REGIONE BASILICATA

kai 開star

CMS Wizard: System and Programming

Ufficio società dell'Informazione
Via Vincenzo Verrastro 4 - V Piano 85100 Potenza (PZ)

Indice

Indice delle illustrazioni

1 Bibliografia

Titolo	File	Versione
Installazione CMF	KSTAREVOL-TEC-INST-CMF	**1.3**
Manuale Utente CMF	KSTAREVOL-TEC-MAN-CMF	**2.1**

2 Definizioni e acronimi

Titolo	File
CDA	**Content Delivery Application**
CMA	**Content Management Application**
CMS	**Content Management System**
RDBMS	**Relational Data Base Management System**
otype	**Forma abbreviata per "Object Type"**

3 La piattaforma Kaistar

3.1 Caratteristiche

KAISTAR è un sistema per la gestione e il delivery delle informazioni, siano esse strutturate o in formato documentale, e per la loro pubblicazione su Internet e Intranet. KAISTAR è basato su piattaforma J2EE (Java 2 Enterprise Edition) e database Oracle e MySQL, ha funzionalità estremamente avanzate e un'interfaccia utente user-friendly.

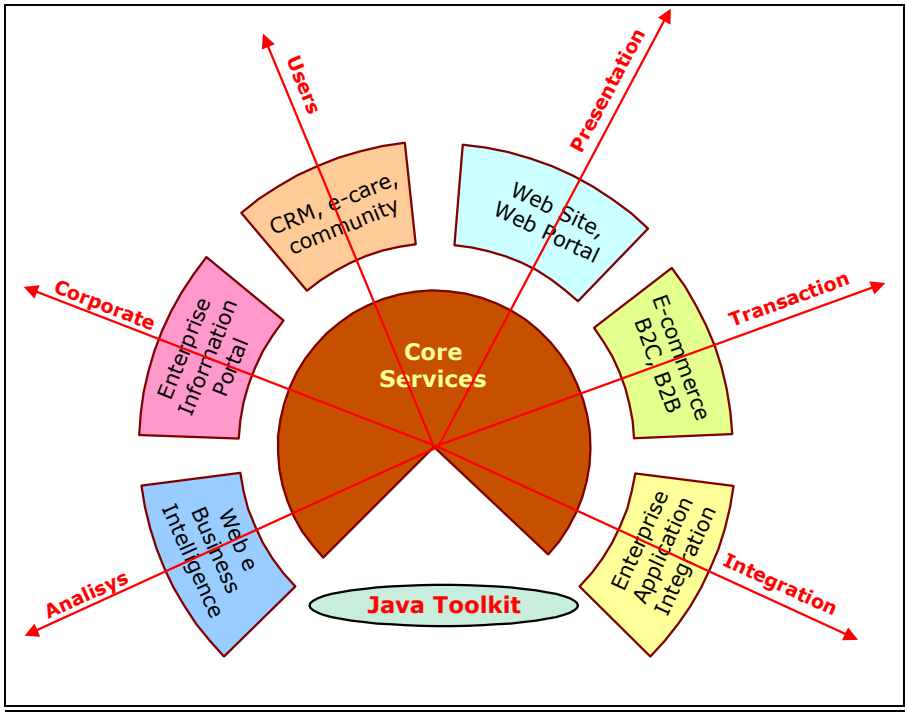

Figura 1: I servizi di gestione dell'informazione e le applicazioni

Sul sistema di gestione delle informazioni si poggiano applicazioni specifiche. La separazione architetturale tra i servizi di base e le applicazioni che le utilizzano consente la massima flessibilità nella distribuzione dell'informazione per usi diversi e nella presentazione grafica.

Le tecnologie utilizzate, in particolare la piattaforma J2EE, sono state scelte in quanto consentono una facile integrazione con prodotti di mercato che possono risultare necessari per verticalizzazioni spinte.

3.2 Content Management in KAISTAR

Con il termine **Content Management** (CM) si intende il processo in base al quale uno o più autori mantengono un repository costantemente aggiornato e facilmente utilizzabile di *contenuti digitali*.

Per **Content Delivery** (CD) si intende l'erogazione di tali contenuti su Internet (Web, Email, FTP) o, eventualmente, su altri canali di distribuzione.

