# POLITECNICO DI MILANO

**Polo Regionale di Como**

**Facoltà di Ingegneria dell'Informazione**

**Corso di Studi in Ingegneria Informatica**


# PATTERN PER LA REALIZZAZIONE DI COMUNITÀ WEB 2.0 CON WEBRATIO


**Tutor universitario:**
**Prof. Piero Fraternali**

**Elaborato finale di:**
**Emmanuele De Andreis**
**matr. 665580**



**A.A. 2009-2010**

*A mio nonno, che per primo ha creduto nel fatto che l'informatica sarebbe stato il mio futuro, quando ancora pochi sapevano cosa fosse l'informatica, e che avrebbe voluto assistere a questo momento.*

*A mia moglie, che mi ha dato la fiducia ed il supporto senza i quali non avrei mai potuto raggiungere questo traguardo.*

# Premessa

# Sommario

**Premessa** ..... **3**

**Introduzione** ..... **8**

*Il problema del design dei progetti software* ..... *8*
*Storia dei pattern* ..... *9*
*Identificazione dei pattern sociali* ..... *10*
*Il web 2.0* ..... *11*
*Le comunità virtuali* ..... *13*

**L'architettura di una comunità** ..... **15**

*Persone* ..... *15*
Il valore dell'appartenenza ..... 15
Pattern relativi alle persone ..... 15
Profilo ..... 15
Pattern: Misuratore di completamento ..... 16
Reputazione ..... 17
Progettare un sistema di reputazioni efficace ..... 18
Pattern: Valutazione del grado di competitività ..... 19
Pattern: Collezione di benemerenze ..... 21
Pattern: Etichette identificative ..... 23
Pattern: Categorizzazione di merito a livelli ..... 24
Pattern: Conteggio punti ..... 24
Pattern: Il gruppo dei migliori X ..... 24
Pattern: Classifiche ..... 24
Pattern: Premi temporali ..... 24
Pattern: Statistiche ..... 24
Pattern: Referenze tra pari ..... 25
Pattern: Premi tra utenti ..... 25
*Contenuto* ..... *26*
Classificazione ..... 26
Classificazione fatta dal proprietario ..... 27
Tassonomie o categorie ..... 27
Folksonomie o tag ..... 27
Collegamento tra folksonomie e tassonomie ..... 28
Pattern: Assegna il tag ad un oggetto ..... 28
Dove e quando? ..... 30
Pattern: Geo-Tagging e Geo-Mashing ..... 31
Pattern: Calendario ..... 32
Classificazione fatta dalla comunità e feedback ..... 34
Pattern: Scrivi una recensione o invia commento ..... 34

Pattern: Valutare un oggetto ....35
Diffusione ....37
Esportazione ....38
Importazione ....38
Privacy ....38
Condivisione ....38
Pattern: Condividi ....38
*Relazione* ....*40*
*Navigazione* ....*41*
Navigazione dalla home page ....41
Navigazione a partire dai contenuti selezionati ....42
Pattern di navigazione ....42
Pattern: Nuvola di tag ....42
*Comunicazione* ....*44*
Comunicazione tra membri ....44
Comunicazioni di sistema ....44

**WebRatio Community Patterns ....45**

*Una comunità per la pubblicazione di racconti* ....*45*
*Il modello dei dati* ....*46*
Il profilo ....46
Entità profilo dell'utente ....46
Entità attributi del profilo ....47
Entità sesso ....47
Il contenuto ....48
Entità Contenuto ....48
Entità Tag ....50
Entità Valutazioni ....50
Entità Tipo contenuto ....50
*Modellazione in Webratio* ....*51*
Descrizione dei modelli ....51
Dati utilizzati ....51
L'implementazione WEBML ....51
La visualizzazione sulla pagina ....51
Modello accesso e registrazione utenti ....52
L'implementazione WEBML ....52
Visualizzazione sulla pagina dell'accesso utenti registrati ....53
Visualizzazione sulla pagina della registrazione nuovi utenti ....53
Modello profilo utenti ....53
L'implementazione WEBML ....53
La visualizzazione sulla pagina ....55
Modello gestione contenuto ....55
L'implementazione WEBML ....55
La visualizzazione sulla pagina ....56
Modello modifica contenuto e assegna tag ....58
Dati utilizzati ....58
L'implementazione WEBML ....58

La visualizzazione sulla pagina ....60
Modello nuvola di tags ....60
Descrizione dell'implementazione ....60
Dati utilizzati ....61
L'implementazione WEBML ....62
La visualizzazione sulla pagina ....63
Modello geomapping (Google Maps) ....64
Descrizione dell'implementazione ....64
Dati utilizzati ....64
L'implementazione WEBML ....65
La visualizzazione sulla pagina ....66
Modello valutare un oggetto ....67
Descrizione dell'implementazione ....67
Dati utilizzati ....68
L'implementazione WEBML ....68
La visualizzazione sulla pagina ....69
Modello condividi ....70
L'implementazione WEBML ....70
La visualizzazione sulla pagina ....71
*Relazione tra pattern, modello WebMl e modello dati ....72*

**Conclusioni e futuri sviluppi .... 73**

*Un progetto open source ....73*

**Appendice A Realizzare unità personalizzate in WebRatio .... 75**

*Le unità personalizzate (custom unit) ....75*
Introduzione ....76
Per cominciare ....76
Unit.xml ....78
Pagina generale (General page) ....78
Pagina sotto elemento (Sub-elements) ....78
Logica (Logic) ....80
Disposizione (Layout) ....81
Documentazione (Documentation) ....81
Versione (Version) ....81
La cartella logic ....81
Il file Input.template ....81
Il file Output.template ....82
Il file Logic.template ....83
Il file WebModel.template ....83
Il codice Java ....84
Il codice che contiene lo stato (Unit Java bean) ....84
Il codice che definisce la logica (Unit Java Service) ....84
Lettura dei parametri statici ....85
Lettura dei parametri dinamici ....85
Esecuzione ....87

La view: Lo stile, ovvero gestire l'output (Layout template) ....87
Quando qualcosa va storto....88
Debug ....92
Attivazione tomcat in modalità debug....92
Attivazione WebRatio in modalità debug ....92
Problemi comuni, soluzioni comuni ....94
Evitare una doppia chiamata al metodo execute....94

**Riferimenti .... 96**

*Bibliografia....96*
*Linkografia ....98*

# Introduzione

## Il problema del design dei progetti software

Progettare software è una delle attività più complesse che si possa immaginare. Questo non solo perché è una disciplina relativamente recente, in cui le pratiche e le metodologie di progettazione sono in continua evoluzione e rivoluzione. Il motivo principale è che il design è considerato da molti progettisti un "*Wicked Problem*" [21], temine che potremmo tradurre in prima approssimazione come "*Problema complesso*" e che può essere definito più precisamente nel seguente modo "*un problema talmente complesso che può essere chiaramente enunciato solo risolvendolo completamente, o almeno in parte*". La definizione riportata è di Horts Rittel e Melvin Webber [1]. Il paradosso implica che un problema di questo tipo deve essere risolto almeno due volte. La prima volta per essere definito, la seconda per produrre finalmente una soluzione che funzioni.

Per fare un esempio di cosa sia un problema di tipo complesso possiamo pensare al ponte Tacoma Narrow. La mattina del 7 novembre 1940 il Tacoma Narrows Bridge crollò, "abbattuto" dal vento sostenuto che soffiava attraverso Puget Sound ad una cinquantina di chilometri a sud di Seattle negli Stati Uniti. Era aperto da soli quattro mesi.

*Il ponte Tacoma Narrow, un esempio di problema complesso*

Dopo il crollo la Federal Works Agency stabilì una commissione d'indagine con tecnici quali Othmar Ammann e Theodore Von Karman che scagionò il progettista, osservando che, se le pec-

che del ponte erano ovvie a uno sguardo retrospettivo, il progetto rispondeva a ogni criterio accettabile nella pratica. Nel drammatico crollo non ci furono per fortuna danni a persone o feriti. Per i successivi venticinque anni non si costruirono più ponti sospesi.

Questo è un ottimo esempio di problema complesso. Senza prima aver costruito il ponte (risolto il problema) gli ingegneri progettisti non riuscivano ad immaginare che l'aerodinamicità di una struttura dovesse essere considerata fino a tal punto.

L'effetto che tale crollo ebbe nel mondo accademico e professionale fu enorme, grazie al fatto che l'intero evento fu filmato fin dal suo inizio, l'interpretazione delle cause innescanti il crollo si è arricchita negli anni grazie agli innumerevoli studi svolti. Oggi i fenomeni aerodinamici e aeroelastici sono ben documentati e studiati e nell'ambito dell'ingegneria civile sono attivi corsi universitari quali: dinamica delle strutture, ingegneria del vento, aeroelasticità delle strutture, aerodinamica applicata, discipline che si occupano in modo approfondito dell'interazione vento/struttura.

La differenza fondamentale tra i progetti che si studiano a scuola e quelli reali è che normalmente gli esercizi scolastici non sono complessi, ed hanno generalmente una risoluzione di tipo lineare.

Nei problemi reali l'oggetto da modellare è di solito estremamente complesso. Come è dunque possibile progettare adeguatamente un prodotto software o un sito web complesso senza muoversi per tentativi?

L' ipotesi è se sia possibile sfruttare le soluzioni complesse ideate e provate da altri, ed evitare così di risolvere il problema due volte.

La risposta a questa complessità si trova nello studio e nella applicazione dei pattern, piccole soluzioni ai problemi comuni che sono contemporaneamente soluzione e definizione del problema e che forniscono le linee guida su come implementare e inserire concetti e pratiche consolidate all'interno di una realizzazione complessa ed articolata.

# Storia dei pattern

L'idea di un modello di linguaggio (pattern) sembra applicarsi in maniera naturale a svariati e complessi compiti di ingegneria. È stata particolarmente influente nel campo dell'ingegneria del software, dove i modelli di progettazione sono stati utilizzati per documentare la conoscenza collettiva nel settore, anche se l'idea originale non è così recente e risale al mondo dell'architettura con le filosofie di Christopher Alexander. Alexander, un architetto austriaco naturalizzato statunitense, ha scritto il libro *A Pattern Language* [1] dove descrive un linguaggio, un insieme di regole o schemi per le modalità di progettazione e di costruzione di città, edifici, e altri spazi umani. L'approccio è ripetibile e lavora a vari livelli di scala. Alexander dice che "*ogni pattern descrive un problema che si verifica più e più volte nel nostro ambiente, e*

*quindi descrive il nucleo della soluzione a questo problema, in modo tale che è possibile utilizzare questa soluzione un milione di volte, senza mai farlo allo stesso modo due volte*".

L'idea di costruire con un linguaggio modello è stato adottato dal settore software per computer nel 1987, quando Ward Cunningham (noto per avere introdotto il concetto di Wiki) e Kent Beck hanno iniziato a sperimentare con l'idea di applicare i pattern alla programmazione.

Questo approccio prese piede e, nel 1995, è stato pubblicato il libro *Design Patterns: Elements of Reusable Object- Oriented Software* di Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides (conosciuti come la *Gang of Four*)[6].

