Apache Roller: Complete Subsystem Documentation

Blogs, Entries, Comments & Rendering Engine

Software Engineering Project

January 26, 2026

1 Overview

The Apache Roller subsystem consists of **39 key classes and interfaces** across five functional areas:

1.1 Subsystem Components

- **Blog Management (9 classes):** CreateWeblog, CreateWeblogBean, WebloggerFactory, Weblogger, WeblogManager, JPAWeblogManagerImpl, UserManager, MediaFileManager, ThemeManager
- **Entry Management (6 classes):** EntryEdit, EntryBean, WeblogEntry, WeblogEntryManager, JPAWeblogEntryManagerImpl, WeblogCategory
- **Comment Management (4 classes):** Comments, CommentsBean, WeblogEntryComment, Comment posting logic
- **Rendering Engine (6 classes):** Renderer, RendererFactory, RendererManager, VelocityRenderer, VelocityRendererFactory, RollerVelocity
- **Content Fetching & Serving (8 classes):** WeblogRequestMapper, PageServlet, FeedServlet, CommentServlet, Model, PageModel, FeedModel, SiteModel
- **Shared Infrastructure (6 classes):** PersistenceStrategy, User, WeblogPermission, IndexManager, CacheManager, MailUtil

2 Blog Management Classes

2.1 CreateWeblog (Action)

Package: `org.apache.roller.weblogger.ui.struts2.core`

Purpose: Controller for blog creation workflow. Handles the presentation and submission of the blog creation form.

Attributes:

- `bean : CreateWeblogBean` – Form data holder
- `pageTitle : String` – Page title for display

Key Methods:

```
public String execute()        // Display form
public String save()           // Process submission
public void myValidate()       // Validate handle uniqueness
public List<SharedTheme> getThemes()  // Get available themes
```

Interactions:

- Uses `WeblogManager` for blog persistence
- Uses `UserManager` for permission granting
- Uses `ThemeManager` for theme selection
- Creates `Weblog` entity

2.2 CreateWeblogBean (DTO)

Package: `org.apache.roller.weblogger.ui.struts2.core`
Purpose: Data Transfer Object that holds blog creation form data.
Properties:

```
private String handle;         // Unique blog identifier
private String name;           // Blog display name
private String description;    // Blog description
private String emailAddress;   // Contact email
private String locale;         // Language setting
private String timeZone;       // Timezone setting
private String theme;          // Visual theme
```

2.3 WebloggerFactory (Factory)

Package: `org.apache.roller.weblogger.business`
Purpose: Singleton factory providing the main Weblogger application instance.
Key Methods:

```
public static Weblogger getWeblogger()  // Returns singleton
public static void bootstrap()          // Initialize application
public static boolean isBootstrapped()  // Check initialization
```

2.4 Weblogger (Interface)

Package: `org.apache.roller.weblogger.business`
Purpose: Main application interface providing access to all managers.
Methods:

```
WeblogManager getWeblogManager();
UserManager getUserManager();
WeblogEntryManager getWeblogEntryManager();
MediaFileManager getMediaFileManager();
ThemeManager getThemeManager();
IndexManager getIndexManager();
PingTargetManager getPingTargetManager();
AutoPingManager getAutopingManager();
void flush();
String getVersion();
String getRevision();
```

2.5 WeblogManager (Interface)

Package: `org.apache.roller.weblogger.business`

Purpose: Interface for blog CRUD operations.

Methods:

```
void addWeblog(Weblog weblog);
Weblog getWeblogByHandle(String handle);
Weblog getWeblog(String id);
void saveWeblog(Weblog weblog);
void removeWeblog(Weblog weblog);
```

2.6 JPAWeblogManagerImpl (Implementation)

Package: `org.apache.roller.weblogger.business.jpa`

Purpose: JPA implementation of WeblogManager.

Attributes:

```
private PersistenceStrategy strategy;
private Weblogger roller;
```

Critical Method:

```
private void addWeblogContents(Weblog weblog) {
    // Creates:
    // - Default categories
    // - Default bookmarks
    // - Owner permissions (ADMIN)
    // - Media file directories
}
```

2.7 Weblog (Entity)

Package: `org.apache.roller.weblogger.pojos`

Purpose: Blog domain model entity.