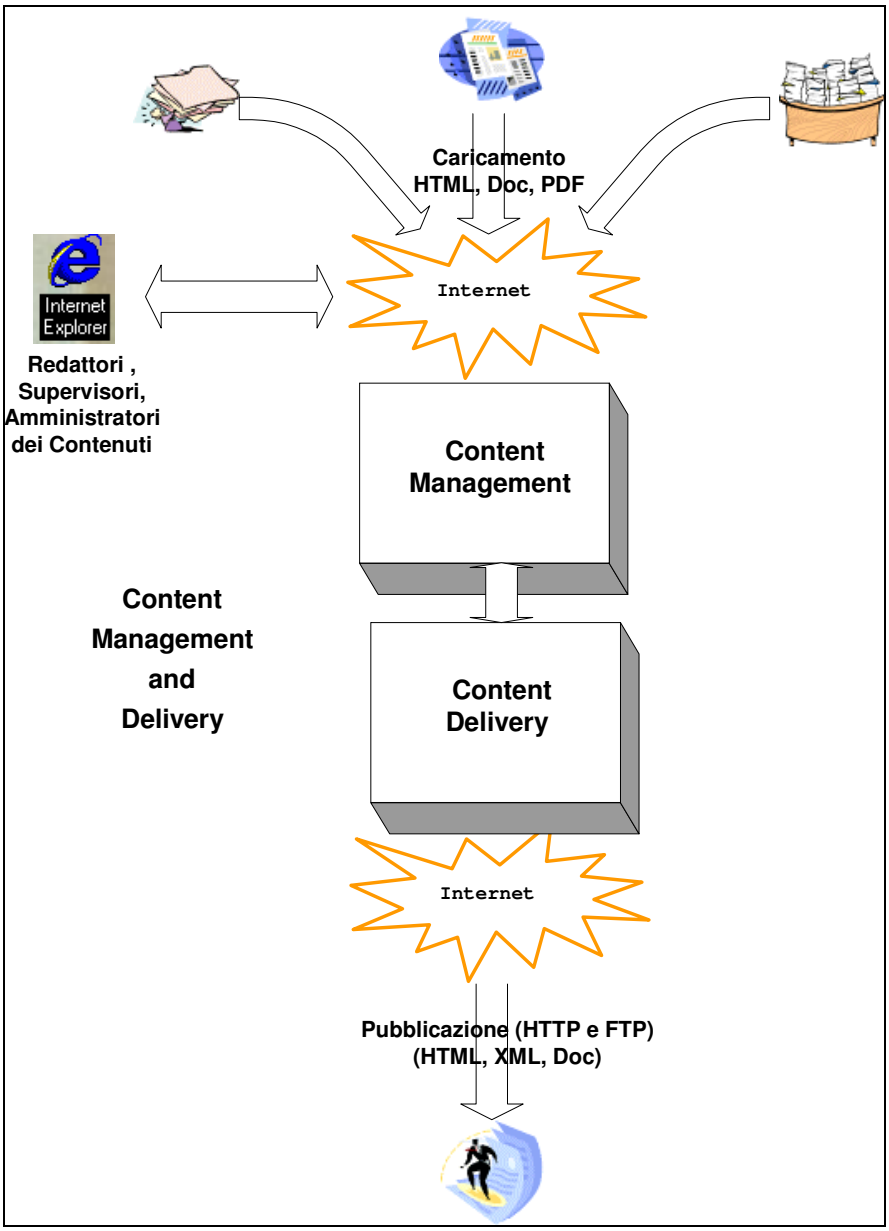

Figura 2: Il processo di Content Management e Delivery

Con il termine **contenuto digitale** si intende genericamente un qualsiasi contenuto informativo memorizzato in formato elettronico archiviato su file o in un Data Base Management System.

Esempi di contenuti digitali sono i tradizionali documenti realizzati con word processor, fogli elettronici, testi in formato più o meno strutturato (plain text, HTML, XML), ma anche immagini e oggetti multimediali come sequenze audio e video.

Logicamente il sistema è organizzato nelle seguenti componenti:

- Content Management Application: per l'inserimento e la manutenzione dei contenuti
- Content Repository: per lo storage ed il retrieval dei dati.
- Interfacce verso fonti date esterne ed applicazioni legacy.

- Content Delivery Application: per l'erogazione dei contenuti

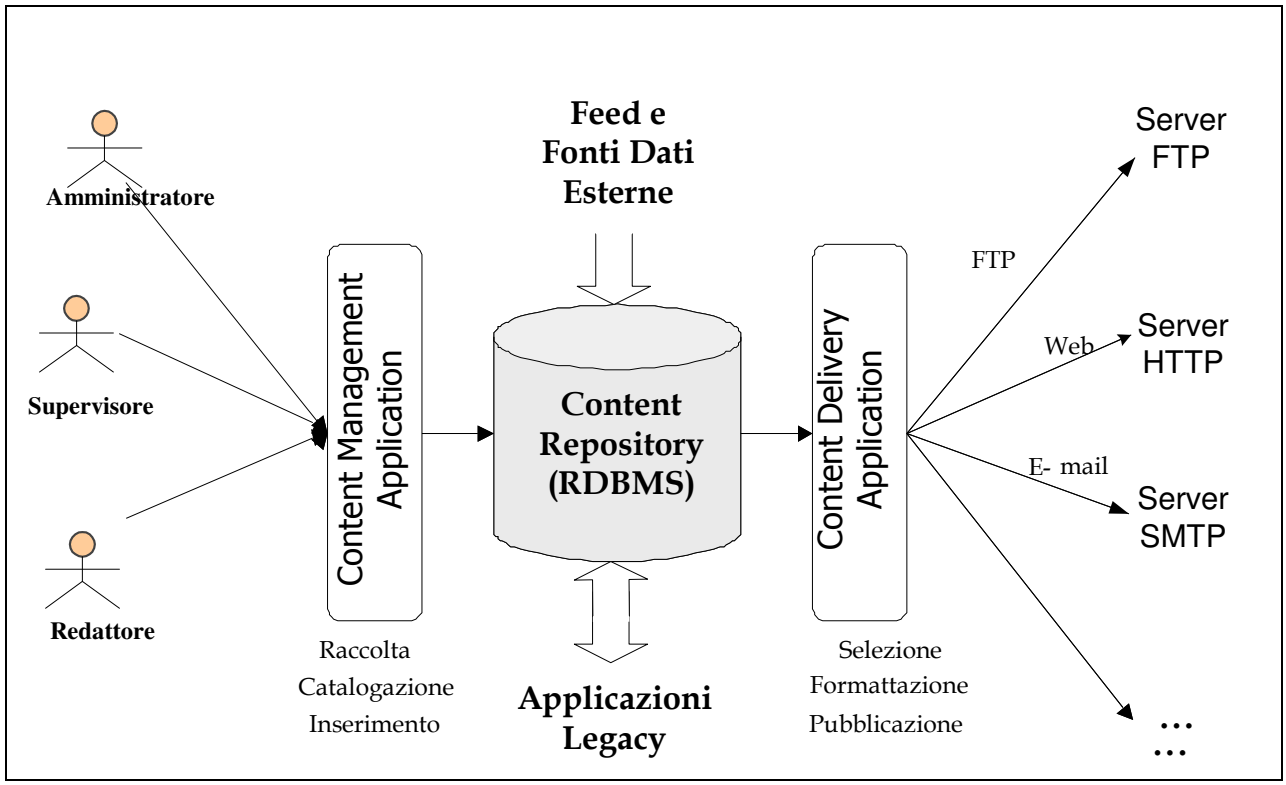

Figura 3: Disaccoppiamento Content Management e Delivery

Il nucleo del sistema è dato dal **Repository** che ospita i contenuti ed i relativi metadati secondo la struttura ed i formati richiesti dalle applicazioni. Il repository costituisce una "libreria di contenuti digitali" che funge da unico punto di raccolta di tutte le informazioni, e da unico punto di distribuzione ai diversi sistemi che fruiranno delle informazioni.

I contenuti del repository sono gestiti tramite una **Content Management Application** (CMA) che consente di inserire, modificare e cancellare i contenuti, nonché annotarli con metadati che ne massimizzino le possibilità di ricerca e di navigazione ipertestuale.

Al repository attingono una o più **Content Delivery Application** (CDA) che estraggono le informazioni e, dopo averle opportunamente trasformate nel formato desiderato (*rendering*), provvedono alla loro diffusione sui diversi canali di distribuzione (*publishing)*.

> La principale caratteristica di tale architettura è il disaccoppiamento tra la fase di raccolta, catalogazione e amministrazione delle informazioni e la fase di diffusione delle stesse (distribuzione o pubblicazione).

Vediamo di seguito le caratteristiche che tali componenti assumono in Kaistar.

3.2.1 Content Repository

3.2.1.1 Il modello dei contenuti

KAISTAR si fonda su una base di dati relazionale caratterizzata un modello dei contenuti estremamente flessibile. Tramite una semplice opera di configurazione, è possibile definire rapidamente uno o più repository di contenuti digitali eterogenei per conformarsi ai diversi requisiti applicativi di ogni specifico progetto.