Nel 1997, Jenifer Tidwell ha pubblicato una raccolta di modelli d'interfaccia utente (pattern) per la comunità degli esperti dell'interazione uomo-macchina (HCI) fondata sulla premessa che, catturando la saggezza collettiva di progettisti esperti, si contribuisce alla formazione dei progettisti inesperti e si fornisce alla comunità nel suo insieme un vocabolario comune per la discussione. L'evoluzione di tale sito e del suo lavoro è diventato il libro *Designing Interfaces*, pubblicato nel 2005 da O'Reilly Media [20].

Diversi autori hanno pubblicato raccolte dedicate al Web, tra cui Martijn van Welie, un sostenitore da lunga data di modelli nel campo della progettazione dell'interazione (*interaction design*), che ha a sua volta stimolato il team di Yahoo! a pubblicare porzioni della loro biblioteca interna di pattern di interazione nel 2006 [28].

Storicamente, lo sviluppo interfaccia utente è stato considerato alla stregua di un'attività di progettazione creativa, piuttosto che un processo sistematico di ingegneria. Tuttavia con l'evolversi delle applicazioni, sempre più presenti in ogni situazione e con il moltiplicarsi dei dispositivi su cui è possibile fruire contenuti multimediali, la progettazione e lo sviluppo di interfacce utente è diventato sempre più complesso. Così alle interfacce utente è richiesto di adattarsi dinamicamente ai vari contesti e variazioni dell'ambiente. Da ciò emerge la necessità di affrontare lo sviluppo di una progettazione strutturata. Approcci basati su modello hanno il potenziale per creare le basi per una metodologia sistematica di tipo ingegneristico per lo sviluppo di interfaccia utente [17].

Un esperimento interessante, anche se non precisamente legato ai pattern di design quanto piuttosto alla organizzazione di comunità online in genere, è stato compiuto nel 2009 dal Community Architecture Team di Red Hat. L'idea alla base di questo esperimento è quella di sfruttare il mezzo stesso che si studia come veicolo di raccolta informazioni. Prodotto finale è il libro, consultabile online, "The Open Source Way", che descrive il modo di gestire una comunità di sviluppatori open source nel modo in cui la stessa comunità ha suggerito [21].

# Identificazione dei pattern sociali

Negli ultimi quindici anni, abbiamo assistito alla proliferazione di diffusione della tecnologia Internet in tutto il mondo e sono stati immersi nella creazione di strumenti e di esperienze in-

terattive per aiutare le persone a loro modo di navigare le informazioni, trovare altre persone, e creare propri luoghi sul web.

Abbiamo visto l'ascesa e la caduta della prima ondata, il boom dot.com, e abbiamo sperimentato in prima persona l'esplosione del Web 2.0 e social media come sia ai progettisti e ai partecipanti. Queste connessioni e l'evoluzione degli strumenti elettronici e sociali stanno cambiando il nostro modo di interagire l'un l'altro.

Nuove tendenze dimostrano come gli strumenti informatici possano essere progettati e semplificati al fine di aiutare le persone ad ampliare le loro esperienze in internet con gli altri. Questi modelli sociali di comportamento e le interfacce che utilizzano sono emersi e continuano a evolversi per trovare il modo migliore per unire le persone. Modelli sociali sono i componenti e pezzi di interattività, che sono i mattoni di esperienze sociali [4].

Proveremo qui ad elencare alcune delle migliori pratiche e dei principi che abbiamo emersi da centinaia di siti e applicazioni con caratteristiche sociali. Essi sono i modelli emergenti d'interazione che sono diventati il metodo standard per gli utenti di interagire con i loro contenuti e con le persone che più contano per loro.

Con la crescente aspettativa degli utenti di esperienze senza soluzione di continuità, è importante per i progettisti analizzare gli standard emergenti per capire come una esperienza di un sito e le sue interazioni influenzino le aspettative per il prossimo sito. Lavorando con gli standard e le migliori pratiche emergenti, principi e modelli d'interazione, il progettista si fa carico dell'onere di capire come l'applicazione funziona, lasciando all'utente la possibilità di concentrarsi sulle proprietà uniche delle esperienze sociali che sta costruendo.

Cos'è dunque un pattern? Prendiamo a prestito la definizione da [4] p.10:

> *Soluzioni di progettazione d'interfaccia e componenti interattivi*
> *per un problema noto in un contesto specifico che sia ripetibile e di successo.*

I pattern sono quindi utilizzati come mattoni. Sono componenti fondamentali dell'esperienza dell'utente e descrivono i processi di interazione. Essi possono essere combinati con altri modelli, nonché altri elementi di interfaccia e contenuti per creare un'esperienza interattiva degli utenti.

# Il web 2.0

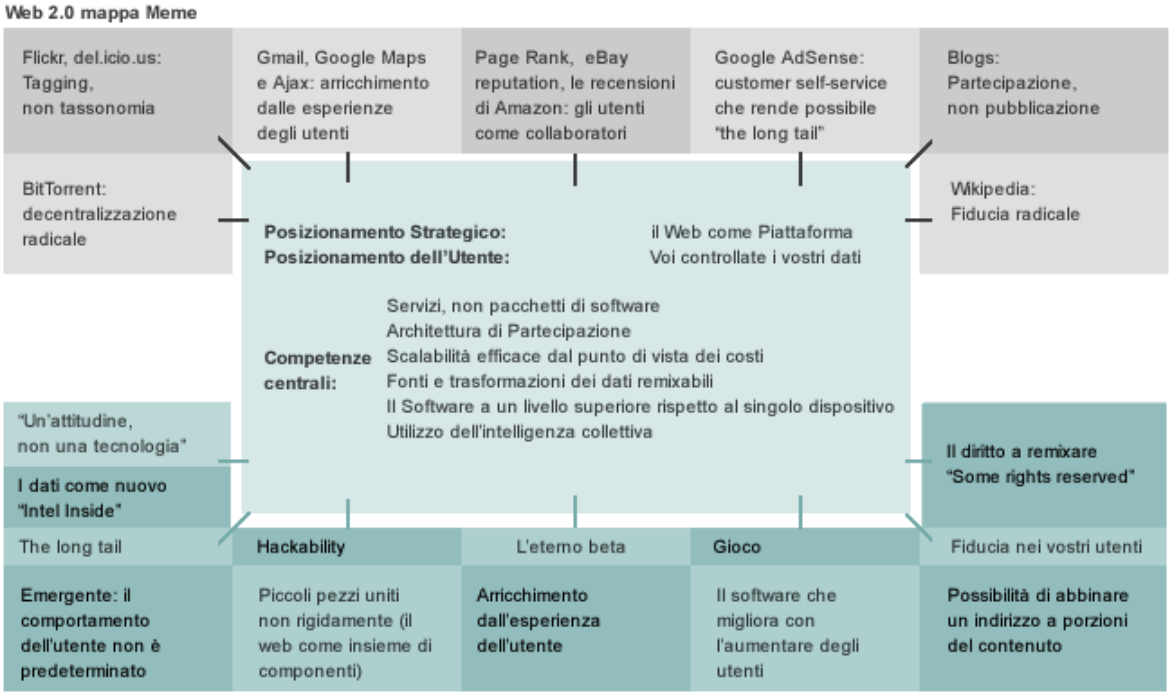
Il Web 2.0 è un termine utilizzato per indicare genericamente uno stato di evoluzione di Internet (e in particolare del World Wide Web), rispetto alla condizione precedente. Si tende ad indicare come Web 2.0 l'insieme di tutte quelle applicazioni online che permettono uno spiccato livello di interazione sito-utente (blog, forum, chat, sistemi quali Wikipedia, Youtube, Facebook, Myspace, Twitter, Gmail, Wordpress, Tripadvisor ecc.).

Oggi siamo di fronte più a un cambiamento culturale che a un'innovazione tecnologica com'era stata Internet anni fa (nell'1.0), e soprattutto perché in linea di massima i "tecnologi" diventano matti quando una cosa è eterea, impalpabile, culturale e devono per forza incatenarla in definizioni ed espressioni logiche o numeriche. Per i non addetti ai lavori la notazione 1.0/2.0 è una tecnica utilizzata per identificare e classificare le diverse versioni di un software nel corso del suo ciclo di vita, un po' come nelle automobili (1.0 = Punto prima versione, 1.1 = Punto prima versione con qualche optional in aggiunta e correzione, 2.0 = Punto nuova versione).

Web 2.0 è un termine coniato da Tim O'Reilly e Dale Dougherty (della O'Reilly Media, casa editrice americana specializzata in pubblicazioni riguardanti le nuove tecnologie e Internet in particolare) nel 2004 e fa riferimento ai cosiddetti servizi Internet di seconda generazione quali siti di social networking, wiki, strumenti di comunicazione e folksonomy che enfatizzano la collaborazione e la condivisione tra utenti [8]. O'Reilly Media, in collaborazione con MediaLive International, ha utilizzato tale termine per intitolare una serie di conferenze che hanno decretato la popolarità (e le critiche) di questo termine ormai diventato una buzzword in alcune community tecniche e di marketing.

La paternità è correttamente da attribuire a Tim O'Reilly che in un articolo, descrive come stiano iniziando a delinearsi nuove tipologie di servizi e dell'evoluzione del Web, in particolare nel sociale, che coinvolgono direttamente le persone. Tra le osservazioni chiave, vale la pena ricordare che dopo la bolla il Web stranamente si presentava più attivo e importante che mai, con nuove applicazioni e siti messi a disposizione degli utenti, dando il più tangibile tra i segni di vitalità dimostrando che le aziende sopravvissute alla bolla avevano alcune caratteristiche comuni, come a delineare una sorta di trend, una corrente di pensiero nel Web design. O'Reilly osservava che evidentemente il processo di selezione economica naturale aveva risparmiato quegli esemplari più forti, con basi più solide e dotati di alcuni elementi comuni nel proprio DNA Web. Questo rendeva possibile iniziare a cercare di capire quali potessero essere queste caratteristiche distintive. Il Web era rinato e chi operava in questo ecosistema doveva per forza aver sviluppato questi nuovi tratti somatici. Questo è principalmente il motivo per cui è difficile definire il Web 2.0, in quanto nato come etichetta dell'appartenenza a un certo insieme di specie aziendali aventi caratteristiche comuni.

O'Reilly nel suo articolo descrive il Web 2.0 come centro attorno al quale gravitano gli altri elementi creando una sorta di sistema solare. La metafora è intrigante e utile a trovare un collante tra i diversi elementi, anche se il suo limite, che è lo stesso di tutto il manifesto, è quello di non dare una vera indicazione di quale sia il modello di riferimento da usare per comprendere le relazioni tra i diversi fenomeni.




*La celebre Meme Map di O'Reilly che in qualche modo mette in relazione i diversi concetti sottesi a quello principe di Web 2.0. (immagine tratta da [8])*

# Le comunità virtuali

Una comunità virtuale o comunità online è, nell'accezione comune del termine, un insieme di persone interessate ad un determinato argomento, o con un approccio comune alla vita di relazione, che corrispondono tra loro attraverso una rete telematica, oggigiorno in prevalenza Internet, e le reti di telefonia, costituendo una rete sociale con caratteristiche peculiari. Tale aggregazione non è necessariamente vincolata al luogo o paese di provenienza; essendo questa una comunità online, chiunque può partecipare ovunque si trovi con un semplice accesso alle reti, lasciando messaggi su forum (Bullettin Board), partecipando a gruppi Usenet (Newsgroups o gruppi di discussione), o attraverso le chat room (Chiacchierate in linea) e programmi di instant messaging (messaggistica istantanea).

