : void

}


interface FileContentManager {

  + getFileContent(Weblog weblog, String fileId) : FileContent

  + saveFileContent(Weblog weblog, String fileId, InputStream is) : void

  + deleteFile(Weblog weblog, String fileId) : void

  + deleteAllFiles(Weblog weblog) : void

  + overQuota(Weblog weblog) : boolean

  + canSave(Weblog weblog, String fileName, String contentType, long size,
RollerMessages messages) : boolean

  + release() : void

}
```

```
class FileContent {
  - resourceFile: File
  - fileId: String
  - weblog: Weblog

  + FileContent(weblog: Weblog, fileId: String, file: File)
  + getWeblog(): Weblog
  + getName(): String
  + getFileId(): String
  + getLastModified(): long
  + getLength(): long
  + getInputStream(): InputStream
}

' ==================== RELATIONSHIPS ====================

' Weblog relationships
Weblog "1" *-- "0..*" WeblogCategory : contains
Weblog "1" *-- "0..*" WeblogEntry : has
Weblog "1" *-- "0..*" WeblogBookmarkFolder : owns
Weblog "1" *-- "0..*" MediaFileDirectory : owns
Weblog "1" -- "0..1" WeblogCategory : bloggerCategory
Weblog "1" -- "0..*" WeblogEntryTag : has
Weblog "1" -- "0..*" WeblogEntryTagAggregate : has
```

```
Weblog "1" o-- "0..*" WeblogTemplate : has

' User relationships
User "1" -- "0..*" Weblog : creates
User "1" -- "0..*" WeblogEntry : creates

' WeblogTheme relationships
WeblogTheme "1" -- "1" Weblog : bound to

' WeblogTemplate relationships
WeblogTemplate "1" *-- "0..*" CustomTemplateRendition : has
WeblogTemplate "*" -- "1" Weblog : belongs to

' WeblogBookmark relationships
WeblogBookmark "*" -- "1" WeblogBookmarkFolder : contained in
WeblogBookmarkFolder "*" -- "1" Weblog : belongs to

' WeblogEntry relationships
WeblogEntry "*" -- "1" Weblog : belongs to
WeblogEntry "*" -- "1" WeblogCategory : categorized by
WeblogEntry "*" -- "1" User : created by
WeblogEntry "1" *-- "0..*" WeblogEntryTag : tagged with
WeblogEntry "1" *-- "0..*" WeblogEntryComment : has
WeblogEntry "1" *-- "0..*" WeblogEntryAttribute : has
```

```
' WeblogEntryAttribute relationships
WeblogEntryAttribute "*" --* "1" WeblogEntry : belongs to

' WeblogEntrySearchCriteria relationships
WeblogEntrySearchCriteria "0..1" -- "0..1" Weblog : filters
WeblogEntrySearchCriteria "0..1" -- "0..1" User : filters by

' WeblogEntryTag relationships
WeblogEntryTag "*" -- "1" Weblog : scoped to
WeblogEntryTag "*" -- "1" WeblogEntry : tags
WeblogEntryTag "*" -- "1" User : created by

' WeblogEntryTagAggregate relationships
WeblogEntryTagAggregate "*" -- "1" Weblog : aggregates for

' WeblogEntryComment relationships
WeblogEntryComment "*" -- "1" WeblogEntry : comments on

' WeblogCategory relationships
WeblogCategory "*" -- "1" Weblog : belongs to
WeblogCategory "1" -- "0..*" WeblogEntry : categorizes

' Manager interfaces
WeblogManager ..> Weblog : manages
WeblogManager ..> User : uses
```