A seguito della configurazione del repository, KAISTAR genera automaticamente una applicazione di content management web-based che fornisce un immediato accesso alle funzionalità amministrative ed editoriali, consentendo di cominciare a popolare da subito il sistema. In alternativa i contenuti possono essere facilmente aggregati da basi di dati esistenti tramite un meccanismo di feed basato su XML.

> Il Content Repository di KAISTAR è caratterizzato da uno schema estremamente flessibile che tramite una semplice opera di configurazione consente di definire nuovi tipi di contenuto e, successivamente, di modificarne arbitrariamente la struttura. L'applicazione di Content Management è automaticamente generata. Tutto ciò si traduce in tempi ridotti di realizzazione dei progetti e in una forte riduzione dei costi di gestione.

Gli elementi di base del modello applicato da KAISTAR sono gli **oggetti**, item informativi elementari o strutturati, di tipo diverso (testo, immagini, documenti) che, come le componenti di un Lego, concorrono nella definizione di strutture più complesse.

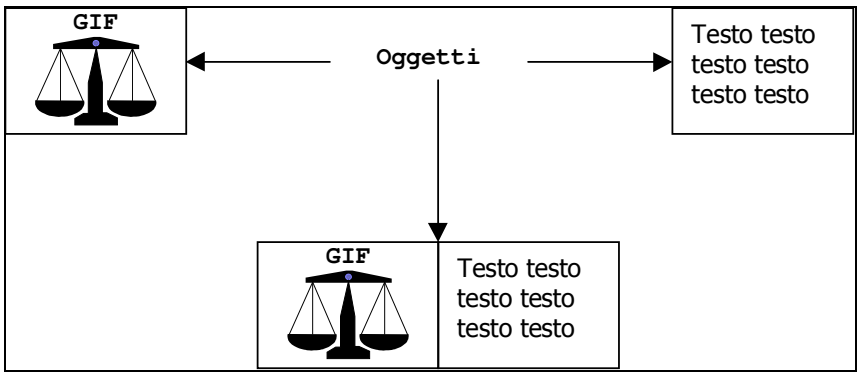

Figura 4. Oggetti elementari e strutturati

I **contenuti** sono tipicamente entità strutturate caratterizzate da un insieme di metadati (a es. la data di creazione/revisione/pubblicazione, l'autore, la lingua, le keywords, etc.) e da una parte informativa tipicamente costituita da una gerarchia di oggetti (**content items**).

Rispetto ad un semplice oggetto, un contenuto rappresenta un'entità "completa" (dal punto di vista dell'informazione che contiene) ed "autonoma", ovvero dotata di un proprio ciclo di vita (creazione, approvazione, pubblicazione, archiviazione, etc.). Un contenuto può concorrere alla creazione di diverse pubblicazioni ed essere oggetto di syndication (ovvero distribuito ad una terza parte).

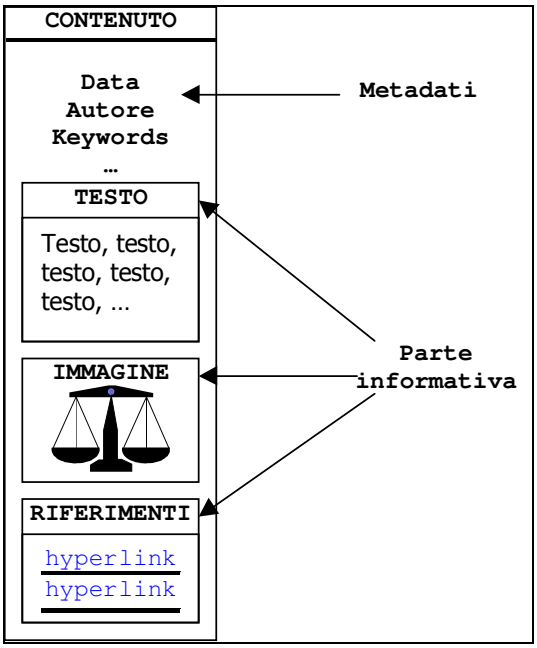

Figura 5. Contenuto

Il sistema consente di definire e gestire **contenuti compositi** ottenuti per aggregazione di contenuti più semplici, ognuno dotato dei propri metadati e di un ciclo di vita indipendente.

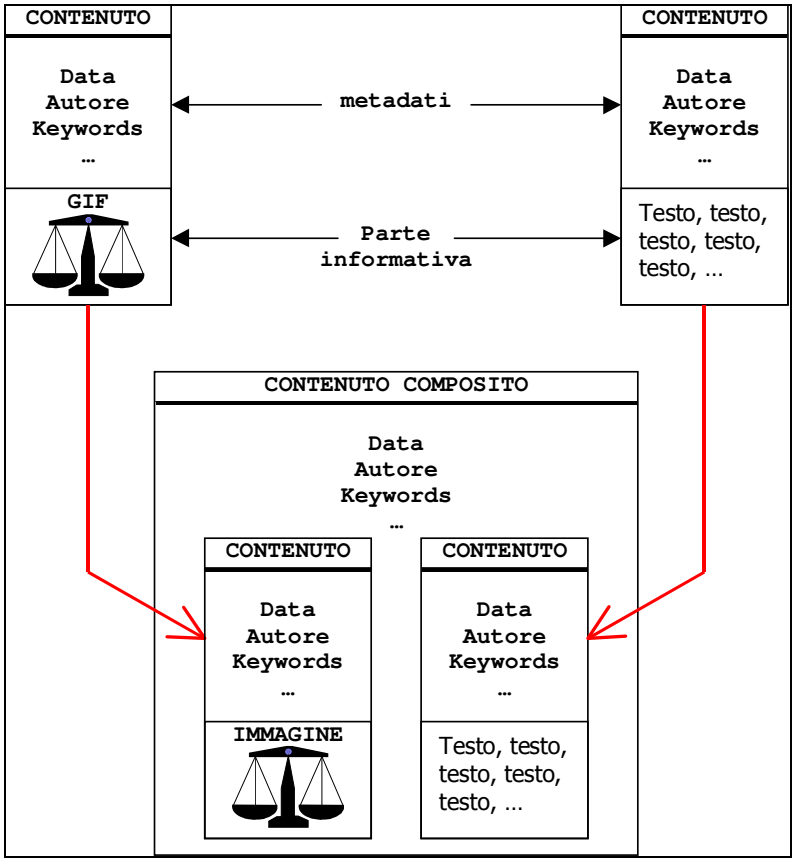

Figura 6. Contenuto Composito

In questo modo il sistema consente di realizzare strutture arbitrariamente complesse.