La diffusione esplosiva di Internet dalla metà degli anni novanta ha favorito la proliferazione di comunità virtuali sotto forma di servizi di social networking e comunità online. La natura di tali Comunità è varia. Comunità virtuali possono utilizzare le tecnologie Web 2.0 e pertanto sono state descritte come Comunità 2.0, sebbene il concetto di comunità online risalga ai primi anni settanta su sistemi come PLATO e successivamente su USENET.

Le Comunità online dipendono dall'interazione sociale e dallo scambio tra utenti online. Questo sottolinea l'elemento di reciprocità del contratto sociale non scritto tra i membri della Comunità.

Un tipo particolare di comunità virtuale è rappresentato dai servizi di social network. Un servizio di Rete sociale, consiste di una struttura informatica che gestisce nel Web le rete basata su

relazioni sociali. La struttura è identificata per mezzo di un sito web di riferimento del social network.

Secondo la definizione data dagli studiosi Boyd-Ellison si possono definire social network sites quei servizi web che permettono: la creazione di un profilo pubblico o semi-pubblico all'interno di un sistema vincolato, l'articolazione di una lista di contatti, la possibilità di scorrere la lista di amici dei propri contatti. Attraverso ciò questi servizi permettono di gestire e rinsaldare online amicizie preesistenti o di estendere la propria rete di contatti.

Il punto più avanzato della ricerca sulle reti sociali attraverso internet è rappresentato però dalla teoria del socio-semantic web (s2w), progetto destinato a "risemantizzare" il web, aggiungendo un approccio pragmatico usando nel semantic browsing classificazioni euristiche e ontologie semiotiche. In base a questi criteri la massa di informazione e produzione culturale immessa nel web viene interconnessa, producendo così una attiva connessione tra gli utenti proattivi della rete.

Un importante sviluppo delle reti sociali è rappresentato dalla possibilità di creare da parte di chiunque ne abbia le competenze (sviluppatori con linguaggi solitamente proprietari) applicazioni orientate alla comunità degli iscritti; tale famiglia di applicazioni beneficiano della rete di contatti e delle informazioni individuali degli iscritti (es. Facebook, MySpace, ABCtribe sono stati i primi) e rendono per taluni i social network i sistemi operativi web del futuro.

Per entrare a far parte di un social network online occorre costruire il proprio profilo personale, partendo da informazioni come il proprio indirizzo email fino ad arrivare agli interessi e alle passioni (utili per le aree "amicizia" e "amore"), alle esperienze di lavoro passate e relative referenze (informazioni necessarie per il profilo "lavoro").

A questo punto è possibile invitare i propri amici a far parte del proprio network, i quali a loro volta possono fare lo stesso, cosicché ci si trova ad allargare la cerchia di contatti con gli amici degli amici e così via, idealmente fino a comprendere tutta la popolazione del mondo, come prospettato nella teoria dei sei gradi di separazione del sociologo Stanley Milgram (1967), la cui validità anche su Internet è stata recentemente avvalorata dai ricercatori della Columbia University.[1].

Diventa quindi possibile costituire delle community tematiche in base alle proprie passioni o aree di business, aggregando ad esse altri utenti e stringendo contatti di amicizia o di affari.

# L'architettura di una comunità

## Persone

L'utente della comunità non è un semplice account a cui concedere o negare l'accesso ad aree pubbliche o private del sito, ma rappresenta un individuo (avatar o alter ego) che interagisce con il sito, crea nuovo contenuto o arricchisce e manipola quello esistente. Questa è prerogativa dei siti web 2.0, che non sono collezioni di elementi passivi, ma piuttosto dinamici e permettono ai membri di interagire con essi.

### Il valore dell'appartenenza

Associati al concetto di membro si trovano anche alcuni dati che identificano la partecipazione del membro alle attività della comunità e che in qualche modo ne classificano il valore relativo rispetto alla sua appartenenza.

Possiamo identificare due possibili valori:

- L'indice di reputazione (valore numerico assoluto)
- Lo stato (che è un'interpretazione relativa legata al contesto dell'indice di reputazione)

### Pattern relativi alle persone

Esistono dei sistemi che cercano di mediare ed automatizzare il processo, prendere nota delle azioni dei membri, catturare la reazione della comunità nei confronti dei comportamenti e mantenere una lista aggiornata delle azioni passate.

### Profilo

Un profilo utente è una fonte di dati relativa a tutte le informazioni dell'utente che possono essere impiegate per determinare il comportamento del sistema.

In aggiunta all'identificazione di base (per esempio, il nome utente o altri elementi anagrafici), un profilo utente può contenere informazioni molto differenti, a seconda del contesto e delle necessità del sistema.

Ad esempio:

- caratteristiche personali (età, sesso eccetera)
- contesto e preferenze generali relative alle funzioni da svolgere
- gli obiettivi dell'utente
- resoconti storici delle interazioni tra utente e servizio

Un profilo utente può dunque contenere dati sensibili, la cui raccolta deve avvenire rispettando gli aspetti legali relativi alla privacy individuale e che devono essere protetti con sistemi di sicurezza informatica adeguati.

### *Pattern: Misuratore di completamento*

L'utente desidera portare a termine un obiettivo, ma ha bisogno di una guida che gli mostri il percorso da completare e la percentuale di completamento [27]. L'utilizzo più diffuso nella comunità virtuali è legato al completamento del proprio profilo personale.

*Linkedin.com*

*Facebook.com*

*Vmax.dk*

## Quale problema risolve?

Suddivide l'obiettivo finale in diverse sotto-attività. L'obiettivo finale può essere arbitrariamente definito, come "La completezza del profilo" o "Il raggiungimento di uno status". Non appena si completa una sotto attività, la percentuale globale sale - raggiungendo il 100% quando l'obiettivo è raggiunto. Nel corso del processo (per esempio: "34% completato"), sono proposti uno o più collegamenti che guidano l'utente nei passaggi successivi. Ciò contribuisce a mantenere l'utente motivato nel passare al compito successivo.

Ci sono diversi approcci su come informare e celebrare che tutte le sotto attività sono state completate e che l'obiettivo finale è stato raggiunto. Quello più semplice è indicarlo in modo testuale (esempio "Il tuo profilo è completo!") magari accompagnato da un indicatore "100%". Un altro è quello di assegnare un trofeo da aggiungere alla collezione con cui può decorare il

proprio profilo e mostrarlo agli amici. Un terzo modo è quello di pubblicarlo sul suo profilo come notifica e magari inserirlo nel feed di notifiche centralizzato di tutto il sito.

### Quando usare questo pattern?

- Utilizzare quando si vuole guidare l'utente al completamento di un obiettivo specifico.
- Utilizzare quando si vuole garantire che gli utenti forniscano un insieme minimo ma completo di informazioni che riguardano il loro profilo e la loro presenza nella comunità.
- Non utilizzare quando per raggiungere la meta finale bisogna compiere una serie di passaggi rigidamente sequenziali.
- Non utilizzare per scopi critici, ma piuttosto per obiettivi interessanti ma facoltativi. L'idea di questo modello è quello di invitare l'utente svolgere più compiti di quanto avrebbe normalmente fatto.

### Motivazioni

Questo modello utilizza alcune tecniche psicologiche per spingere l'utente a proseguire verso l'obiettivo finale.

Uno è la *curiosità*. Siamo curiosi di scoprire cosa succede quando si raggiunge il 100%. Sarò premiato oppure il mio profilo avrà un aspetto diverso?

Un altro è l'effetto retroazione. Non appena un utente completa una sotto attività, il suo progresso si muove verso il 100%, stabilendo così un chiaro legame tra i compiti completati e il raggiungimento della meta finale.

#### *Domande aperte*

In quanto esseri umani, ci sentiamo inclini a completare obiettivi che ci abbiamo deciso. Il più delle volte scegliamo per noi stessi quali sono gli obiettivi che vogliamo e quanto tempo vogliamo dedicarvi. Il misuratore di completamento è un tentativo di presentare un tale obiettivo per l'utente in modo che lui a decidere completarlo. Presentando le sotto attività in maniera semplice e chiara, è possibile persuadere l'utente a passare del tempo che in altre circostanze non avrebbe.

## Reputazione

La tua reputazione in una comunità equivale alla somma delle tue azioni passate (belle o brutte) fatte all'interno della comunità. Il buon e il cattivo è soggettivo e dipende dalla comunità stessa. Quello che la comunità vede come valore è buono, quello che non sopporta è cattivo.

Una persona che partecipa ad una struttura sociale desidera sviluppare una propria reputazione e valuta la reputazione altrui, ma ciascun modello di partecipazione incarna il proprio insieme di pregiudizi e di strutture di incentivazione. Il bilanciamento di queste forze determina in larga misura il successo o il fallimento di un sistema sociale.

### *Progettare un sistema di reputazioni efficace*

Le dieci domande a cui rispondere per progettare di un sistema di reputazione efficace [30].

1. **Quali sono gli obiettivi di business della comunità?**
   Alcuni esempi possono essere: coinvolgere gli utenti, promuovere un prodotto o una funzionalità, premiare che contribuisce in modo più significativo, migliorare la qualità dei contenuti, aumentare la partecipazione degli utenti
2. **Quale spirito di comunità vuoi incoraggiare?**
   Di supporto, Collaborativa, Cordiale, Competitiva, Combattiva
3. **Quale sono le motivazioni che spingono gli utenti?**
   Meglio che non siano gli incentivi e basta, in questo modo si favorisce l'emergere dei membri peggiori. Non basatevi sull'altruismo soltanto. Le motivazioni tendono a variare notevolmente da un contesto all'latro quindi è necessario fare alcune ricerche.
4. **Quali entità permettono di migliorare la propria reputazione?**
   Persone? Cose? Entrambe? Una persona appartenete ad una comunità molto spesso non può essere distinta dal contributo tangibile che ha portato alla comunità stessa in fatto di contenuti o altro. Si valuta la persona valutando quello che ha fatto.
5. **A quali input occorre prestare attenzione?**
   Verifica quali contenuti possiedi e quali azioni gli utenti possono compiere. Dai a queste azioni un valore puntando alla qualità e tenendo conto della difficoltà.
6. **Quanto sono trasparenti le regole che portano ad accrescere la reputazione?**
   Alcuni siti sono completamente trasparenti, altri forniscono solo linee guida generali. In entrambi i casi il modo di generare punti influenzerà molto il comportamento della comunità.
7. **La reputazione decade se l'attività rallenta o si ferma?**
   Sicuramente si! Se l'attività di un utente decade anche la sua reputazione decade di conseguenza ma non dovrebbe mai tornare ad un valore minimo per non incoraggiare gli utenti a ricominciare da zero.
8. **Sono stati considerati gli aspetti culturali?**
   Considerate la valutazione di un esperto nel campo (es: per una comunità di auto aiuto in campo medico valutate con attenzione la attribuzione del ruolo di "Esperto").
9. **In quali ambiti la reputazione viene accresciuta?**
   La reputazione è sempre legata ad un contesto, e le persone sono varie. È possibile eccellere in un campo, non riuscire in un altro ed essere generalmente insignificanti in maggior parte di tutto il resto .
10. **Quale è il modello migliore nelle varie situazioni?**
    Vedere tabella nella pagina seguente.