Key Properties:

```
private String id;
private String handle;          // Unique URL identifier
private String name;            // Display name
private String tagline;         // Blog subtitle
```

```
private String creatorUserName;   // Owner username
private String emailAddress;      // Contact email
private String locale;            // Language
private String timezone;          // Timezone
private String theme;             // Template theme
private String editorpage;        // Editor settings
private Boolean allowcomments;    // Allow comments flag
private Boolean defaultAllowComments;
private Integer defaultCommentDays;
private String defaultPlugins;
private Boolean enabled;          // Active status
private Date datecreated;         // Creation date
```

Relationships:

- OneToMany with `WeblogEntry` – contains entries
- OneToMany with `WeblogCategory` – has categories
- OneToMany with `WeblogPermission` – grants permissions

3 Entry Management Classes

3.1 EntryEdit (Action)

Package: `org.apache.roller.weblogger.ui.struts2.editor`

Purpose: Controller for creating and editing blog entries.

Attributes:

```
private EntryBean bean;        // Form data
private WeblogEntry entry;     // Current entry
private String trackbackUrl;   // Trackback URL
```

Key Methods:

```
public String execute()        // Display entry editor
public String saveDraft()      // Save as draft
public String publish()        // Publish entry
public String firstSave()      // Initial save
public String trackback()      // Send trackback
private String save()          // Core save logic
public List<WeblogCategory> getCategories()
public List getRecentPublishedEntries()
public List getRecentDraftEntries()
```

Flow:

1. Create/load WeblogEntry
2. Copy form data from EntryBean
3. Set timestamps
4. Save to database
5. Update search index

6. Invalidate cache
7. Queue ping notifications

3.2 EntryBean (DTO)

Package: `org.apache.roller.weblogger.ui.struts2.editor`

Purpose: Holds entry form data for transfer between UI and domain.

Properties:

```
private String id;
private String title;              // Entry title
private String text;               // Main content
private String summary;            // Short summary
private String searchDescription; // SEO description
private String categoryId;         // Category selection
private String tagsAsString;       // Tags (comma-separated)
private String status;             // Publication status
private String locale;             // Language
private Boolean allowComments;     // Comments allowed
private Integer commentDays;       // Days comments open
private String enclosureURL;       // Podcast attachment
```

Key Methods:

```
public void copyFrom(WeblogEntry entry, Locale locale)
public void copyTo(WeblogEntry entry)
public Timestamp getPubTime(Locale locale, TimeZone tz)
```

3.3 WeblogEntry (Entity)

Package: `org.apache.roller.weblogger.pojos`

Purpose: Blog entry domain model.

Key Properties:

```
private String id;
private String title;              // Entry title
private String text;               // Main content (HTML)
private String summary;            // Short summary
private String searchDescription; // Meta description
private String anchor;             // URL slug
private String creatorUserName;    // Author username
private Timestamp pubTime;         // Publication time
private Timestamp updateTime;      // Last update time
private PubStatus status;          // Publication status
private Boolean allowComments;     // Comments enabled
private Integer commentDays;       // Comment window
private Boolean pinnedToMain;      // Featured flag
private String locale;             // Language
private String contentType;        // Content type
```

Status Values (PubStatus enum):

- `DRAFT` – Saved but not visible
- `PUBLISHED` – Live on blog

- `SCHEDULED` – Publish at future date
- `PENDING` – Awaiting approval

Relationships:

- ManyToOne: `Weblog` (parent blog)
- ManyToOne: `WeblogCategory`
- ManyToOne: `User` (creator)
- OneToMany: `WeblogEntryComment` (comments)

3.4 WeblogEntryManager (Interface)

Package: `org.apache.roller.weblogger.business`
Purpose: Entry CRUD operations interface.
Key Methods:

```
void saveWeblogEntry(WeblogEntry entry);
WeblogEntry getWeblogEntry(String id);
List<WeblogEntry> getWeblogEntries(WeblogEntrySearchCriteria criteria);
void removeWeblogEntry(WeblogEntry entry);
List<WeblogCategory> getWeblogCategories(Weblog weblog);
void saveComment(WeblogEntryComment comment);
WeblogEntryComment getComment(String id);
List getComments(CommentSearchCriteria criteria);
void removeComment(WeblogEntryComment comment);
int removeMatchingComments(Weblog weblog, WeblogEntry entry,
    String search, Date start, Date end, ApprovalStatus status);
```

3.5 JPAWeblogEntryManagerImpl (Implementation)

Package: `org.apache.roller.weblogger.business.jpa`
Purpose: JPA implementation of WeblogEntryManager.
Key Operations:

- Persists entries to database
- Manages entry-tag relationships
- Handles category assignments
- Processes mediacast attachments

3.6 WeblogCategory (Entity)

Package: `org.apache.roller.weblogger.pojos`
Purpose: Category for organizing blog entries.
Properties:

```
private String id;
private String name;
private String description;
private String image;
```

4 Comment Management Classes

4.1 Comments (Action)

Package: `org.apache.roller.weblogger.ui.struts2.editor`

Purpose: Controller for managing comments (blog owner view).