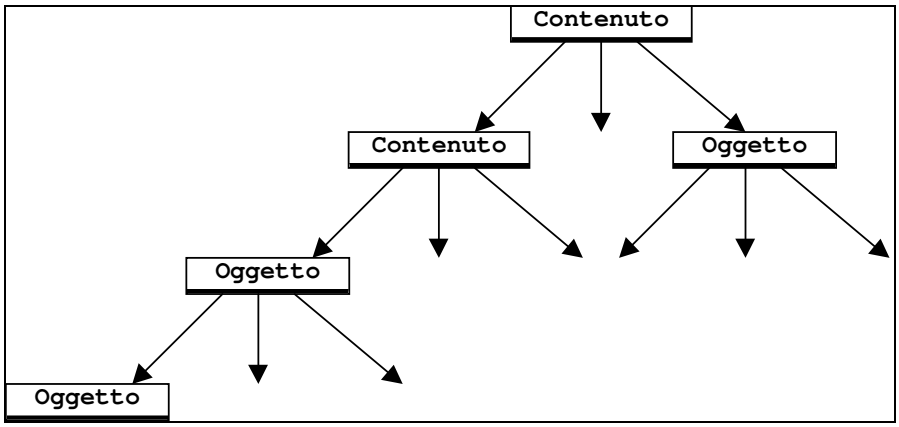

Figura 7. Gerarchia dei contenuti

Tecnicamente il meta-modello di KAISTAR è costituito dalle seguenti entità:

- Alla base stanno le generiche "entità infomative" denominate oggetti (**objects**).
- Un oggetto è individuato da un tipo (**object type**) e da un identificatore unico (**id**).
- Un object type è caratterizzato da un ben definito insieme di proprietà (**properties**) ognuna delle quali è un attributo elementare (non dotato di struttura) riconducibile a un tipo di dato base (ad es. intero, decimale, stringa, data, file, etc.).

  Ad esempio, è possibile definire il tipo "Articolo" caratterizzato dalle proprietà "Titolo", "Autore", "Data", etc., ed il tipo "Paragrafo" con un'unica proprietà "Testo".

- Un object type può intervenire nella definizione di relazioni di composizione con altri object type (**object type components**), dando così luogo a strutture gerarchiche. Per ciascuna componente è definita la *molteplicità*, ovvero il numero di istanze di oggetti associabili per mezzo di questa componente.

  Ad esempio, per specificare che ogni paragrafo può avere un'immagine ed un insieme di link di approfondimento è sufficiente assegnare al tipo "Paragrafo" una componente singola "Immagine" ed una componente multipla "Link". Analogamente è possibile associare una lista di "Paragrafi" ad un "Articolo".

- Su un object type è possibile definire una o più tassonomie (**object type taxonomies**) che determinano le categorie con cui classificare gli oggetti.

  In questo modo è possibile specificare che tutti gli oggetti di tipo "Articolo" possono essere classificati rispetto alla tassonima "Argomenti" (ad es. "politica", "economia", "sport", etc.) e rispetto alla tassonomia "Regione" (ad es."Abruzzo", "Basilicata", "Campania", etc.).

- Ad un object type è possibile associare degli specifici servizi (**object type services**) quali ad esempio il workflow, il locking (check-in/check-out) e l'auditing.

  In questo modo è possibile definire diversi tipi di workflow di approvazione, in base al tipo di oggetto, e decidere quali oggetti sono sottoposti ad auditing e quali no.

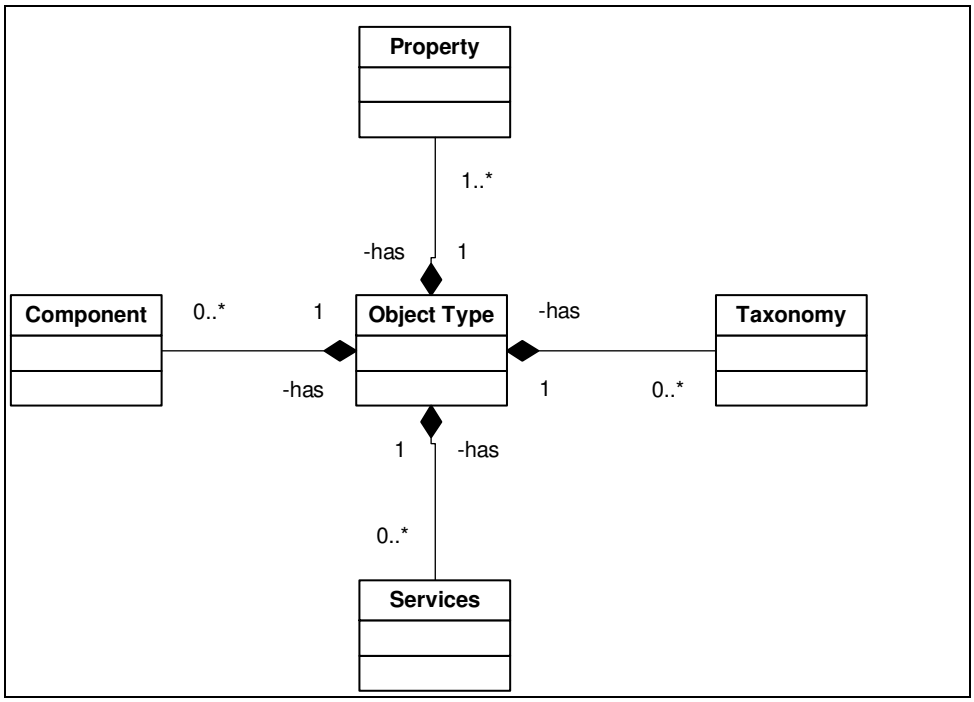

Figura 8: Elementi base del meta-modello

3.2.1.2 Building-block

KAISTAR è rilasciato con una libreria di object type general-purpose, da utilizzare come building block per la costruzione delle applicazioni.