### *Pattern: Valutazione del grado di competitività*

Il grado di competitività di una Comunità dipende i singoli obiettivi di membri della Comunità, dalle azioni che essi vi esercitano e a quale grado i confronti interpersonali sono desiderabili. Definendo la competitività della Comunità si può aiutare il progettista di un sistema di reputazione a determinare quali specifici modelli di reputazione impiegare [35].

#### Quali problemi risolve?

Quando una Comunità nuova o esistente richiede un sistema di reputazione, il progettista deve prestare un'attenta valutazione del grado di competitività della Comunità. Introdurre arbitrariamente incentivi competitivi in contesti non competitivi può creare problemi e può causare un scisma all'interno della Comunità.

#### Quando utilizzare questo pattern?

Questo modello va usato quando si sceglie il tipo di sistema di reputazione per la progettazione di una Comunità.

#### Quali problemi risolve?

La tabella qui proposta tenta di descrivere una Comunità in base al suo grado di "*competitività*" un termine qui usato per descrivere una combinazione di cose: gli obiettivi individuali dei membri della Comunità e in che misura tali obiettivi coesistono pacificamente, o in conflitto; le azioni cui partecipano membri della Comunità e in che misura tali azioni possono interferire con le esperienze di altri membri della Comunità; e in che misura sono voluti i confronti interpersonali o concorsi. Questo 'grado di competitività' è certamente soggettivo ed è sicuramente possibile trovare altre interpretazioni a questo modello. Il tentativo è quello di avere una traccia da cui partire piuttosto un modello definitivo e completo.

A seconda del relativo livello di competitività presenti, sono formulate raccomandazioni per modelli di reputazione appropriati.

#### Perché usare questo pattern?

Per avere un esempio di cosa può succedere a non usare questo pattern possiamo leggere il (dal titolo meraviglioso) "*I Love My Chicken Wire Mommy*" [35] dove Ben Brown parla dello "Sfortunato sistema a punti di Consumating.com" e gli effetti negativi causati allo spirito della Comunità quando un sistema a punti procura danno anziché beneficio.

| | Grado di competitività | Utilizzare la reputazione per… | Quali pattern usare? | Esempi |
|---|---|---|---|---|
| | **Di supporto**<br>I suoi membri sono motivati nell'aiutare altri membri dando consigli, consolazione o conforto. | Identificare i membri della comunità senior di buon livello, in modo che altri possono trovare di consulenza e orientamento. | Volontari che indossano una **etichetta di identificazione**: 'Referente' o 'Capo squadra'. Nuovi membri possono fidarsi di queste persone per muovere i loro primi passi nella Comunità. | • Gruppi di supporto<br>• Forum di esperti di salute |
| | **Collaborativa**<br>Gli obbiettivi dei membri sono ampiamente *condivisi*. I membri lavorano insieme per ottenere un obiettivo comune. | Identificare i membri della comunità che possano essere considerati partner affidabili. | Utilizzare **livelli denominati** per comunicare la storia dei membri: membri con gradi più alti dovrebbero essere ritenuti attendibili più facilmente dai neofiti. | • Wikipedia<br>• Yelp<br>• Siti di incontri |
| | **Cordiale**<br>I membri hanno ognuno una propria motivazione che non entra in conflitto con le motivazioni degli altri. | Mostra la storia della partecipazione di un membro in modo che altri possono avere un senso generale dei loro interessi, identità e valori. | Considerate le statistiche per evidenziare i contributi dei membri: basta mostrare i fatti e lasciare alla Comunità il compito di decidere il loro valore. Facoltativamente, i **gruppi dei migliori "X"** possono evidenziare membri con più contributi valutati. | • Bacheche e forum<br>• Youtube<br>• Yahoo! Answers<br>• Slashdot<br>• Ebay |
| | **Competitiva**<br>I membri condividono lo stesso obiettivo ma competono tra loro per raggiungerlo. | Mostra il livello di realizzazione di un membro, in modo che altri possano riconoscere (e ammirare) il loro livello di prestazione. | Consentire un facile confronto tra membri con i **livelli numerati**. Fornire le mini-motivazioni tramite la **collezione di benemerenze**. | • Fantasport<br>• Giochi di strategia |
| | **Combattiva**<br>I membri hanno obiettivi opposti: quando uno dei membri raggiunge il proprio ha *negato* la possibilità di raggiungerlo agli altri. | Mostra la storia delle realizzazione di un membro, comprese le vittorie e le sconfitte. Utilizzare la reputazione di stabilire diritti di merito. | Visualizzate l'avanzamento di ogni membro assegnando punti per diverse azioni (**Conteggio punti**). Confrontate i punteggio tra i membri, mostrando i vincitori e vinti (**Classifiche**). | • Xbox live<br>• Hot o Not |

Aumento della competitività

Aumento della comparazione tra persone

## *Pattern: Collezione di benemerenze*

Collezionare benemerenze online può sembrare sciocco o banale, ma il collezionismo genera dipendenza e può obbligare gli utenti a esplorare le parti della vostra offerta che magari non verrebbero considerati [27] [28].

### Quale problema risolve?

Alcuni partecipanti nelle comunità rispondono alle opportunità di guadagnare o vincere premi che possono essere raccolti e visualizzati dagli altri membri della comunità.

### Quando usare questo pattern

- Quando si vuole far leva sulla natura compulsiva degli utenti. Può sembrare sciocco o banale, ma il collezionare traguardi può generare dipendenza, quando fatto bene, e obbligare gli utenti ad esplorare parti della vostra offerta che altrimenti non interesserebbero loro.
- Quando si vuole incoraggiare la comunità a sperimentare tutti gli aspetti della vostra offerta. Se ci sono servizi specifici aspetti o del prodotto offerto che si desidera promuovere: per esempio, se si desidera incoraggiare più operazioni in un contesto di fantasport, si può considerare gli utenti premiando il decimo scambio avvenuto con successo. ('Successo' è la parola chiave: bisogna introdurre e imporre qualche nozione di qualità nella realizzazione.)

### Quale soluzione offre questo pattern

Fornite qualche vantaggio, o ricompensa per gli utenti al raggiungimento di determinati obiettivi all'interno della comunità. Fate in modo che esista una collezione, o un programma di collezionismo.

- Fate leva sulla voglia di divertimento sviluppando feticci desiderabili:
- Sviluppate trofei originali, attraenti, icone ben disegnate da mostrare in ogni occasione
- Date la possibilità agli utenti di mostrarle nel proprio profilo
- Provvedete a creare il giusto mix di difficoltà:
- Alcuni premi devono essere facili da raggiungere, altri devono richiedere tempo e fatica per essere conquistati
- "Sbloccate" i premi più ambiziosi quando sono stati raggiunti quelli più semplici.

### *Raccomandazioni*

- Le benemerenze, come la maggior parte dei meccanismi di reputazione, dovrebbero incoraggiare la partecipazione di qualità e non una semplice ripetizione di attività. Quindi, non ricompensate un utente per aver "giocato 20 partite". Premiate piuttosto le "20 vittorie in una stagione."
- Potrebbe essere utile sviluppare una serie di badge per 'la prima volta che..' (ad esempio, 'la prima recensione', 'la prima ricetta pubblicata', 'il primo commento ricevuto'.)
    - Rendete queste conquiste di valore inferiore ad altre ben più difficili da ottenere.

  - Non continuare a premiare più volte lo stesso comportamento. Questi 'primi' premi sono utili per incoraggiare le persone a provare le nuove funzioni ma non premiateli per più di una volta.
- Elencate tutti i possibili traguardi da raggiungere, in modo che gli utenti sappiano quali sono a loro disposizione.
  - Indicate quelli che sono stati già raggiunti.
  - Mostrare alcuni premi "bloccati" fino a quando non sono stati conseguiti.
- La collezione di benemerenze non deve essere confusa con punti, anche se si potrebbero usare congiuntamente (per esempio, affidare un certo numero di punti guadagnati per ogni traguardo raggiunto).

### *Esclusività*

Sentitevi liberi di essere abbastanza generosi con risultati da collezione. Ogni membro della vostra comunità dovrebbe avere facile accesso ad alcune conquiste. Ma mantenete alcuni risultati rari e difficili da ottenere.

Rendete sempre tali elementi cliccabili. Essi dovrebbero fornire una spiegazione di quello che significano.

### Esempi

Sul profilo di un utente **Amazon.com** sono visibili i risultati ottenuti.

Nella figura vediamo il "Recensore numero 1", che si presenta con il suo vero nome "Real Name", verificato dalla carta di credito, ed è membro del programma "Vine™ Voice" di Amazon (un programma su invito che dà diritto ad alcuni utenti selezionati di visionare in anteprima materiale su cui scrivere recensioni.

I questo esempio vediamo come alcuni badge sono stati conseguiti per azioni (numero di recensioni scritte), mentre altri per avere deciso di presentarsi in un certo modo (usando il proprio nome, cercando di guadagnare credibilità e rispetto per questo).

**Yahoo! Fantasy Sports** dà la possibilità di collezionare trofei e premi per i diversi sport. Essi sono di natura temporale, e vengono assegnati per una stagione di gioco. Uno "Scaffale dei trofei" (Trophy Case)" rimane visualizzato sul profilo dei giocatori.

## *Pattern: Etichette identificative*

Una famiglia (una o più) di reputazioni, che non sono di natura sequenziale. Ogni reputazione è creata per identificare e premiare comportamenti particolari o qualità all'interno di una comunità. [40] [41]

### Quali problemi risolve?

I membri di una Comunità hanno necessità di identificare altri membri della comunità che si sono in qualche modo distinti per azioni specifiche, abilità dimostrate o perché si sono offerti volontari nell'aiuto ad altri membri della comunità.

### Quando usare questo pattern

- Si sono identificati alcuni comportamenti desiderabili per la Comunità che si desidera promuovere.
- Si desidera consentire agli utenti di offrirsi volontari per un "ruolo" o responsabilità all'interno della Comunità.
- C'è necessità di convalidare una reputazione e considerarla ufficialmente attendibile attraverso un sistema imparziale.

### Quale soluzione offre questo pattern

Definire una famiglia (uno o più) di reputazioni che non sono sequenziali. Ogni reputazione è predisposta per identificare e premiare particolari comportamenti o qualità all'interno della Comunità.

Le etichette di identificazione sono utili per i consumatori per individuare i contribuenti più con esperienza che possiedono specifiche qualità. (Ad esempio, guide 'utili', o recensori di 'fama'), mentre non sono particolarmente utili per confrontare uno stato con un altro.

### *Raccomandazioni*

- E' possibile conseguire più di un titolo identificativo contemporaneamente.
- Si po' richiedere all'utente se accetta o meno che l'identificazione della sua reputazione appaia sul suo profilo.

### *Esempi*

Yelp descrive la sua squadra di Elite come "il nostro modo di riconoscere alcuni dei nostri membri più attivi e influente, dentro e fuori il sito".