```
WeblogManager ..> WeblogTemplate : manages

WeblogManager ..> CustomTemplateRendition : manages


UserManager ..> User : manages

UserManager ..> Weblog : manages permissions


WeblogEntryManager ..> WeblogEntry : manages

WeblogEntryManager ..> Weblog : uses

WeblogEntryManager ..> WeblogCategory : manages

WeblogEntryManager ..> WeblogEntryComment : manages

WeblogEntryManager ..> WeblogEntrySearchCriteria : uses

WeblogEntryManager ..> CommentSearchCriteria : uses

WeblogEntryManager ..> TagStat : returns


' Weblogger relationships

Weblogger ..> UserManager : provides

Weblogger ..> WeblogManager : provides

Weblogger ..> WeblogEntryManager : provides

Weblogger ..> ThemeManager : provides

Weblogger ..> PluginManager : provides

Weblogger ..> MediaFileManager : provides

Weblogger ..> FileContentManager : provides


' Plugin relationships

PluginManager ..> Weblog : uses
```

PluginManager ..> WeblogEntryPlugin : manages

PluginManager ..> WeblogEntry : applies to

PluginManager ..> WeblogEntryCommentPlugin : manages

PluginManager ..> WeblogEntryComment : applies to

PluginManagerImpl ..|> PluginManager : implements

PluginManagerImpl ..> WeblogEntryPlugin : instantiates

PluginManagerImpl ..> WeblogEntryCommentPlugin : instantiates

PluginManagerImpl ..> Weblog : uses

PluginManagerImpl ..> WeblogEntry : uses

PluginManagerImpl ..> WeblogEntryComment : uses

WeblogEntryPlugin ..> Weblog : initialized with

WeblogEntryPlugin ..> WeblogEntry : renders

WeblogEntryCommentPlugin ..> WeblogEntryComment : processes

' Comment Search and Management

CommentSearchCriteria "0..1" -- "0..1" Weblog : filters

CommentSearchCriteria "0..1" -- "0..1" WeblogEntry : filters

GlobalCommentManagement ..> WeblogEntryManager : uses

GlobalCommentManagement ..> WeblogEntryComment : manages

GlobalCommentManagement ..> Weblog : uses

GlobalCommentManagement ..> CommentSearchCriteria : uses

CommentValidationManager "1" o-- "0..*" CommentValidator : manages

CommentValidationManager ..> WeblogEntryComment : validates

' Rendering relationships

VelocityRenderer ..|> Renderer : implements

CustomTemplateRendition ..|> TemplateRendition : implements

CustomTemplateRendition "*" -- "1" WeblogTemplate : belongs to

PageModel ..> WeblogPageRequest : uses

PageModel ..> Weblog : references

PageModel ..> WeblogEntriesPager : creates

WeblogPageRequest ..> WeblogEntry : loads

WeblogPageRequest ..> WeblogCategory : loads

WeblogPageRequest ..> WeblogEntryManager : uses

WeblogFeedRequest ..> WeblogCategory : filters by

WeblogFeedRequest ..> WeblogEntryManager : uses

WeblogSearchRequest ..> WeblogCategory : filters by

WeblogSearchRequest ..> WeblogEntryManager : uses

WeblogEntriesPermalinkPager ..> WeblogEntry : displays

WeblogEntriesPermalinkPager ..> Weblog : scoped to

WeblogEntriesPermalinkPager ..> WeblogEntryManager : uses

WeblogEntriesListPager ..> Weblog : filters by

WeblogEntriesListPager ..> User : filters by

ThemeManager ..> Weblog : uses

ThemeManager ..> WeblogTheme : returns

' Media relationships

MediaFile "*" -- "1" MediaFileDirectory : stored in

MediaFile "*" -- "0..1" Weblog : belongs to

MediaFile "*" -- "1" User : created by

MediaFile "1" *-- "0..*" MediaFileTag : tagged with

MediaFile "1" -- "0..2" FileContent : has content

MediaFileDirectory "*" -- "1" Weblog : belongs to

MediaFileDirectory "1" *-- "0..*" MediaFile : contains

MediaFileTag "*" --* "1" MediaFile : tags

MediaFileManager ..> MediaFile : manages

MediaFileManager ..> MediaFileDirectory : manages

MediaFileManager ..> Weblog : scoped to

MediaFileManager ..> MediaFileFilter : uses