- ***Content***. È l'object type principale che racchiude tutti i dati caratteristici di un qualsiasi contenuto ed i meta-dati necessari per supportare i meccanismi di pubblicazione: il titolo, l'autore, una descrizione, l'abstract, la lingua, le keyword, le date di creazione, pubblicazione e la durata del periodo di validità, la priorità, etc.
- ***Text***. Un generico frammento di testo eventualmente arricchito con tag di formattazione (es. HTML, XML, etc.). Nel caso HTML la redazione dei testi avviene in una finestra WYSIWYG dotata di toolbar che permette ai redattori di passare senza difficoltà dall'utilizzo di strumenti come Word a KAISTAR.
- ***Uri***. Utilizzato per memorizzare un Uniform Resource Identifier (ad esempio una URL) è caratterizzato da un protocollo e da un indirizzo.
- ***Link***. E' possibile definire link logici, ad altri contenuti dell repository, e link fisici (delle URI). E' possibile decidere la modalità di apertura del link (nella pagina o in un popup).
- ***Upload***. Consente di memorizzare sul repository uno stream di dati corrispondente al contenuto di un file. Fisicamente i dati vengono memorizzati sul repository iin un campo BLOB e distribuiti, tramite un meccanismo di code, su tutti i file system dei server di front-end (*deploy*).

- ***Image***. Oltre ad indicare il file che contiene l'immagine è possibile specificare ulteriori informazioni, ad es. il ridimensionamento orizzontale e verticale (rescaling), l'allineamento e quant'altro possa comportare la necessità di predisporre un rendering specifico dell'oggetto riferito (si noti come tali informazioni non si riferiscano a caratteristiche proprie dell'oggetto ma sono piuttosto relative alla modalità ed al contesto in cui esso deve essere presentato). E' inoltre possibile associare allo stesso oggetto versioni alternative della stessa imagine (alta risoluzione, bassa risoluzione, thumbnail).

La libreria di building-block è adeguata ai requisiti delle più comuni applicazioni di content management. Tuttavia il sistema è modulare e si presta all'implementazione di nuovi e object-type (solo in tal caso si renderà necessario l'intervento di un programmatore).

3.2.1.3 Portabilità dei contenuti

3.2.2 Content Management Application (CMA)

Al termine della configurazione del Repository KAISTAR genera **automaticamente** una CMA web-based (ovvero caratterizzata da un'interfaccia HTML, fruibile con un comune Web browser) di cui si descrivono di seguito le principali funzionalità.

3.2.2.1 Siti

KAISTAR consente la gestione contemporanea di diversi siti con redazioni autonome. I siti possono condividere i contenuti o essere completamente disgiunti. I redattori possono lavorare a uno o più siti.

La definizione dei siti viene fatta rapidamente configurando sul database i metadati che ne descrivono i contenuti. Terminata l'opera di configurazione, l'amministratore del sistema potrà creare gli account, assegnare i permessi agli utenti e definire la struttura del sito. Lo stesso modello può essere applicato a sezioni diverse di uno stesso sito; le sezioni, infatti, possono avere redazioni e contenuti indipendenti.

3.2.2.2 Utenti e permessi

Questa componente consente la creazione, modifica, eliminazione degli utenti della redazione. Il sistema dei permessi consente di gestire livelli di privilegio differenziati assegnando agli utenti diversi **ruoli** e **gruppi** (**community**). I ruoli servono a definire le autorizzazioni dell'utente, ovvero l'insieme delle azioni che costui può svolgere sulle diverse tipologie di contenuto. L'appartenenza ai gruppi determina invece l'insieme delle aree del sito su cui l'utente è autorizzato ad operare.

Queste funzionalità consentono di strutturare le redazioni in accordo all'organigramma aziendale e definire i permessi in base alle competenze professionali.

Sono definite le seguenti classi di utenti:

- **Redattore**. Acquisisce le informazioni dalle fonti e le inserisce nel sistema. Ha il controllo sull'inserimento e la modifica dei contenuti limitatamente ai tipi ed alle aree del sito di competenza.
- **Supervisore**. Il compito principale del supervisore è di visionare i contenuti inseriti o modificati dai redattori e decidere se pubblicarli o respingerli.

- **Amministratore**. Crea e assegna i ruoli agli utenti che compongono la redazione stessa. Detiene il privilegio di supervisore su tutti i tipi di contenuto definiti sul sito.

KAISTAR supporta un team geograficamente distribuito di operatori che intervengono con diversi ruoli e a diversi livelli, nel processo di sviluppo e manutenzione dei contenuti.

3.2.2.3 Contenuti e workflow

Il sistema editoriale, formato dall'insieme degli strumenti per la gestione e classificazione delle informazioni strutturate e dei documenti, è estremamente user-friendly in modo da fornire anche all'utente meno esperto (quindi privo di particolari conoscenze tecniche quali HTML, programmazione server side, etc.) strumenti di immediato utilizzo per effettuare le tipiche operazioni di aggiornamento e manutenzione dei contenuti.

La creazione del contenuto avviene secondo il **workflow** o processo definito sulla particolare tipologia del contenuto. Il workflow definisce i passi necessari che portano ad avere un contenuto redatto in modo completo e approvato, stabilendo altresì la possibilità che ha un utente, a seconda del ruolo ricoperto, di intervenire su di esso per controllarne l'evoluzione.

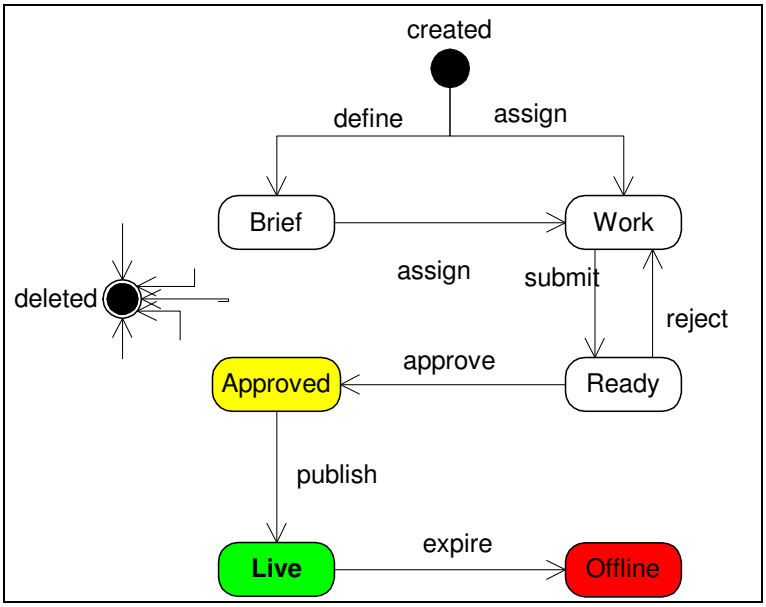

Figura 9: Esempio di workflow editoriale

Il workflow descrive l'insieme della fasi che definiscono il ciclo di vita dei contenti. Gli utenti, attraverso un'interfaccia Web-based intervengono esplicitamente per attivare i passi del workflow; il sistema gestisce e controlla la corretta applicazione di tali passi.