Si noti che lo status Elite è limitato nel tempo ed ha durata annuale.

http://www.yelp.com/elite

### *Pattern: Categorizzazione di merito a livelli*

#### Livelli numerati

Una famiglia di reputazioni in un continuum progressivo. Ogni livello che si ottiene è superiore a quello precedente. I livelli sono indicati con il loro numero, il che rende il confronto tra i livelli molto semplice e facile da fare.

#### Livelli nominati

Una famiglia di reputazioni in un continuum progressivo. Ogni livello che si ottiene è superiore a quello precedente. I livelli sono indicati con un nome univoco, che può indicare la qualità in un modo divertente e immediato. Fare un confronto rapido tra i vari livelli, tuttavia, diventa un po' più difficile.

### *Pattern: Conteggio punti*

Un conteggio cumulativo del numero di punti che un utente si è guadagnato all'interno di una comunità. I punti in genere provengono dall'aver eseguito una serie di attività all'interno del gruppo.

### *Pattern: Il gruppo dei migliori X*

I collaboratori sono raggruppati, numericamente, in gruppi e i migliori contribuenti sono riconosciuti per i loro risultati eccellenti. I primi 10, 50 e 100 sono alcuni raggruppamenti di uso comune.

### *Pattern: Classifiche*

La prestazione degli utenti è traccia in modo empirico e granulare (più tipicamente punti), al fine di confrontarli uno contro l'altro. Gli utenti accumulano reputazione crescente l'uno contro l'altro, il che avviene necessariamente a spese degli altri utenti della comunità. Gli utenti vengono visualizzati in classifiche.

### *Pattern: Premi temporali*

Gli utenti sono ricompensati o riconosciuti per i loro contributi per un periodo o intervallo di tempo specifico. I riconoscimenti settimanale, mensile o annuale sono gli esempi più comuni. Queste reputazioni, una volta guadagnate, non sono mai perse: sono utili sia per riconoscimento agli utenti più meritevoli sia per dare un più ampio numero di utenti l'opportunità di guadagnare una reputazione. I premi temporali possono anche essere utilizzati ricordare la reputazione legata ad eventi speciali o promozioni.

### *Pattern: Statistiche*

Le statistiche sulla partecipazione di un utente o la storia della comunità vengono visualizzati. Nessun tentativo è fatto di aggregare o di astratto le statistiche in schemi di ordine superiore: piuttosto, i lettori possono decidere da soli il valore dei contributi di quell'utente. Dati statistici possono anche essere utilizzati per convalidare un altro modello di reputazione (ad esempio, di fornire una giustificazione del motivo per cui qualcuno ha guadagnato una medaglia d'oro).

### *Pattern: Referenze tra pari*

I membri della comunità sono forniti di un meccanismo per parlare delle qualità di altri membri. In genere, testimonianze positive sono incoraggiate (spesso forzate, consentendo al destinatario il diritto di approvare le testimonianze presentate prima che siano rese pubbliche).

Spesso, le testimonianze dei colleghi sono dirette a favorire la reciprocità.

### *Pattern: Premi tra utenti*

E' simile alla collezione di benemerenze. La differenza notevole è che questi sono concessi tra pari (e non sono ufficializzate attraverso i canali principali.) Pertanto varranno meno e, probabilmente, non rifletteranno reali qualità oggettive.

# Contenuto

Definiamo contenuto tutto ciò che ha un valore per la comunità, sono i contenuti il motivo per cui gli utenti navigano il sito, si incontrano e si scambiano opinioni. Esempi di contenuti possono essere una foto, un testo, un link, un video, o qualunque altra espressione particolare che possa essere pubblicata nella comunità.

Un contenuto possiede alcuni elementi di base che sono generalmente comuni a tutti:

- **Titolo**, o breve descrizione testuale del contenuto, utile soprattutto per l'indicizzazione
- **Proprietario**: è il membro che ha pubblicato il contenuto, è fondamentale sia per attribuire il giusto riconoscimento a quanto pubblicato sia per identificare eventuali responsabilità nel caso in cui il contenuto violi le linee guida della comunità o, peggio, le leggi vigenti
- **Data di pubblicazione**: importante per fornire una collocazione temporale e per distinguere tra contenuti nuovi e contenuti datati.

Oltre a questi abbiamo valori opzionali ma raccomandati se disponibili:

- **Riassunto**, breve testo usato in elenchi e nell'esposizione del contenuto in RSS.
- **Descrizione**, intesa come testo descrittivo, che può essere anche piuttosto lungo.
- **Contenuto multimediale:** potrebbe essere video, audio o immagine.
- **Localizzazione topografica**. È particolarmente importante per fornire una rappresentazione cartografica e spaziale del contenuto disponibile. Molti contenuti possono avere questa classificazione (una foto può avere le coordinate del luogo dove è stata scattata e un video dove è stato girato, un testo può essere collocato dove si trovava l'autore quando è stato scritto o dove si trova l'oggetto di cui tratta, e così via…). La localizzazione può essere composta da una descrizione testuale (indirizzo postale) e dalle coordinate di latitudine e longitudine. In aggiunta alle coordinate è possibile indicare anche l'altezza. Esistono servizi che possono ricavare le coordinate a partire da un indirizzo postale o da una località.
- **Localizzazione temporale**. Aggiungendo una data ed un'ora ad un contenuto lo si contestualizza nel tempo. Possiamo dunque rappresentare la data e l'ora di uno scatto fotografico o la ripresa di un video. Ma anche la data di un evento o una manifestazione di cui disponiamo soltanto di una descrizione testuale.
- **Licenza**: ad ogni contenuto possono essere agganciate informazioni relative alle condizioni d'uso (esempio: wikipedia, filckr… tipi di licenza: pubblico dominio, riutilizzabile indicando l'autore, non riutilizzabile)

## Classificazione

La maggior parte delle comunità virtuali si evolvono intorno a una particolare categoria di oggetti, ad esempio segnalibri, video, news.

Gli elementi possono essere aggregati tra loro sfruttando le parti che hanno in comune. Lo scopo è duplice. Prima di tutto essa fornisce un accesso strutturato a elementi di contenuto: ad esempio la navigazione dei contenuti condivisi e l'individuazione di elementi interessanti possono essere favoriti dalla presentazione di una gerarchia di gruppo. In secondo luogo, le aggregazioni di contenuto possono essere utilizzate dai membri della comunità, come tutti i metadati, come ulteriori informazioni per capire le proprietà di un oggetto determinato [10].

## Classificazione fatta dal proprietario

Credo sia importante distinguere l'autore della classificazione per cui la mia proposta è di distinguere le possibili classificazioni innanzitutto sulla base di chi è autorizzato a classificare il contenuto. L'autore e proprietario del contenuto è il primo (e normalmente il solo) autorizzato a catalogare il proprio contenuto per categoria (sia essa tassonomia che folksonomia).

Di questa abbiamo due classificazioni:

### *Tassonomie o categorie*

La tassonomia è, nel suo significato più generale, la scienza della classificazione. Con il termine tassonomia ci si può riferire sia alla classificazione gerarchica di concetti, sia al principio stesso della classificazione. Praticamente tutti i concetti, gli oggetti animati e non, i luoghi e gli eventi possono essere classificati seguendo uno schema tassonomico. Secondo la matematica, una tassonomia è una struttura ad albero di istanze (o categorie) appartenenti ad un dato gruppo di concetti. A capo della struttura c'è un'istanza singola, il nodo radice, le cui proprietà si applicano a tutte le altre istanze della gerarchia (sotto-categorie). I nodi sottostanti a questa radice costituiscono categorie più specifiche le cui proprietà caratterizzano il sotto-gruppo del totale degli oggetti classificati nell'intera tassonomia [29].

La tassonomia è dunque un metodo di classificazione in cui la lista delle categorie tra cui scegliere è prefissata e decisa a priori dai progettisti o dagli amministratori della comunità.

Il vantaggio di questa classificazione è che è facile trovare aggregazioni di contenuti omogenei.

D'altra parte, però, la rigidità dell'elenco rende a volta difficile trovare una collocazione specifica di un contenuto che magari appartiene solo parzialmente a più categorie oppure risulta inclassificabile col sistema adottato.

È consigliabile comunque avere un solo livello di categorie per semplificare la catalogazione e la ricerca anche se in linea teorica è possibile avere un albero abbastanza articolato.

### *Folksonomie o tag*

Mentre le tradizionali applicazioni Web in genere forniscono agli utenti solo una classificazione standard gerarchica per gli elementi di contenuto, le applicazioni sociali preferiscono attuare un approccio collaborativo anche alla definizione delle etichette di contenuto (*tag*).

Ai membri della comunità è quindi lasciata la libertà assoluta di associare ai contenuti diverse parole chiave che siano ritenute opportune a descrivere quanto pubblicato, delegando poi ad algoritmi di aggregazione (*clustering*)[33] il compito di reperire e mettere in relazione tra loro contenuti simili. Il vantaggio evidente è la libertà di catalogazione e la precisione con cui si può considerare il contenuto, anche recente o inclassificabile.

Lo svantaggio è la polverizzazione delle possibili classificazioni che diventa più evidente tanto più il numero di contenuti è basso, diventa sempre più indicato al crescere del numero di contenuti perché in proporzione aumenterà il numero di contenuti che condividono parole chiave ed è indispensabile quindi attivare algoritmi di ricerca e aggregazione sempre più sofisticati.

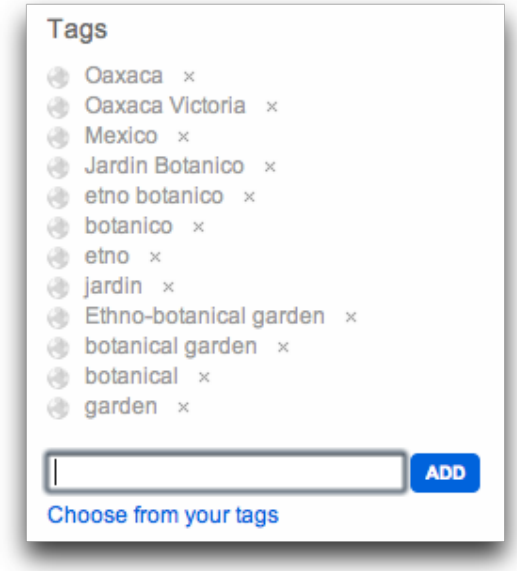
Per limitare questo approccio alcuni siti hanno cominciato a limitare la creazione di nuovi tag, riservandola ad utenti più esperti o che hanno all'attivo un certo numero di contenuti già inseriti. E' il caso di stackoverflow.com, che permette di creare nuovi tags solo ad utenti che hanno inserito almeno 30 domande, per gli altri è necessario scegliere uno o più tags già presenti nel sistema. In questo modo si evita la polverizzazione in tags con grafia magari simile, ma non identica (es: ASPNET, ASP.NET, ASP-NET).

### *Collegamento tra folksonomie e tassonomie*

Alcuni membri possono essere pigri nel classificare quanto pubblicano e questo potrebbe portare ad avere contenuti che risultano invisibili ai più perché non ricercabili con i sistemi di classificazione. Per ovviare a questo problema si può attivare un sistema di relazione tra tag e categorie. In una prima fase viene analizzato il testo del titolo e di eventuali descrizioni, estraendo le parole con più di tre lettere. Queste parole verranno poi ricercate nell'elenco dei tags utilizzati e, se presenti, aggiunti come tag del contenuto. Per ogni tag inserito si verifica se esiste una corrispondenza con una categoria esistente e, se disponibile, si seleziona automaticamente la categoria più rilevante che corrisponde ai criteri.