Attributes:

```
private CommentsBean bean;        // Filter/action data
private CommentsPager pager;      // Paginated comments
private WeblogEntry queryEntry;   // Entry filter
private int bulkDeleteCount;      // Deletion counter
```

Key Methods:

```
public String execute()   // List all comments
public String query()     // Filter comments
public String update()    // Approve/disapprove/mark spam
public String delete()    // Bulk delete
private void loadComments()
public List getCommentStatusOptions()
```

Flow:

1. Load comments based on criteria
2. Display with approval status checkboxes
3. Blog owner marks approve/spam/delete
4. Update comment statuses
5. Save to database
6. Invalidate cache
7. Send notifications

4.2 CommentsBean (DTO)

Package: `org.apache.roller.weblogger.ui.struts2.editor`

Purpose: Holds comment filter and action data.

Properties:

```
private int page;                  // Current page
private String entryId;            // Filter by entry
private String searchString;       // Search text
private String startDateString;    // Date range start
private String endDateString;      // Date range end
private String approvedString;     // Status filter
private String ids;                // Comma-separated IDs
private String[] deleteComments;   // Delete checkboxes
private String[] approvedComments; // Approve checkboxes
private String[] spamComments;     // Spam checkboxes
```

4.3 WeblogEntryComment (Entity)

Package: `org.apache.roller.weblogger.pojos`

Purpose: Comment domain model.

Key Properties:

```
private String id;
private String name;          // Commenter name
private String email;         // Commenter email
private String url;           // Website URL
private String content;       // Comment text
private Timestamp postTime;   // Submission time
private ApprovalStatus status; // Approval status
private Boolean notify;       // Notify on reply
private String remoteHost;    // IP address
private String referrer;      // Referrer URL
private String userAgent;     // Browser info
private String plugins;       // Processing plugins
private String contentType;   // Content type
```

Status Values (ApprovalStatus enum):

- `APPROVED` – Visible on blog
- `PENDING` – Awaiting moderation
- `SPAM` – Marked as spam
- `DISAPPROVED` – Rejected by moderator

5 Rendering Engine Classes

5.1 Renderer (Interface)

Package: `org.apache.roller.weblogger.ui.rendering`

Purpose: Core interface for template rendering.

Method:

```
void render(Map<String, Object> model, Writer writer)
    throws RenderingException;
```

Description: The Renderer interface is the foundation of the rendering subsystem. Implementations take a map of model objects (data) and write the rendered output to a Writer.

5.2 RendererFactory (Interface)

Package: `org.apache.roller.weblogger.ui.rendering`

Purpose: Factory interface for creating Renderer instances.

Method:

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

Description: Each RendererFactory determines if it can handle a given template and returns an appropriate Renderer. Returns null if it cannot handle the template.

5.3 RendererManager (Utility)

Package: `org.apache.roller.weblogger.ui.rendering`
Purpose: Central manager coordinating templates and renderer factories.
Attributes:

```
private static List<RendererFactory> rendererFactories;
```

Method:

```
public static Renderer getRenderer(Template template,
    DeviceType deviceType);
```

Flow:

1. Load renderer factories from configuration
2. For each render request, iterate through factories
3. Return first matching Renderer
4. Throw exception if none found

5.4 VelocityRenderer (Implementation)

Package: `org.apache.roller.weblogger.ui.rendering.velocity`
Purpose: Renders templates using Apache Velocity template engine.
Attributes:

```
private Template renderTemplate;           // Original template
private DeviceType deviceType;             // Mobile/standard
private org.apache.velocity.Template velocityTemplate;
private org.apache.velocity.Template velocityDecorator;
```

Key Methods:

```
public void render(Map<String, Object> model, Writer out);
private void renderException(Map<String, Object> model,
    Writer out, String template);
```

Flow:

1. Initialize with Template
2. Load Velocity template
3. Convert model Map to VelocityContext
4. If decorator exists:
   - Render base to StringWriter
   - Put result in context as "decorator_body"
   - Render decorator to output
5. Else: Merge template directly to output
6. Handle errors with error-page.vm template

5.5 VelocityRendererFactory (Implementation)

Package: `org.apache.roller.weblogger.ui.rendering.velocity`
Purpose: Creates VelocityRenderer instances for Velocity templates.
Method:

```
public Renderer getRenderer(Template template,
    DeviceType deviceType);
```

Flow:

1. Check if template is not null
2. Get TemplateRendition for STANDARD type
3. Check if template language is VELOCITY
4. Create and return new VelocityRenderer
5. Return null for non-Velocity templates

6 Content Fetching & Serving Classes

6.1 WeblogRequestMapper (URL Router)

Package: `org.apache.roller.weblogger.ui.rendering`
Purpose: Routes incoming weblog requests to the appropriate servlet.
Key Methods:

```
void handleRequest(HttpServletRequest request,
    HttpServletResponse response);
String calculateForwardUrl(HttpServletRequest request,
    String handle, String locale, String context, String data);
boolean isWeblog(String potentialHandle);
boolean isLocale(String potentialLocale);
```

URL Patterns:

URL Pattern	Target Servlet
/<handle>/	PageServlet (home page)
/<handle>/entry/<anchor>	PageServlet (single entry)
/<handle>/page/<name>	PageServlet (custom page)
/<handle>/category/<cat>	PageServlet (category view)
/<handle>/tags/<tags>	PageServlet (tag view)
/<handle>/date/<date>	PageServlet (date archive)
/<handle>/feed/<type>	FeedServlet (RSS/Atom)
/<handle>/search	SearchServlet
/<handle>/comment/<entry>	CommentServlet

6.2 PageServlet (Servlet)

Package: `org.apache.roller.weblogger.ui.rendering.servlets`
Purpose: Main servlet for serving weblog pages.
Attributes:

```
private WeblogPageCache weblogPageCache;
private SiteWideCache siteWideCache;
private boolean processReferrers;
```

Key Methods:

```
void init(ServletConfig servletConfig);
void doGet(HttpServletRequest request, HttpServletResponse response);
void doPost(HttpServletRequest request, HttpServletResponse response);
void processHit(Weblog weblog);
boolean processReferrer(HttpServletRequest request);
```

Flow:

1. Parse WeblogPageRequest from HTTP request
2. Check cache for cached content
3. If not cached:
   - Load Weblog entity
   - Determine Template to use
   - Build model objects (PageModel, URLModel, etc.)
   - Get Renderer from RendererManager
   - Render template to output
   - Cache result
4. Track page hit
5. Process referrer for analytics
6. Return rendered content

6.3 FeedServlet (Servlet)

Package: `org.apache.roller.weblogger.ui.rendering.servlets`
Purpose: Serves RSS and Atom feeds for weblogs.
Feed Types:

- **entries** – Recent blog entries (RSS/Atom)
- **comments** – Recent comments (RSS/Atom)
- **media** – Recent media files

6.4 CommentServlet (Servlet)

Package: `org.apache.roller.weblogger.ui.rendering.servlets`
Purpose: Handles comment submissions on blog entries.
Features:

- Spam detection (Akismet, banned words, excess links)
- Comment throttling (GenericThrottle)
- IP ban list checking
- Email notifications to blog owner
- Comment preview support

Flow:

1. Check if IP is banned
2. Check throttle limits
3. Parse WeblogCommentRequest
4. Create WeblogEntryComment from form data
5. Run comment validators
6. Determine status (APPROVED, PENDING, SPAM)
7. Save comment to database
8. Send email notifications
9. Redirect to entry page with message

6.5 Model (Interface)

Package: `org.apache.roller.weblogger.ui.rendering.model`
Purpose: Base interface for all rendering models.
Methods:

```
String getModelName();
void init(Map<String, Object> params) throws WebloggerException;
```

6.6 PageModel (Model Implementation)

Package: `org.apache.roller.weblogger.ui.rendering.model`
Purpose: Provides data for rendering weblog pages.
Key Methods:

```
WeblogWrapper getWeblog();
WeblogEntryWrapper getWeblogEntry();
boolean isPermalink();
WeblogCategoryWrapper getWeblogCategory();
List<String> getTags();
Pager getWeblogEntriesPager();
WeblogEntryCommentForm getCommentForm();
String getRequestParameter(String paramName);
```