3.2.2.4 Tassonomie

KAISTAR utilizza un meccanismo di classificazione dell'informazione che usa una struttura ad albero di categorie (**tassonomia**). I redattori possono assegnare e modificare le categorie di un contenuto senza limiti sul numero e sulla complessità delle strutture tassonomiche. I contenuti e i documenti possono essere classificati per più categorie facenti parte di tassonomie indipendenti, consentendo così un utilizzo e una ricerca efficienti dell'informazione.

La classificazione dei contenuti attraverso le tassonomie ha un ruolo centrale in KAISTAR e ha molteplici applicazioni tra le quali la definizione di permessi su risorse per area tematica, ricerche mirate per tassonomia e/o categoria e non ultima la possibilità di "navigare" un albero tassonomico accedendo ai contenuti associati a ciascuna categoria.

> La classificazione tramite una tassonomia centralizzata abilita la distribuzione dell'informazione per più usi, l'aggregazione della stessa e l'integrazione tra sistemi eterogenei.

3.2.3 Content Delivery Application (CDA)

La pubblicazione dei contenuti avviene configurando per ogni sito o dominio la struttura gerarchica delle sezioni. Il contenuto di ciascuna sezione e delle relative pagine viene definito in base a regole flessibili basate sui tipi di contenuto disponibili e sulla loro classificazione.

Il sistema di delivery consente la realizzazione di template per la visualizzazione dinamica delle informazioni e dei documenti. La seguente figura illustra il modello di di una tipica CDA.

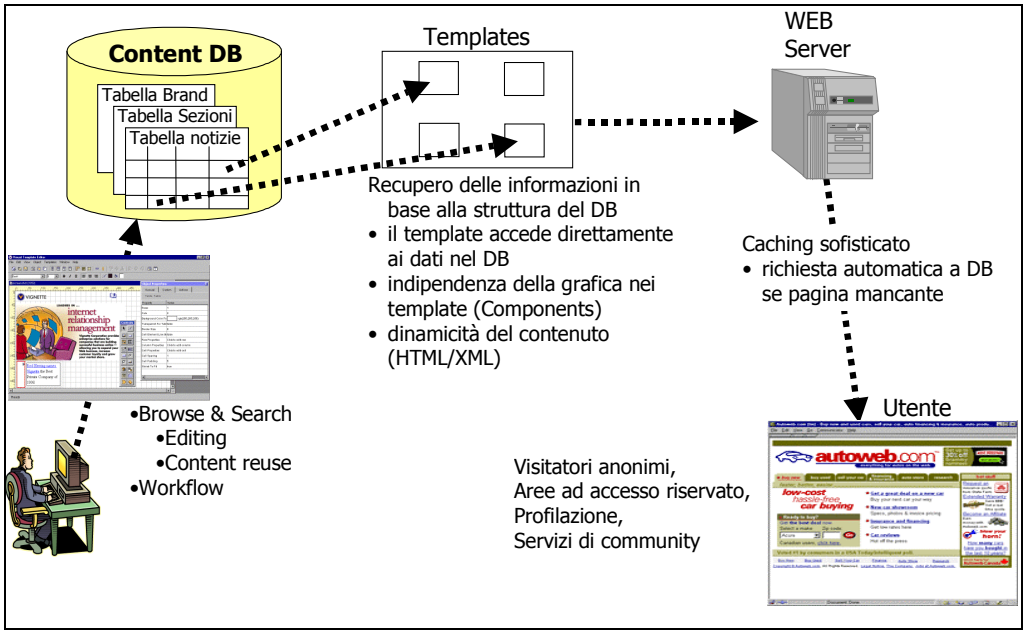

Figura 10: Applicazione di CDA

3.2.3.1 Caching

Per ottimizzare le prestazioni il sistema è dotato di meccanismi di cache. Una "**cache dati**" consente di ridurre il carico sul database memorizzando il risultato delle query più frequenti. Una "**cache risultati**" consente di memorizzare pagine intere o componenti di pagina riducendo il carico dovuto all'elaborazione. La permanenza delle informazioni nella cache può essere limitata nel tempo o illimitata. Nel secondo caso è

cura del sistema editoriale di notificare automaticamente il gestore della cache in modo che, alla modifica di un contenuto, i risultati che da esso dipendono vengano invalidati e di conseguenza rigenerati.

3.2.3.2 Rendering multi-formato

3.2.3.3 Ricerca

Il sistema offre una funzionalità di **ricerca** su tutti i contenuti e documenti; la ricerca può essere effettuata sulla base delle **categorie** usate per la classificazione, di campi specifici di ciascun tipo di dato, del testo del documento o della struttura dati in modalità **full-text**.

La tecnologia di indicizzazione utilizzata è Lucene Java (http://lucene.apache.org/), un progetto open source, disponibile per il download gratuito.

3.3 Programmazione

3.3.1 Model API

L'accesso ai contenuti è mediato dalle **Model API** (package `it.ksolutions.kcmf.model`)

Il package definisce delle interfacce e delle classi di supporto per astrarre il DataModel del Kaistar:

- la classe **`ModelManager`** è il punto di partenza applicativo, è il factory del Model. Il metodo statico **`ModelManager.getModel()`** gestisce la creazione di un *Concrete-Model*;
- l'interfaccia **`Model`** astrae il DataModel del Kaistar e consente di accedere a:
    - *Metadati*
      definizione dei tipi di contenuti (ObjectType), delle loro Proprietà e Componenti, delle Tassonomie e Relazioni tra ObjectTypes. Es: `model.getObjectTypeInfo(otypeId)`
    - *Contenuti*
      istanze di ObjectType. Ogni contenuto è identificato dalla coppia <tipo,id>. Il metodo `model.getObject(otypeId, objId)` ritorna un handle per accedere ad un contenuto, alle sue proprerties, componenti e tassonomizzazioni. Es: `content.getTitle()`, `content.getComponent("IMAGE")`

      E' possibile creare nuovi contenuti. Es: `model.newObject(otypeId)`

      Oppure selezionarne un insieme rispetto a determinati criteri tramite un selector Es: `model.getObjectSelector()`

- l'interfaccia **`ObjectTypeInfo`** astrae la definizione di un tipo di contenuto. Oltre a ritornare gli attributi specifici come id, nome, label, ritorna la lista dei descrittori dei metadati associati all'ObjectType:
  - **`ObjectTypePropertyInfo`** - definizione delle property
  - **`ObjectTypeComponentInfo`** - definizione delle componenti
  - **`ObjectTypeTaxonomyInfo`** - definizione delle tassonomie
  - **`ObjectTypeRelationInfo`** - definizione delle relazioni con altri tipi
  - **`ObjectTypeTranslationInfo`** - definizione delle traduzione ammissibili

  Per ognuna di queste collezioni è possibile accedere ad una specifica istanza identificandola col nome o con l'id, oppure scorrere la collezione stessa tramite un Iterator secondo l'ordine attribuito in fase di definizione (position). Es. `otype.getComponent("IMAGE")`, `otype.iterateComponentByPosition()`.