### *Pattern: Assegna il tag ad un oggetto*

Consente agli utenti di aggiungere i propri tag (parole chiave) a un oggetto. Essere chiari su come separare (delimitare) tag distinti. Si consideri la possibilità di consentire i aggiungere tag anche ad oggetti che non si sono creati e non si posseggo. Non si deve aver paura di mescolare un vocabolario controllato con tag generati dagli utenti [30].




*Tag an Object usato da Flickr, immagine da [30]*

## Quale problema risolve?

Un utente desidera collegare le sue parole chiave o un insieme di parole chiave ad un oggetto per l'organizzazione e il successivo recupero.

## Quando usare questo pattern?

- Utilizzare questo modello, quando un utente sta raccogliendo una grande quantità di dati non strutturati, ad esempio foto.
- Utilizzare questo modello, quando un utente vuole gestire una vasta collezione di oggetti, ad esempio libri.
- Utilizzare questo modello per fondere etichette parole chiave inserite direttamente dagli utenti con metadati strutturati.

## Quale è la soluzione?

- Consentire agli utenti di aggiungere i propri tag di un oggetto.
- Consentire agli utenti di cancellare i tag associati a un oggetto. Ciò consente l'eliminazione di duplicati o di correggere errori ortografici.
- Fornire indicazioni molto chiare su come separare i tag distinti. Ci sono due metodi attualmente conosciuti in tutto il web in questo momento: delimitato da virgole e delimitato da spazio. Possono essere usati entrambi ma occorre coerenza e chiarezza nello specificare quale formato ci si aspetta: non c'è niente di più frustrante nel pensare di delimitare con la virgola dei tag in un sistema che li separa con spazi e rendersi conto che il'intero significato del proprio tag è stato modificato.
- Per consentire un impegno sociale maggiore è opportuno consentire alle connessioni e/o agli amici di modificare i tag degli oggetti in una collezione.

- Non abbiate paura di mescolare un vocabolario controllato - definito dagli architetti sito - con tag inseriti dagli utenti.

### *Raccomandazioni*

- L'aggiunta di un meccanismo di catalogazione di oggetti come una caratteristica del prodotto dovrebbe offrire un vantaggio per l'utente. I tag aiutano un utente a trovare e gestire la sua collezione? Il modello è tanto più efficace quanto più vi è un vantaggio per l'utente e le sue connessioni.
- È preferibile fornire dei suggerimenti agli utenti se i tag vengono aggiunti ad un elemento pubblico (un articolo di giornale, un segnalibro). Il sito Delicious mostra alcuni tag associati in precedenza con l'oggetto in modo che l'utente possa scegliere tra questi ed evitare errori o ripetizioni.
- I tag e la possibilità di aggiungerli devono essere in stretta vicinanza l'oggetto a cui si riferiscono.

### *Domande aperte*

L'uso di una virgola o dello spazio come delimitatore è ancora una questione in sospeso e non si è ancora imposto uno standard. È improbabile che uno prevalga sugli altri in quanto entrambi sono pervasive ed hanno radici nei delimitatori tradizionalmente usati nei database, che consentono diverse opzioni.

### Perché usare questo pattern?

Consentire alle persone di inventare liberamente le loro etichette o parole-chiave può portare ad una folksonomia collaborativa curata che può aiutare gli altri a trovare ciò che stanno cercando.

## *Dove e quando?*

Una delle cose che ci piace fare come creature sociali è quello di pianificare eventi e riunirci in gruppi. Pranzi, incontri, feste, matrimoni, funerali. Gli eventi sono diventati più facili da pianificare con una varietà di strumenti online, e con l'esplosione di strumenti sociali, possono essere più collaborativi che mai.

Funzionalità GPS sono disponibili ormai nella maggior parte dei nostri dispositivi e applicazioni, e la proliferazione di dispositivi mobili significa che possiamo essere connessi indipendentemente da dove siamo. Strumenti di posizione forniscono un contesto di luogo attorno a ciò che stiamo facendo on-line. L'incrocio tra gli interessi comuni maturato online ed il desiderio di vedersi faccia a faccia è una miscela potente.

La via per realizzare tutto questo è l'integrazione di diverse tecnologie ed informazioni in modo da rendere il reperimento ed il collegamento delle informazioni il più semplice possibile.

I seguenti modelli, in combinazione con i modelli di identità, presenza e attività come forum, collezioni e gruppi, creano una ricca suite di strumenti che possono essere combinati per aiutare a unire le persone, sia quelli che già si conosco che persone ancora sconosciute.

### *Pattern: Geo-Tagging e Geo-Mashing*

#### Quale problema risolve?

Un utente desidera annotare una persona, luogo o cosa con un tag geografico, di solito in forma di latitudine e longitudine, che poi viene tradotto in un indirizzo che possa essere posizionato su una mappa [4] pp. 422. I Geo mashups sono applicazioni e strumenti che combinano una mappa fornita da un gestore di mappe come Yahoo!, Google e MapQuest e contenuto geo-taggato, ad esempio foto, utente di blog, articoli di notizie, dati immobiliari, video e altri dati che possono avere tags geografici associati ad esso [4] pp. 428.

#### Quando usare questo pattern?

Utilizzare questo modello quando si desidera posizionare oggetti (persone, luoghi o cose) su una mappa.

#### Quale è la soluzione?

- Se l'elemento è una foto e ci sono dati exif che includono informazioni di posizione (latitudine e longitude), l'oggetto può essere inserito automaticamente su una mappa.
- Se l'elemento ha un indirizzo associato ad esso, può essere inserito automaticamente su una mappa.
- Consentire all'utente di associare un indirizzo a un oggetto.
- Si consideri di poter consente agli utenti di trascinare e rilasciare gli elementi (foto, inserzioni, amici) su una mappa memorizzare l'associazione così creata.
- Consentire all'utente di perfezionare la posizione offrendo un modulo per inserire un indirizzo postale completo (opzionale).
- Consentire agli utenti di immettere specifici valori di latitudine e longitudine per indicare un punto su una mappa in alternativa al posizionamento interattivo.

#### Perché usare questo pattern?

Offrendo funzionalità Geo-tagging per le risorse, in particolare per le foto, permette loro di esistere non solo nel tempo, ma anche in un contesto relativo al mondo reale. L'immissione di immagini su una mappa dà alle persone un senso che questa immagine esiste davvero e che si potrebbe andare a vedere di persona. Geo-tagging mostra a tutti che qualcuno è stato lì, proprio in quel luogo.

### Esempi

Aggiunta di una foto a una mappa su Flickr attraverso la selezione su una mappa. Le coordinate Lat/long sono presentate in basso a destra dell'interfaccia, mentre una traduzione di facile comprensione reale luogo della posizione è presentata nella parte superiore destra dell'interfaccia.

Panoramio è un sito dove è possibile condividere le proprie foto preferite in ogni luogo del mondo. A differenza di altri siti di condivisione di foto è integrato con Google maps in modo da poter individuare da dove provengono le foto.

Nella immagine a sinistra una veduta del porto di San Remo (IM)

http://www.panoramio.com/

## *Pattern: Calendario*

### Quale problema risolve?

Un utente vuole trovare o presentare un evento (pubblico o privato), basato su una data o in un determinato intervallo di date [4] pp. 416.

### Quando usare questo pattern?

- Utilizzare questo modello per creare eventi basati su Data.
- Utilizzare questo modello per trovare gli eventi per data.
- Utilizzare questo modello in combinazione con gli eventi “Faccia a faccia” e le “Feste”.

### Quale è la soluzione?

- Consentire agli utenti di associare un evento a una data. Questo può essere fatto attraverso un'interfaccia di pianificazione degli eventi o all'interno di un'interfaccia calendario .
- Consentire il creatore di evento indicare se l'evento è pubblico o privato .

- Consentire di condividere l'evento del calendario da, sia attraverso le selezioni dirette dalla rete dell'utente, tramite e-mail o tramite RSS, blog o altre reti sociali.
- **Selezione data**: La selezione della data dovrebbe avvenire usando un selettore data con calendario, ciò consente all'utente di visualizzare la data nel contesto di altre date, il giorno della settimana e garantisce meno errori di immissione dei dati.
- **Dettagli calendario**: Fornire un campo titolo e descrizione. Fornire un campo URL e un campo Note per altre informazioni. Consentire all'utente di associare una località per l'evento. Quando i dettagli sono completi, presentare l'evento in tutte le presentazioni del calendario (cioè visualizzazione elenco, giorno, settimana e mese). Visualizzare il titolo, la posizione e la maggior parte di una descrizione quando appropriato, in base allo spazio disponibile nella visualizzazione corrente. Utilizzare i calendari a scomparsa che si attivano al passare del mouse.

## Perché usare questo pattern?

Gli eventi sono vincolato nel tempo, l'utilizzo di solidi strumenti on-line di calendario rende la creazione di feste, eventi e riunioni più facile da programmare.

## Esempi

Con Google Calendar è possible aggiungere eventi e inviare inviti con la massima facilità, condividere appuntamenti con amici e familiari e cercare eventi interessanti. Fornisce una visualizzazione giornaliera, settimanale o mensile. La scheda Agenda visualizza gli eventi sotto forma di elenco personalizzabile.

http://www.google.com/intl/it/googlecalendar/tour.html

FullCalendar è un plugin per jQuery che fornisce un calendario a dimensione piena con drag&drop.

Utilizza AJAX per recuperare gli eventi al volo per ogni mese e facilmente. È personalizzabile nella sua rappresentazione ed espone eventi per ulteriore personalizzazione (come clic o trascina).

http://arshaw.com/fullcalendar/

## Classificazione fatta dalla comunità e feedback

Una volta che il contenuto è stato pubblicato i membri della comunità possono interagire con esso fornendo commenti e valutazioni, dimostrando il proprio gradimento o il proprio disappunto.

In particolare possiamo evidenziare alcuni modi d'interazione standard:

### *Pattern: Scrivi una recensione o invia commento*

Fornire collegamenti contestualmente pertinenti che consentono all'utente di avviare il processo di scrittura di una recensione o commento, etichettato con un chiaro invito all'azione, come ad esempio "Scrivi la tua recensione".

**Commenti**, sono brevi commenti testuali che i membri della comunità possono assegnare ad un contenuto. La particolarità dei commenti è che essi stessi sono dei mini contenuti che ne seguono le regole di base (hanno un proprietario, una data, un testo, possono essere segnalati e commentati a loro volta)

Se è il caso, è consigliabile consentire una forma più semplice di commento negativo o positivo.

Tenete presente di non confondere lo scrittore con troppi campi.

La maschera di inserimento dovrebbe agevolare la compilazione e non deve necessariamente presentare le informazioni nell'ordine in cui verranno pubblicate.

#### Quale problema risolve?

Un utente vuole condividere il suo parere con gli altri su un oggetto (luogo, persona, cosa) in maggiore dettaglio rispetto a una semplice valutazione.