```
MediaFileManager ..> FileContentManager : delegates to

JPAMediaFileManagerImpl ..|> MediaFileManager : implements
JPAMediaFileManagerImpl ..> Weblogger : uses
JPAMediaFileManagerImpl ..> MediaFile : manages
JPAMediaFileManagerImpl ..> MediaFileDirectory : manages
JPAMediaFileManagerImpl ..> FileContentManager : delegates to
JPAMediaFileManagerImpl ..> Weblog : uses
JPAMediaFileManagerImpl ..> User : uses
JPAMediaFileManagerImpl ..> MediaFileTag : manages
JPAMediaFileManagerImpl ..> MediaFileFilter : uses
JPAMediaFileManagerImpl ..> FileContent : uses

FileContentManager ..> Weblog : scoped to
FileContentManager ..> FileContent : returns

FileContent "*" -- "1" Weblog : belongs to

@enduml
```

Uml code without attributes and methods:

```
@startuml
skinparam classAttributeIconSize 0
skinparam linetype ortho

' ==================== CLASS DECLARATIONS ====================

class Weblog
class User
class WeblogTheme
class WeblogTemplate
class WeblogBookmark
class WeblogBookmarkFolder
class WeblogEntry
class WeblogEntryAttribute
class WeblogEntrySearchCriteria
class WeblogEntryTag
class WeblogEntryTagAggregate
class WeblogEntryComment
class WeblogCategory
class MailProvider
class RendererManager
interface WeblogManager
interface UserManager
interface WeblogEntryManager
interface Weblogger
```

class TagStat

interface PluginManager

class PluginManagerImpl

interface WeblogEntryPlugin

interface WeblogEntryCommentPlugin

class CommentSearchCriteria

class GlobalCommentManagement

interface CommentAuthenticator

interface CommentValidator

class CommentValidationManager

interface Renderer

class VelocityRenderer

interface TemplateRendition

class CustomTemplateRendition

class PageModel

class WeblogPageRequest

class WeblogFeedRequest

class WeblogSearchRequest

interface Pager

interface WeblogEntriesPager

class WeblogEntriesPermalinkPager

class WeblogEntriesListPager

class RollerVelocity

interface ThemeManager

class MediaFile

```
class MediaFileComparator

class MediaFileDirectory

class MediaFileDirectoryComparator

class MediaFileFilter

class MediaFileTag

enum MediaFileType

interface MediaFileManager

class JPAMediaFileManagerImpl

interface FileContentManager

class FileContent


' ==================== RELATIONSHIPS ====================


' Weblog relationships

Weblog "1" *-- "0..*" WeblogCategory

Weblog "1" *-- "0..*" WeblogEntry

Weblog "1" *-- "0..*" WeblogBookmarkFolder

Weblog "1" *-- "0..*" MediaFileDirectory

Weblog "1" -- "0..1" WeblogCategory

Weblog "1" -- "0..*" WeblogEntryTag

Weblog "1" -- "0..*" WeblogEntryTagAggregate

Weblog "1" o-- "0..*" WeblogTemplate


' User relationships

User "1" -- "0..*" Weblog
```

```
User "1" -- "0..*" WeblogEntry

' WeblogTheme relationships
WeblogTheme "1" -- "1" Weblog

' WeblogTemplate relationships
WeblogTemplate "1" *-- "0..*" CustomTemplateRendition
WeblogTemplate "*" -- "1" Weblog

' WeblogBookmark relationships
WeblogBookmark "*" -- "1" WeblogBookmarkFolder
WeblogBookmarkFolder "*" -- "1" Weblog

' WeblogEntry relationships
WeblogEntry "*" -- "1" Weblog
WeblogEntry "*" -- "1" WeblogCategory
WeblogEntry "*" -- "1" User
WeblogEntry "1" *-- "0..*" WeblogEntryTag
WeblogEntry "1" *-- "0..*" WeblogEntryComment
WeblogEntry "1" *-- "0..*" WeblogEntryAttribute

' WeblogEntryAttribute relationships
WeblogEntryAttribute "*" --* "1" WeblogEntry