6.7 FeedModel (Model Implementation)

Package: `org.apache.roller.weblogger.ui.rendering.model`
Purpose: Provides data for rendering RSS/Atom feeds.
Key Methods:

```
WeblogWrapper getWeblog();
boolean getExcerpts();
String getCategoryName();
Pager getWeblogEntriesPager();
Pager getCommentsPager();
Pager getMediaFilesPager();
List<String> getTags();
```

6.8 SiteModel (Model Implementation)

Package: `org.apache.roller.weblogger.ui.rendering.model`
Purpose: Provides site-wide data across all weblogs.
Key Methods:

```
Pager getWeblogEntriesPager(int sinceDays, int length);
Pager getCommentsPager(int sinceDays, int length);
Pager getUsersByLetterPager(String letter, int sinceDays, int length);
Pager getWeblogsByLetterPager(String letter, int sinceDays, int
    length);
List<TagStat> getPopularTags(int sinceDays, int length);
List<Weblog> getNewWeblogs(int sinceDays, int length);
List<Weblog> getHotWeblogs(int sinceDays, int length);
long getCommentCount();
long getEntryCount();
long getWeblogCount();
long getUserCount();
```

7 Shared Infrastructure Classes

7.1 PersistenceStrategy (Interface)

Purpose: Database operations abstraction.
Methods:

```
void store(Object obj);
void remove(Object obj);
void flush();
```

```
Object find(Class clazz, Object id);
Query createQuery(String queryString);
```

7.2 User (Entity)

Purpose: User account model.

Properties:

```
private String id;
private String userName;
private String password;
private String emailAddress;
private String locale;
private String timezone;
private Date dateCreated;
private Boolean enabled;
```

7.3 WeblogPermission (Entity)

Purpose: User permissions on blogs.

Properties:

```
private String id;
private String actionsString;  // ADMIN, POST, EDIT_DRAFT
private short pending;
private Date datecreated;
```

7.4 IndexManager (Interface)

Purpose: Search indexing operations.

Methods:

```
void addEntryReIndexOperation(WeblogEntry entry);
void removeEntryIndexOperation(WeblogEntry entry);
void executeIndexOperations();
```

7.5 CacheManager

Purpose: Cache invalidation utility.

Methods:

```
public static void invalidate(Weblog weblog);
public static void invalidate(WeblogEntry entry);
public static void clear();
```

7.6 MailUtil

Purpose: Email notification utility.

Methods:

```
public static boolean isMailConfigured();
public static void sendPendingEntryNotice(WeblogEntry entry);
public static void sendEmailApprovalNotifications(
    List comments, MessageResources resources);
public static void sendNewCommentNotification(
    WeblogEntryComment comment);
```

8 Enumerations

8.1 PubStatus

Purpose: Publication status for blog entries.

```
public enum PubStatus {
    DRAFT,       // Saved but not visible
    PUBLISHED,   // Live on blog
    SCHEDULED,   // Publish at future date
    PENDING      // Awaiting approval
}
```

8.2 ApprovalStatus

Purpose: Approval status for comments.

```
public enum ApprovalStatus {
    APPROVED,      // Visible on blog
    PENDING,       // Awaiting moderation
    SPAM,          // Marked as spam
    DISAPPROVED    // Rejected by moderator
}
```

9 Interaction Flows

9.1 Flow 1: Create Blog

```
User -> CreateWeblog.execute()
  -> Display form

User submits -> CreateWeblog.save()
  -> myValidate() (check handle)
  -> new Weblog(handle, name, ...)
  -> WeblogManager.addWeblog()
    -> JPAWeblogManagerImpl.addWeblog()
      -> strategy.store(weblog)
      -> addWeblogContents()
        -> UserManager.grantWeblogPermission() (ADMIN)
        -> Create WeblogCategory entities
        -> Create WeblogBookmark entities
        -> MediaFileManager.createDefaultMediaFileDirectory()
      -> strategy.flush()
  -> SUCCESS
```