- l'interfaccia **`CmfObject`** rappresenta un generico contenuto. Consente di accede alle proprietà e di navigare la struttura del contenuto stesso
  - getObjectTypeId() - ritorna l'identificativo dell'ObjectType
  - getId() - ritorna l'identificativo
  - getTitle() - ritorna il titolo
  - getSubtitle() - ritorna il sottotitolo
  - getCategories() - ritorna le tassonomizzazioni
  - getStatus() - ritorna lo stato corrente
  - getComponents(name) - ritorna i componenti con il nome specificato
  - getComponent(name@position) - ritorna il componente alla posizione specificata

- l'interfaccia **`ObjectSelector`** consente di selezionare un insieme di contenuti rispetto a specifici criteri e facilita l'estrazione di '*finestre*' di risultati utili in particolare per supportare meccanismi di paginazione.
  - setObjectTypes() - imposta i tipi di contenuto da selezionare
  - setStatus() - imposta gli stati dei contenuti da selezionare
  - setLimit() - imposta il numero massimo di contenuti da selezionare
  - setOrderBy() - imposta il criterio di ordinamento della selezione
  - setAndCategories() - imposta le tassonomie (AND) per la selezione

- setOrCategories() - imposta le tassonomie (OR) per la selezione
- select() effettuala la selezione
- isValid() determina se la selezione è stata completata con successo
- size() ritorna il numero di contenuti selezionati
- fetch() predispone l'estrazione di una finestra di risultati
- next() scorre la selezione dei risultati
- getProperty(name) ritorna il valore della property del contenuto corrente
- getObject() ritorna un handle al contenuto corrente

3.3.1.1 Esempi METADATI

Elenco delle property e delle componenti di un ObjectType:

```
String otypeId = "...";

Model model = ModelManager.getModel();

ObjectTypeInfo otype = model.getObjectTypeInfo(otypeId);

System.out.println("Properties:");
for (Iterator i = otype.iteratePropertyInfoByPosition(); i.hasNext(); ) {
    ObjectTypePropertyInfo property = (ObjectTypePropertyInfo)i.next();
    System.out.println(property.getName());
}

System.out.println("Components:");
for (Iterator i = otype. iterateComponentInfoByPosition (); i.hasNext(); ) {
    ObjectTypeComponentInfo component = (ObjectTypeComponentInfo)i.next();
    System.out.println(component.getName());
}
```

3.3.1.2 Esempi CONTENUTI

Recupero di un contenuto, noti ObjectType e Id:

```
String otypeId = "...";
String objId   = "...";

// Accedere al Model
Model model = ModelManager.getModel();

// recuperare un contenuto specifico
CmfObject content = model.getObject(otypeId, objId);

System.out.println("Content:" +
    "\n  otype : " + content.getObjectTypeInfo().getName() +
    "\n  id    : " + content.getId()) +
    "\n  title : " + content.getTitle()) +
    "\n  status: " + content.getStatus()
);
```

Elenco di un insieme di contenuti tramite ObjectSelector:

```
String otypeId = "...";

Model model = ModelManager.getModel();

ObjectSelector selector = null;
try {
    // inizializzazione ed esecuzione selezione
    selector = model.getObjectSelector()    // istanza sel selettore
    .setObjectTypes(otypeId)                // criterio: tipo di contenuto
    .setStatus("LIVE")                      // criterio: stato dei contenuti
    .setOrderBy("LASTMODIFIEDDATE DESC")    // criterio: ordinamento dei risultati
    .setLimit(10)                           // criterio: numero massimo risultati
    .select();                              // esecuzione selezione

    // check: selezione valida
    if (selector.isValid()) {
        // scorrimento selezione
        ObjectTypeInfo otype = model.getObjectTypeInfo(otypeId);
        while (selector.next()) {
            System.out.println(
                "\n  Content:" +
                "\n    id     = " + selector.getProperty(otype.getPrimaryKeyField() +
                "\n    title  = " + selector.getProperty(otype.getTitleField()) +
                "\n    date   = " + selector.getProperty(otype.getDateField()) +
                "\n    status = " + selector.getProperty("STATUS")
            );
        }
    }
}
catch (Exception ex) {
    // TODO
}
finally {
    // IMPORTANTE: chiusura selettore
    if (selector != null) selector.close();
}
```

Creazione di un contenuto:

```
String otypeId = "...";

Model model = ModelManager.getModel();

CmfObject content = model.newObject(otypeId);

content.setProperty("CNT_DATE", "09/09/2009");
content.setProperty("CNT_TITLE", "Grave incendio a Calci");
content.setProperty("CNT_SUBTITLE", "Ieri intorno alle 18 un disastroso incendio" +
                                    " si è esteso dalla Verruca a Caprona." +
                                    " Certa l'origine dolosa");
content.setProperty("CNT_SUMMARY", "Dalle 18 di ieri pomeriggio un gravissimo incendio" +
                                   " si è sviluppato nella zona di Nicosia. Le fiamme" +
                                   " di quasi 30 metri si sono estese poi nell'area" +
                                   " di Caprona e Crespignano, fino a lambire con" +
                                   " importanti ondate di fumo Calci e il suo centro" +
                                   " abitato. Il fronte del fuoco si è allungato per" +
                                   " circa 5 chilometri, in direzione di Uliveto," +
                                   " distruggendo più di 200 ettari di bosco.
content.setStatus("LIVE");

content.save();
```

Creazione di una component di un contenuto:

```
String otypeId = "...";
String objId   = "...";

Model model = ModelManager.getModel();

CmfObject content = model.getObject(otypeId, objId);

CmfObject cntImage = content.addComponent("IMAGE");
cntImage.setProperty("UPL_TITLE", "La torretta di Caprona avvolta dalle fiamme");
// altre property

content.save();
```

Cancellazione di un contenuto:

```
String otypeId = "...";
String objId   = "...";

Model model = ModelManager.getModel();

CmfObject content = model.getObject(otypeId, objId);

content.delete();
```

3.3.2 Programmare sulla CDA (Content Delivery Application)

La presentazione dei contenuti su un sito è soggetta a diversi fattori quali: grafica, accessibilità, layout, strategie comunicative, compatezza. Non è quindi conveniente definire degli schemi rigidi per la presentazione.