' WeblogEntrySearchCriteria relationships
```

```
WeblogEntrySearchCriteria "0..1" -- "0..1" Weblog

WeblogEntrySearchCriteria "0..1" -- "0..1" User


' WeblogEntryTag relationships

WeblogEntryTag "*" -- "1" Weblog

WeblogEntryTag "*" -- "1" WeblogEntry

WeblogEntryTag "*" -- "1" User


' WeblogEntryTagAggregate relationships

WeblogEntryTagAggregate "*" -- "1" Weblog


' WeblogEntryComment relationships

WeblogEntryComment "*" -- "1" WeblogEntry


' WeblogCategory relationships

WeblogCategory "*" -- "1" Weblog

WeblogCategory "1" -- "0..*" WeblogEntry


' Manager interfaces

WeblogManager ..> Weblog

WeblogManager ..> User

WeblogManager ..> WeblogTemplate

WeblogManager ..> CustomTemplateRendition


UserManager ..> User
```

UserManager ..> Weblog

WeblogEntryManager ..> WeblogEntry

WeblogEntryManager ..> Weblog

WeblogEntryManager ..> WeblogCategory

WeblogEntryManager ..> WeblogEntryComment

WeblogEntryManager ..> WeblogEntrySearchCriteria

WeblogEntryManager ..> CommentSearchCriteria

WeblogEntryManager ..> TagStat

' Weblogger relationships

Weblogger ..> UserManager

Weblogger ..> WeblogManager

Weblogger ..> WeblogEntryManager

Weblogger ..> ThemeManager

Weblogger ..> PluginManager

Weblogger ..> MediaFileManager

Weblogger ..> FileContentManager

' Plugin relationships

PluginManager ..> Weblog

PluginManager ..> WeblogEntryPlugin

PluginManager ..> WeblogEntry

PluginManager ..> WeblogEntryCommentPlugin

PluginManager ..> WeblogEntryComment

```
PluginManagerImpl ..|> PluginManager

PluginManagerImpl ..> WeblogEntryPlugin

PluginManagerImpl ..> WeblogEntryCommentPlugin

PluginManagerImpl ..> Weblog

PluginManagerImpl ..> WeblogEntry

PluginManagerImpl ..> WeblogEntryComment


WeblogEntryPlugin ..> Weblog

WeblogEntryPlugin ..> WeblogEntry


WeblogEntryCommentPlugin ..> WeblogEntryComment


' Comment Search and Management

CommentSearchCriteria "0..1" -- "0..1" Weblog

CommentSearchCriteria "0..1" -- "0..1" WeblogEntry


GlobalCommentManagement ..> WeblogEntryManager

GlobalCommentManagement ..> WeblogEntryComment

GlobalCommentManagement ..> Weblog

GlobalCommentManagement ..> CommentSearchCriteria


CommentValidationManager "1" o-- "0..*" CommentValidator

CommentValidationManager ..> WeblogEntryComment
```

```
' Rendering relationships

VelocityRenderer ..|> Renderer

CustomTemplateRendition ..|> TemplateRendition

CustomTemplateRendition "*" -- "1" WeblogTemplate

PageModel ..> WeblogPageRequest

PageModel ..> Weblog

PageModel ..> WeblogEntriesPager

WeblogPageRequest ..> WeblogEntry

WeblogPageRequest ..> WeblogCategory

WeblogPageRequest ..> WeblogEntryManager

WeblogFeedRequest ..> WeblogCategory

WeblogFeedRequest ..> WeblogEntryManager

WeblogSearchRequest ..> WeblogCategory

WeblogSearchRequest ..> WeblogEntryManager

WeblogEntriesPermalinkPager ..> WeblogEntry

WeblogEntriesPermalinkPager ..> Weblog

WeblogEntriesPermalinkPager ..> WeblogEntryManager

WeblogEntriesListPager ..> Weblog
```

```
WeblogEntriesListPager ..> User

ThemeManager ..> Weblog
ThemeManager ..> WeblogTheme

' Media relationships
MediaFile "*" -- "1" MediaFileDirectory
MediaFile "*" -- "0..1" Weblog
MediaFile "*" -- "1" User
MediaFile "1" *-- "0..*" MediaFileTag
MediaFile "1" -- "0..2" FileContent