9.2 Flow 2: Create Entry

```
User -> EntryEdit.execute()
  -> new WeblogEntry() or load existing
  -> Display editor

User writes post -> EntryEdit.publish()
  -> setStatus(PUBLISHED)
  -> save()
    -> bean.copyTo(weblogEntry) (title, text, category, tags)
    -> weblogEntry.setPubTime(now)
    -> weblogEntry.setUpdateTime(now)
    -> WeblogEntryManager.saveWeblogEntry()
      -> JPAWeblogEntryManagerImpl.saveWeblogEntry()
        -> strategy.store(weblogEntry)
        -> strategy.flush()
    -> IndexManager.addEntryReIndexOperation()
    -> CacheManager.invalidate(weblogEntry)
    -> AutoPingManager.queueApplicableAutoPings()
  -> SUCCESS
```

9.3 Flow 3: Add Comment

```
Visitor submits comment -> Comment Handler
  -> new WeblogEntryComment()
  -> setName(name)
  -> setEmail(email)
  -> setContent(content)
  -> setPostTime(now)
  -> setWeblogEntry(entry)
  -> Spam Check
    -> if spam: setStatus(SPAM)
    -> else if moderation: setStatus(PENDING)
    -> else: setStatus(APPROVED)
  -> WeblogEntryManager.saveComment()
    -> strategy.store(comment)
    -> strategy.flush()
  -> if APPROVED: show on blog
  -> if PENDING: await moderation
  -> Send notifications
```

9.4 Flow 4: Render Blog Page

```
Visitor HTTP GET -> WeblogRequestMapper.handleRequest()
  -> Parse URL: /<handle>/entry/<anchor>
  -> isWeblog(handle) -> true
  -> calculateForwardUrl()
  -> Forward to PageServlet

PageServlet.doGet()
  -> Parse WeblogPageRequest
  -> Check WeblogPageCache for cached content
  -> If cached: return cached HTML
  -> If not cached:
```

```
    -> Load Weblog by handle
    -> Determine Template (entry, page, category, etc.)
    -> Build model objects:
      -> PageModel.init(request params)
      -> URLModel.init()
      -> UtilitiesModel.init()
    -> RendererManager.getRenderer(template, deviceType)
      -> VelocityRendererFactory.getRenderer()
        -> new VelocityRenderer(template, deviceType)
    -> Renderer.render(model, writer)
      -> VelocityRenderer.render()
        -> Convert model to VelocityContext
        -> velocityTemplate.merge(context, writer)
    -> Cache rendered HTML
  -> processHit(weblog) -> Update hit counter
  -> Return HTML to visitor
```

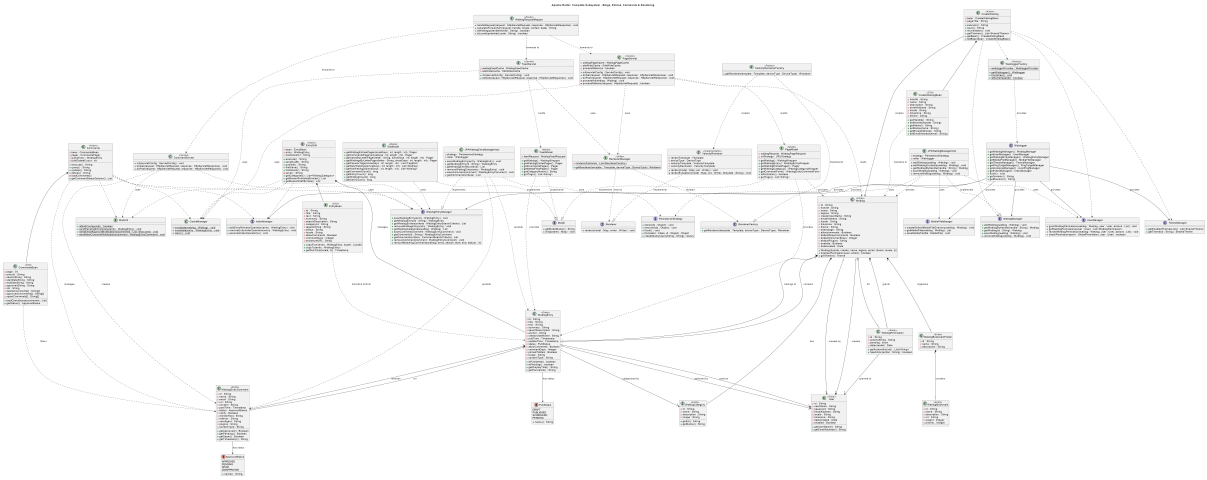

Figure 1: Enter Caption

10 UML Class Diagram