

# Weblog & Content Subsystem — Class Documentation

---



## CORE DOMAIN CLASSES (Data Objects)

### Weblog

**What it is:** Represents one complete blog website.

**Why relationships exist:**

It *owns* categories and bookmark folders because they belong to a blog. It *uses managers* to fetch posts and apply themes.

---

### WeblogCategory

**What it is:** A grouping label for blog posts (like Tech, Travel).

**Why relationships exist:**

It belongs to a Weblog and retrieves entries through `WeblogEntryManager` because categories don't store posts directly.

---

### WeblogEntry

**What it is:** A single blog post/article.

**Why relationships exist:**

It belongs to a blog and category, uses plugins to process content, calls managers to fetch data, and links to comments.

---

### WeblogEntryComment

**What it is:** A user's comment on a blog post.

**Why relationships exist:**

It is attached to a specific `WeblogEntry` because comments cannot exist without a post.

---

### WeblogBookmarkFolder

**What it is:** A folder that groups blog bookmarks (blogroll).

**Why relationships exist:**

It belongs to a blog and contains bookmarks.

---

## WeblogBookmark

**What it is:** A single external link saved in the blogroll.

**Why relationships exist:**

It belongs to a folder because bookmarks are organized hierarchically.

---

# BUSINESS / MANAGER LAYER

## WeblogManager (interface)

**What it is:** Service that creates, updates, and deletes blogs.

**Why relationships exist:**

It manages [Weblog](#) objects and interacts with [User](#) because blogs have owners.

---

## WeblogEntryManager (interface)

**What it is:** Service responsible for posts, comments, categories, tags, and statistics.

**Why relationships exist:**

It handles all content-related domain objects.

---

## BookmarkManager (interface)

**What it is:** Manages bookmark folders and links.

**Why relationships exist:**

It controls bookmark data instead of the Weblog doing DB work.

---

## UserManager

**What it is:** Handles users and permissions.

**Why relationships exist:**

Posts check permissions via users.

---

## ThemeManager

**What it is:** Controls blog themes and layout styles.

**Why relationships exist:**

Weblog uses it to determine how content is displayed.

---

## CORE SYSTEM CONTROLLER

### Weblogger (interface)

**What it is:** Central service hub of the application.

**Why relationships exist:**

Provides access to all managers so components don't create them directly.

---

### WebLoggerFactory

**What it is:** Bootstraps and provides the Weblogger instance.

**Why relationships exist:**

Used by domain classes to reach managers without tight coupling.

---

### WebloggerRuntimeConfig

**What it is:** Holds runtime system settings.

**Why relationships exist:**

Weblog and entries read configuration values from it.

---

## EXTENSIBILITY

### PluginManager

**What it is:** Controls execution of plugins.

**Why relationships exist:**

Used by entries to process content dynamically.

---

### WeblogEntryPlugin

**What it is:** A plugin that modifies entry content.

**Why relationships exist:**

Registered and executed via PluginManager.

---



## SUPPORTING DATA CLASSES

Class	Role
WeblogEntrySearchCriteria	Filters posts
CommentSearchCriteria	Filters comments
TagStat	Tag statistics
WeblogHitCount	Blog traffic data
User	Represents a system user
WebloggerException	Error handling

---



## Design Strengths (Improved)

- ✓ Clear separation between **data (POJOs)** and **logic (Managers)**
  - ✓ Centralized service access through **Weblogger**
  - ✓ Extensible architecture via **Plugin system**
  - ✓ Proper hierarchical ownership (Blog → Category → Entry → Comment)
  - ✓ Loose coupling using interfaces
  - ✓ Modular managers (each handles one domain)
- 



## Design Weaknesses (Expanded)

- ✗ **WeblogEntry** is a God Class — too many responsibilities
  - ✗ Repeated retrieval methods across managers
  - ✗ Service Locator pattern hides dependencies
  - ✗ Tight coupling to **WebLoggerFactory** inside domain objects
  - ✗ Many bidirectional relationships make it hard to follow flow
  - ✗ Business logic mixed inside domain models
-

# Final Summary

This subsystem follows a **service-oriented layered architecture** where:

- Domain objects store data
- Managers control logic and database access
- Weblogger acts as a central gateway
- Plugins allow extension
- Configuration is centralized

The design is **modular but complex**, making it powerful but harder to understand for new developers.