MediaFileDirectory "*" -- "1" Weblog
MediaFileDirectory "1" *-- "0..*" MediaFile

MediaFileTag "*" --* "1" MediaFile

MediaFileManager ..> MediaFile
MediaFileManager ..> MediaFileDirectory
MediaFileManager ..> Weblog
MediaFileManager ..> MediaFileFilter
MediaFileManager ..> FileContentManager

JPAMediaFileManagerImpl ..|> MediaFileManager
JPAMediaFileManagerImpl ..> Weblogger
```

JPAMediaFileManagerImpl ..> MediaFile

JPAMediaFileManagerImpl ..> MediaFileDirectory

JPAMediaFileManagerImpl ..> FileContentManager

JPAMediaFileManagerImpl ..> Weblog

JPAMediaFileManagerImpl ..> User

JPAMediaFileManagerImpl ..> MediaFileTag

JPAMediaFileManagerImpl ..> MediaFileFilter

JPAMediaFileManagerImpl ..> FileContent

FileContentManager ..> Weblog

FileContentManager ..> FileContent

FileContent "*" -- "1" Weblog

@enduml

Strengths and weakness of this design

Strengths

1. **Clean Separation**: The design keeps different types of work separate - you have classes for storing data (Weblog, User), classes for business logic (WeblogManager), and classes for displaying pages (PageModel, Renderer). Each part has its own job.

2. **Easy to Swap Parts**: Managers use interfaces, so you can replace how data is stored (switch databases) without breaking the rest of the code.

3. **No Monster Method Parameters**: Instead of methods with 10 parameters like searchEntries(weblog, user, startDate, endDate, category, tags, status, text, sortBy, sortOrder), they use objects like WeblogEntrySearchCriteria that

bundle everything together.

4. **Plugins Work Well**: You can add new features (comment validators, text processors) by creating new plugin classes without touching existing code.

5. **Parent-Child Relationships Are Clear**: When a WeblogEntry is deleted, its comments and tags go with it. When a MediaFileDirectory is deleted, its files are handled properly. The ownership is obvious.

6. **Smart Tag Handling**: Individual tags (WeblogEntryTag) track each use, while aggregates (WeblogEntryTagAggregate) pre-calculate totals for displaying tag clouds quickly.

Weaknesses

1. **Domain Objects Call Managers**: A MediaFile shouldn't know how to save itself by calling MediaFileManager. A Weblog shouldn't fetch its own entries. This mixes "what it is" with "how it's managed."

2. **Weblog Class Does Everything**: Weblog handles configuration, user permissions, analytics, theme settings, content retrieval, categories, bookmarks, and media. One class shouldn't wear this many hats.

3. **WeblogEntryManager Is Overloaded**: This one manager handles blog entries, categories, comments, tags, and hit counters. That's 5 different jobs - should be 5 different managers.

4. **Who Does What?**: WeblogEntry can get its own comments AND WeblogEntryManager can get comments for an entry. Which one should you use? The design doesn't make this clear.

5. **Unpredictable Data Loading**: Sometimes related data loads automatically, sometimes you have to explicitly fetch it. This makes it hard to know if your code will make 1 database call or 100.

6. **Too Many Steps for Simple Tasks**: To create a blog post with tags and a category, you call WeblogEntryManager, then add tags, then set category, then save again. Should be one operation.

7. **Request Classes Do Too Much**: WeblogPageRequest reads the URL, validates it, AND fetches database objects. Should just parse the URL and let something else load data.

8. **Circular References**: Weblog contains MediaFileDirectories, MediaFileDirectory contains MediaFiles, MediaFile points back to Weblog. This creates a web that's hard to test and understand.

9. **Unclear Save Points**: When you call saveWeblogEntry(), does it save immediately to database or wait? When does a transaction actually commit? Not obvious.

10. **Pager Interfaces Don't Match**: WeblogEntriesPager adds extra methods that regular Pager doesn't have. This means code expecting a Pager won't work correctly with WeblogEntriesPager.

Overall: The design does a good job separating different concerns and using interfaces, but it puts too much responsibility in a few "superclasses," blurs the lines between what stores data and what manages it, and creates tangled relationships that make testing and maintenance harder than necessary.