#### Quando usare questo pattern?

- Un utente vuole scrivere una recensione di un oggetto.
- Si desidera integrare il contenuto di un prodotto/sito con i pareri e le recensioni degli utenti.
- Se si utilizza anche un oggetto di valutazione [46], la combinazione di una valutazione con una recensione contribuisce ad ottenere un feedback migliore.
- Si utilizza anche in combinazione con le classifiche reputazione (per incoraggiare la qualità delle recensioni)

#### Quale è la soluzione?

- Fornire collegamenti contestualmente pertinenti che consentono all'utente di avviare il processo di scrittura di una recensione.
- Il testo dovrebbe essere etichettato con un chiaro invito all'azione, come ad esempio "Scrivi la tua recensione"
- Il modulo da compilare include in genere i seguenti cinque elementi fondamentali:
    - Valutazione quantitativa (un voto, in genere da 1 a 5)

  - Un campo per inserire una valutazione qualitativa (recensione) dell'utente in merito all'oggetto
  - Alcune linee guida per aiutare l'utente a scrivere una recensione utile
  - Liberatorie legali e quelle legate alla privacy
  - Un campo che identifica l'utente che sta scrivendo la recensione, di solito obbligatori e pre-compilato con le generalità dell'utente collegato.
- Se lo si ritiene opportuno, si consideri di ammettere una semplice recensione positiva o negativa, evitare di inserire troppi campi per non confondere l'utente che scrive la recensione.
- Indicare chiaramente i campi che sono obbligatori.
- Organizzare i campi in modo funzionale all'inserimento della recensione, senza preoccuparsi di come verranno visualizzati quando pubblicati.
- Utilizzare delle restrizioni sulla lunghezza dei testi per incoraggiare la lunghezza appropriata (breve e concisa oppure narrativa).
- Dopo che l'utente ha completato la revisione, consentirgli di vedere il risultato del suo lavoro in anteprima, con la possibilità di vedere l'anteprima o annullare la pubblicazione.
- Fornite informazioni chiare e precise circa la pubblicazione della recensione, se è disponibile immediatamente o se invece deve passare da un processo di approvazione.
- Se possibile, fornire oggetti aggiuntivi di cui è possibile fornire un'altra recensione.
- Se l'utente non ha compilato i campi obbligatori, fornire appropriati messaggi di errore per dare la possibilità di correggersi.

### Perché usare questo pattern?

Brevi valutazioni positive o negative sembra che siano più semplici per gli utenti da creare che non una narrativa completa. Non è necessario pensare a frasi complesse ed hanno una direzione più specifica su cosa scrivere. Per i lettori è più semplice leggere velocemente i commenti che non una completa narrazione.

### Accessibilità

- Consentire all'utente di spostarsi tra i campi premendo il tasto TAB.
- Consentire all'utente di pubblicare la revisione premendo il tasto invio.

## *Pattern: Valutare un oggetto*

Identifica una serie di oggetti cliccabili (spesso stelle) che si illuminano al passaggio del mouse ed invitano l'utente a valutare il contenuto (ad esempio, "Vota!"). Lo stato iniziale deve essere vuoto. Mentre il cursore del mouse viene spostato sopra le icone, visualizzare una descrizione di testo della votazione per in ogni punto (ad esempio, "Eccellente"). Il voto finale deve essere indicato con un cambiamento di colore e un'indicazione del testo corrispondente alla valutazione [46] [48].

Le classifiche, identificano un apprezzamento qualitativo con un voto numerico. L'insieme di tutti i valori può essere un indice di qualità e di apprezzamento del contenuto da parte della comunità.

### Quale problema risolve?

Un utente vuole lasciare rapidamente il proprio parere su un contenuto, senza interrompere le attività che sta compiendo.

### Quando usare questo pattern

- Un utente vuole lasciare rapidamente un parere.
- Va utilizzato in combinazione con le recensioni per una esperienza più ricca.
- Si può utilizzare per inserirsi rapidamente all'interno della "comunità" degli utilizzatori di un particolare contenuto.
- Utilizzare per toccare rapidamente nella "Comunità" esistente di un prodotto.
- Le valutazioni sono raccolte per presentare una valutazione media di un oggetto da parte cella Comunità di utilizzatori.

### Quale è la soluzione?

- Mostra gli oggetti cliccabili (le stelle sono il tipo più usato) che si illuminano al passaggio del mouse per indicare che sono cliccabili.
- Stato iniziale dovrebbe essere "vuoto" e mostrare un testo che inviti l'utente ad esprimere il proprio parere (ad esempio. "Vota qui!").
- Quando il cursore del mouse si sposta sopra le icone, vai indicato il voto che si sta esprimendo (attraverso un cambiamento di colore) e una descrizione testuale adeguata (ad esempio, "Eccellente").
- Una volta che l'utente ha fatto clic la votazione (5 stelle, 3 stelle ecc.) dovrebbe essere salvata e aggiunta alla votazione media (visualizzata separatamente).
- La votazione salvata deve essere indicata con un cambiamento del colore finale degli elementi e un'indicazione testuale della votazione salvata.
- Dovrebbe essere visualizzata anche una votazione media o aggregata.
- Gli utenti dovrebbero essere in grado di cambiare il loro giudizio se cambiano idea.
- Recentemente sta prendendo piede un formato più semplice, iniziato da Facebook e recentemente scelto da YouTube al posto del tradizionale metodo con le stelline. La forma più semplice di apprezzamento per un contenuto si basa su un semplice "Mi piace" / "Non mi piace", basta un click. Può essere considerata una forma di classifica a voto singolo (+/-1). E' utile soprattutto nei contesti più semplici ed immediati, ad esempio per valutare i commenti.

### Perché usare questo pattern?

La valutazione di un oggetto fornisce un modello semplice per coinvolgere gli utenti nelle valutazioni dei contenuti

E' spesso usato in congiunzione con le recensioni per incoraggiare la partecipazione degli utenti nel fornire più ricchezza ai contenuti.

### Accessibilità

Utilizzare DHTML e CSS per la visualizzazione degli Stati al passaggio del mouse e per la registrazione istantanea della votazione. Nei casi in cui ciò non è possibile, può essere aggiunto un pulsante "Memorizza votazione" per confermare la selezione finale del giudizio.

### Esempi

Su **Amazon.com**, ogni valutazione è accompagnata da una recensione testuale che spiega la votazione assegnata.

La valutazione è data su una scala da 1-5, accanto alla revisione testuale è presente il nome dell'autore e la sua posizione.

Sul sito **UI-patterns**, gli utenti possono far sapere a tutti quanti, se ritengono che un modello rappresenti una buona pratica oppure no.

**Youtube** ha recentemente cambiato il metodo di valutazione strasofrmando le classiche 5 stelline in un più immediato "Mi piace", "Non mi piace", sul modello di Facebook. Notare il fumetto in nero che etichetta la votazione e il cambio di colore al passare del mouse e la visualizzazione delle statistiche.

**Segnalazione**: in un sito creato e gestito dagli utenti è possibile trovare di tutto. Le segnalazioni sono nate come uno strumento che gli utenti hanno per segnalare ai moderatori che il contenuto pubblicato viola alcune regole.

Si potrebbe estendere il concetto anche a segnalazioni di tipo positivo, che identifichino contenuti meritevoli di plauso

## Diffusione

È interesse di ogni membro che il contenuto che pubblica possa essere fruito dal massimo numero di persone possibile e quindi da più di una comunità. Anziché pubblicare il contenuto su diverse

comunità è più semplice inserirlo su una e poi fare in modo che le altre possano collegarsi al contenuto e farlo proprio, magari aggiungendo poi meccanismi di commento e classificazione propri.

La diffusione si basa su alcuni aspetti

### *Esportazione*

Capacità del sito di fornire contenuti in un formato facilmente fruibile da siti di terze parti.
Tipicamente i metodi sono due:

- codice html da inserire direttamente sulla pagina di destinazione (per i video e contenuto multimediale in genere), come tag object o javascript
- url (in formato rss o altro standard, tipicamente in formato XML)
- web services come fonte dati

### *Importazione*

Capacità del sito di riproporre sulle proprie pagine contenuti pubblicati in altre comunità.

### *Privacy*

Il livello di privacy indica chi può accedere al contenuto nelle intenzioni dell'autore. Impostazioni tipiche sono: privato (solo l'utente può accedere), solo amici (solo gli amici, concetto che può essere esteso a gruppi selezionati in varie declinazioni, etc.) oppure tutti quanti.

Le impostazioni di diffusione e condivisione possono esser molto complesse e granulari, permettendo all'utente di definire per ogni contenuto il livello di visibilità e di interazione possibile.

### *Condivisione*

Possibilità di segnalare il contenuto ad altri membri o a siti esterni

- **Segnala ad un amico**, funzione che permette di inviare un messaggio (tipicamente email) a degli amici per segnalare il contenuto
- **Collegamento con siti esterni**. Proporre i propri contenuti in maniera del tutto automatica (con un semplice click) ad altri siti di comunità. Presuppone la conoscenza dei siti a cui proporre il contenuto (Contribution pattern)

## *Pattern: Condividi*

### Quale problema risolve?

Un widget di condivisione è un piccolo elemento grafico inserito all'interno del markup di un file di ipertesto che consente agli utenti di condividere le risorse di contenuti e informazioni con la Comunità, attraverso un altro sito oppure una piattaforma sociale come Facebook o MySpace [45].

Per esempio, il Widget Condividi consente all'utente di condividere il contenuto di una pagina (o un componente di una pagina) con gli amici su siti di social networking come MySpace e Facebook fungendo da ponte tra il contenuto di un determinato sito e le piattaforme di applicazione.

## Quando usare questo pattern?

Fornire un widget condivisione in contesti dove l'utente desidera invitare qualcuno a vedere qualcosa in uno spazio pubblico o condiviso a cui essi possono avere accesso.

In realtà un utente potrebbe semplicemente copiare ed in collare il collegamento ed inviarlo via email. Fornendo un meccanismo automatico, però, il sistema può tenere traccia di quello che viene condiviso.

Gli utenti meno esperti, inoltre, troveranno più semplice e veloce utilizzare un meccanismo guidato senza dover manipolare i collegamenti direttamente.

## Quale è la soluzione?

Fornire un pulsante o un collegamento con scritto “Condividi” (“Share”) o “Invia” o qualcosa di simile. Quando l'utente clicca sul pulsante, visualizzare una maschera sovrapposta che consenta l’invio. Facoltativamente è possibile includere altre utilità di oggetto, ad esempio "stampa" in questo stesso contesto.

*AddToAny offre un servizio Condividi per l'incorporamento sui siti .*

Gli utenti si aspettano delle facilitazioni nell’invio e nella condivisione di informazioni. Ricordate che ognuno è oberato da decine di siti collegati di cui spesso non si ricorda tutte le procedure. Se condividere un link è una operazione semplice ed immediata è più facile che un utente agisca di impulso in cambio di un gratificazione immediata.

## Perché usare questo pattern?

Incorporare widget condivisione quando si presenta un contenuto o provvedere a fornire ad altri un metodo per incorporare un collegamento al proprio contenuto può facilitare l’interazione e lo scambio sulla tua rete o latri servizi servizi di condivisione di file multimediali.

# Relazione

Il termine relazione ha una serie ampia di significati.