Tuttavia ci sono alcuni pattern ricorrenti nella presentazione dei contenuti:

- INDEX — Lista, eventualmente paginata, dei contenuti di un certo tipo
- HIGHLIGHT — Rimandi brevi ssimi a contenuti da mettere in evidenza. Es. lancio di news
- HEADLINE — Breve descrizione del contenuto. Es Sommario di un articolo con immagine
- DETAIL — Presentazione completa di un contenuto

La piattaforma consente di definire, all'interno della stessa installazione, diversi siti che condividono lo stesso tipo di contenuti e ognuno dei quali può essere specializzato, relativamente alla presentazione, adottando un proprio layout definito tramite un figlio di stile (Cascading Style Sheets) e specializzando i tiles di presentazione relativi ad ogni contesto.

Nella CDA di esempio `kcda-multisite.war`, sono presenti dei tiles generici per la presentazione dei contenuti per ognuno dei pattern previsti: `generic-index.jsp`, `generic-highlights.jsp`, `generic-headlines.jsp`, `generic-detail.jsp`

Tuttavia, per ogni ObjectType e per ogni sito, è possibile specializzare la presentazione per ognuno dei contesti.

Il package `it.ksolutions.kcda.multisite` raggruppa un insieme di classi e interfacce per supportare il recupero di contenuti sia in relazione al contesto di presentazione che in relazione a filtri quali sito, sezione e tassonomizzazioni:

- La classe **`SiteHelper`** è il punto di partenza applicativo e agisce anche da cache per minimizzare l'accesso al database:
  - `clearSiteHelpers()` — azzera la cache
  - `getAllSiteHelpers()` — Ritorna tutti i SiteHelper definiti nel database.
  - **`getSiteHelperByDomain(String domain)`** — Ritorna il SiteHelper relativo al sito corrente. Il "domain" è una chiave identificativa sel sito così come definita lato CMA all'atto della creazione del sito.

  Inoltre, oltre a ritornare una serie di property del sito corrente, consente di recuperare velocemente i metadati dei tipi di contenuto (ObjectType) del sito:
  - `getDomain()` — Ritorna la chiave identificativa per il sito
  - `getTitle()` — Ritorna il titolo del sito

- `getSubtitle()` Ritorna il sottotitolo del sito
- `getCSS()` Ritorna lo style-sheet da utilizzare per il sito
- **`getObjectTypeHelper(String otypeId)`** Ritorna l'ObjectTypeHelper relativo all'id specificato.

- La classe **`ObjectTypeHelper`** è di supporto sia alla lettura dei metadati dell'ObjectType che alla gestione della presentazione dei contenuti nei diversi contesti:
  - `getLabel()` Ritorna la stringa che descrive l'ObjectType
  - `getTitleField()` Ritorna il nome della property che rappresenta il titolo
  - `getSubtitleField()` Ritorna il nome della property che rappresenta il sottotitolo
  - `getDateField()` Ritorna il nome della property che rappresenta la data

  Per il supporto ai contesti di presentazione tramite i TileHelper:
  - `getTileIndex()` tile per il contesto di presentazione INDEX
  - `getTileHighlights()` tile per il contesto di presentazione HIGHLIGHT
  - `getTileHeadlines()` tile per il contesto di presentazione HEADLINE
  - `getTileDetail()` tile per il contesto di presentazione DETAIL

  Per il supporto alla selezione dei contenuti di un ObjectType:
  - `getSelector()` Ritorna un ObjectSelector opportunamente inizializzato e pronto per iterare la selezione

- La classe **`TileHelper`** supporta la gestione della presentazione dei contenuti nei diversi contesti:
  - `getURI()` Ritorna la jsp da utilizzare per il rendering
  - `getCondition()` Ritorna la condizione di filtro per la selezione
  - `getOrderBy()` Ritorna il criterio di ordinamento per la selezione
  - `getSize()` Ritorna il numero di elementi della selezione (o per pagina)
  - `isPaging()` Determina se la presentazione deve essere paginata

3.3.2.1 Realizzare una lista di oggetti

```
String siteKey = "...";       // request.getXxx()
String otypeId = "...";       // request.getParameter()

SiteHelper site = SiteHelper.getSiteHelper(siteKey);

ObjectTypeHelper otype      = site.getObjectTypeHelper(otypeId);
String           otypeLabel = otype.getLabel();

ObjectSelector selector = otype.getSelector(otype.getTileIndex());

<div id="index">
    <h1><%=otypeLabel%></h1>
<%
    if (selector != null && selector.isValid()) {
        while (selector.next()) {
            String _id       = selector.getProperty(otype.getPrimaryKeyField());
            String _title    = selector.getProperty(otype.getTitleField());
            String _subtitle = selector.getProperty(otype.getSubtitleField());
            String _date     = selector.getProperty(otype.getDateField();
            String _summary  = getAbstract(selector.getProperty("CNT_BODY"));
            String _url      = urlDetail(otypeId, _id);
%>
    <div class="box">
        <h2><a href="<%= _url %>"><%= _title %></a></h2>
        <p class="date"><%= _date %></p>
        <p class="subtitle"><%= _subtitle %></p>
        <p><%= _summary %></p>
    </div>
<%
        } // end-while(selector)
    } // end-if()
%>
</div>
```

3.3.2.2 Recupero di un singolo oggetto

```
String siteKey = "...";       // request.getXxx()
String otypeId = "...";       // request.getParameter()
String objId   = "...";       // request.getParameter()

SiteHelper site = SiteHelper.getSiteHelper(siteKey);

ObjectTypeHelper otype      = site.getObjectTypeHelper(otypeId);
String           otypeLabel = otype.getLabel();

CmfObject content = ModelManager.getModel().getObject(otypeId, objId);
String _title    = content.getTitle();
String _subtitle = content.getProperty(otype.getSubtitleField());
String _date     = content.getProperty(otype.getDateField());
String _summary  = content.getProperty("CNT_SUMMARY");

<div id="detail">
    <h1><%=_title%></h1>
    <p class="date"><%= _date %></p>
    <p class="subtitle"><%= _subtitle %></p>
    <p><%= _summary %></p>
</div>
```

3.3.2.3 Recupero dei componenti di un oggetto

Sezione in costruzione

3.3.2.4 Recupero delle sezioni

Sezione in costruzione

3.3.2.5 Lista di oggetti categorizzati per sezione

Sezione in costruzione

3.3.2.6 Utilizzo attributo id e parentId

Sezione in costruzione

3.3.2.7 Utilizzo del tag per la specifica di un URL

Sezione in costruzione

3.3.2.8 Recupero di un parametro di configurazione

Sezione in costruzione