Nel caso specifico dei siti di comunità intendiamo una relazione che può essere:

- tra un membro ed un contenuto (relazione tra membri)
- tra un membro ed un altro membro (relazione ibrida)
- tra due contenuti (relazione tra contenuti)

Le caratteristiche comuni a tutte le relazioni sono:

- **tipologia** (descrive il tipo di relazione: amicizia, colleghi di lavoro, subordinato)
- **direzione** (la direzione indica il grado di subordinazione, può anche indicare il fatto che l'amicizia non è reciproca)
- **esplicita/implicita** (una relazione esplicita viene indicata in maniera chiara, ad esempio condividendo delle informazioni o accettando una richiesta di amicizia, la relazione implicita può invece derivare da aggregazioni o ricerca di affinità basati su diversi criteri)

# Navigazione

I contenuti della comunità possono essere molteplici, è dunque importante che siano previsti dei metodi per raggiungere il contenuto desiderato in modo veloce ed efficiente.

La distinzione qui riguarda il punto di partenza.

## Navigazione dalla home page

In questo caso non ci sono elementi per capire cosa l'utente desidera, al massimo, solo se si può associare a chi naviga un profilo noto, si può cercare di indovinare a partire dalla storia delle ricerche già fatte dallo stesso utente in passato.

- **Ricerca a testo libero (membri e contenuti):** permette di inserire alcune parole che saranno analizzate da un motore di ricerca interno che risponderà con una serie di risultati più o meno corrispondenti.
  Tipicamente l'oggetto della ricerca è un profilo (e quindi un altro membro della comunità) oppure un contenuto (o un raggruppamento di contenuti) o anche entrambi, in quest'ultimo caso si deve prevedere una visualizzazione semplice ma omogenea che permetta di visualizzare nella stessa maschera di risultati oggetti diversi.
- Il concetto di ricerca libera è molto complesso ed articolato, solo nei casi più semplici e con basi di dati molto ridotte è possibile ricercare direttamente il database, in modo più generale si basa sull'utilizzo e l'integrazione di prodotti di terze parti che gestiscono l'indicizzazione delle risorse (vedi lucene, rif. All'integrazione con webratio, paper: Integrating Databases, Search Engines and Web Applications, A Model-Driven Approach)
- **Per data (membri e contenuti):** per i contenuti fa fede la data di pubblicazione, quindi una tipica selezione è: contenuto più recente. Per i membri si può scegliere tra la visualizzazione degli ultimi iscritti, ultimi accessi, compleanni o attualmente online.
- **Per categoria, tassonomia (solo contenuti):** è composta da un albero di categorie e da una selezione di tutti I contenuti appartenenti alla categoria scelta. I risultati possono essere ordinati per data o per valore di classifica
- **Nuvola di tag, folksonomie (solo contenuti):** è una modalità di ricerca che propone una selezione dei tags più utilizzati evidenziando in maniera grafica il loro peso relativo. Ogni tag è modellizzato come un link che opera una selezione dei contenuti che sono etichettati con quel tag particolare.
- **Per mappa (solo contenuti):** per i contenuti che possiedono informazioni di geolocalizzazione è possibile pensare una visualizzazione basata su mappa che associa ad ogni contenuto un simbolo ed eventualmente una descrizione e relativo link per raggiungere una pagina in cui il contenuto viene descritto in maniera più completa. Richiede integrazione con prodotti cartografici di terze parti come google maps [18].

## Navigazione a partire dai contenuti selezionati

In questo caso esistono alcuni elementi che possono essere usati per navigare le relazioni esplicite o implicite che vengono usati come punto di partenza.

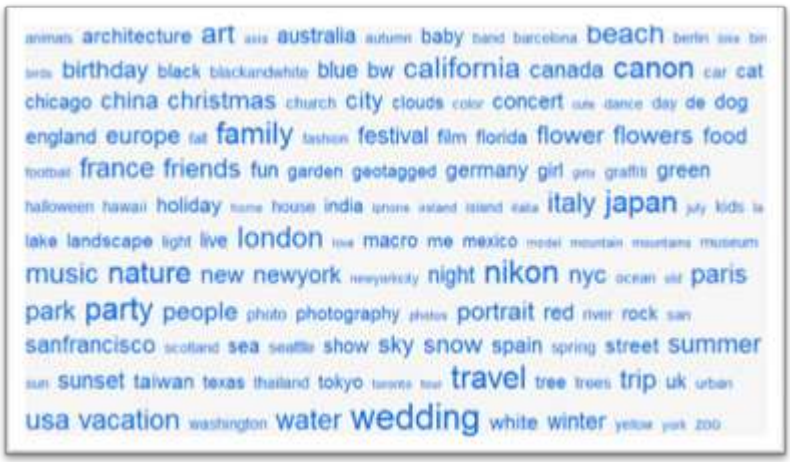
- **Stesso proprietario (contenuti):** seleziona tutti i contenuti pubblicati dallo stesso autore in ordine di data pubblicazione o classifica.
- **Per relazione esplicita (membri e contenuti):** nel caso di membri può essere la lista degli amici del membro evidenziato, nel caso di contenuti si possono avere relazioni esplicite quando un contenuto viene legato ad un altro da un membro (es: il video di risposta di youtube)
- **Per relazione implicita (membri e contenuti):** nel caso di membri può essere un suggerimento di amicizia (ad esempio una persona che ha parecchi amici in comune con un'altra, ma non è tra i sui amici) , nel caso di contenuti si possono avere relazioni implicite di contenuti che hanno uguale data data, stessi tag o appartengono alla stessa categoria.

## Pattern di navigazione

### *Pattern: Nuvola di tag*

Una nuvola di tag (tag cloud in Inglese) è una rappresentazione visiva delle etichette (tag) o parole chiave usate in un sito web. Il modello comunemente adottato è quello di presentare i tag in ordine alfabetico e quindi allargare il tag proporzionalmente in base alla popolarità. Si tratta quindi di una lista pesata. Questa presentazione è diffusa, ma non sempre facilmente comprensibile da parte di utenti e può rubare un sacco di spazio nell'interfaccia [31].

Le nuvole di tag costituiscono un nuovo elemento d'interfaccia per gli architetti dell'informazione, che le possono utilizzare per progettare navigazioni alternative all'interno di un sito web. C'è gran dibattito sul fatto che sia veramente utile, alcuni la ritengono soltanto un gadget bello da vedere ma poco utile nella pratica.




*Esempio: Tag cloud dei tag più usati da Flickr (http://www.flickr.com/photos/tags/)*

### Qual è il problema che risolve?

L'utente vuole conoscere tutti i tag associati a un oggetto, un luogo, una persona e quali sono usati più spesso.

### Quando utilizzare questo modello?

- Si utilizza quando si vuole presentare tutti i tag associati ad un oggetto.
- Si utilizza quando si vuole presentare tutti i tag associati a un sito.
- Si utilizza quando si vuole presentare il tag più popolari su un sito.
- Si utilizza quando si vuole presentare tutti i tags associati con l'identità di una persona.
- Si utilizza come meccanismo di navigazione per il recupero di contenuti.

### Qual è la soluzione che propone?

- Presenta i tag in una forma facile da comprendere.
- Molti siti presentano i tags in ordine alfabetico e quindi allargano proporzionalmente la dimensione il tag in base alla popolarità. Questa presentazione è diffusa, ma non sempre facilmente comprensibile da parte di utenti e può rubare un sacco di spazio nell'interfaccia.
- Se si usano i tag di dimensioni proporzionali in base alla popolarità è necessario definire l'algoritmo di dimensionamento per riflettere il senso di ciò che è popolare, piuttosto che la reale ripartizione dei tag. Utilizzando la distribuzione reale produrrebbe enormi differenze nelle dimensioni e sarebbe pesante senza fornire all'utente informazioni più significative da cui trarre conclusioni.

#### *Domande aperte*

L'utilità di una nuvola di tag è ancora in discussione. Sembrano pulite e graficamente piacevoli, ma le sfumature delle differenze di calibrazione sono spesso perse dall'utente medio. Se l'oggetto o il sito ha molti di tag, la nube diventa meno utilizzabile di un sito o un oggetto con pochi tag. Siti come Flickr, che ha migliaia di etichette, ha organizzato la presentazione dei tag dividendo quelli dello spazio pubblico "Esplora" e i tag personalizzati "i tuoi Tag", che presenta un sottoinsieme dei tag più utilizzati per gli oggetti di una persona. In entrambi i casi, essi presentano un piccolo sottoinsieme del totale.

Inoltre, alcuni siti visualizzano i tag di dimensioni diverse per rappresentare con una certa enfasi altri dati oltre alla popolarità. La mancanza di un significato standard nella presentazione delle nuvole di tag rende difficile per gli utenti finali sapere cosa aspettarsi o essere in grado di predire il significato insito nella visualizzazione.

### Perché usare questo modello?

La presentazione a nuvola di tag, resa popolare da Flickr, è un buon modo per mostrare agli utenti una rappresentazione visiva dei concetti popolari agli oggetti di una persona o il sito. Una nuvola di tag incoraggia la navigazione e l'esplorazione in un modo alternativo di standard di navigazione del sito e supporta la scoperta casuale.

# Comunicazione

La comunicazione avviene tra membri della stessa comunità o con soggetti terzi.

## Comunicazione tra membri

La comunicazione può avvenire tra membri, ed in questo caso è di due tipi:

- **sincrona**: in tempo reale, interazione diretta tra due utenti nello stesso momento utilizzando strumenti proposti dalla comunità o da terze parti (chat, video conferenza, skype, messenger, etc..)
- **asincrona**: utilizzando un sistema di messaggistica interna alla comunità o sfruttando canali esterni, tipicamente il servizio di email in una delle sue molte declinazioni o varianti (client di posta pop3, imap, web mail, etc..).

## Comunicazioni di sistema

La comunicazione può avvenire anche tra il sistema ed i suoi utenti, ed avviene in questo caso generalmente in maniera asincrona.

- **notifiche:** il sistema può comunicare attraverso notifiche (nuovo contenuto disponibile, nuovi messaggi, aggiornamenti, richieste di amicizia o di altro tipo…)
- **messaggi:** messaggi recapitati nella casella di posta interna o via email legati alle attività compiute nel sito (richiesta password dimenticata, nuova registrazione, ban, etc…)

# WebRatio Community Patterns

WebRatio Community Patterns è un tentativo di fornire un punto di partenza efficace per chiunque intenda creare o integrare funzionalità di tipo "Comunity Web 2.0" all'interno di applicazioni Web grazie al metodo di sviluppo visuale che contraddistingue la piattaforma WebRatio.

WebRatio Community Patterns è una collezione di unità custom, stili ed una applicazione di esempio completa di modello dati che permettono di implementare in maniera rapida ed efficace i più noti ed usati pattern di interfaccia di tipo sociale che comunemente si vedono nei siti di comunità.

## Una comunità per la pubblicazione di racconti

In questa parte del lavoro presento una carrellata di applicazioni pratiche dei pattern che nella sezione precedente sono stati discussi solo in teoria.

Il progettino di Webratio qui presentato non deve essere visto come una applicazione finita quanto piuttosto una collezione di elementi, concetti e pratiche ottimali che servano come un punto di partenza per sviluppare la propria comunità web virtuale.

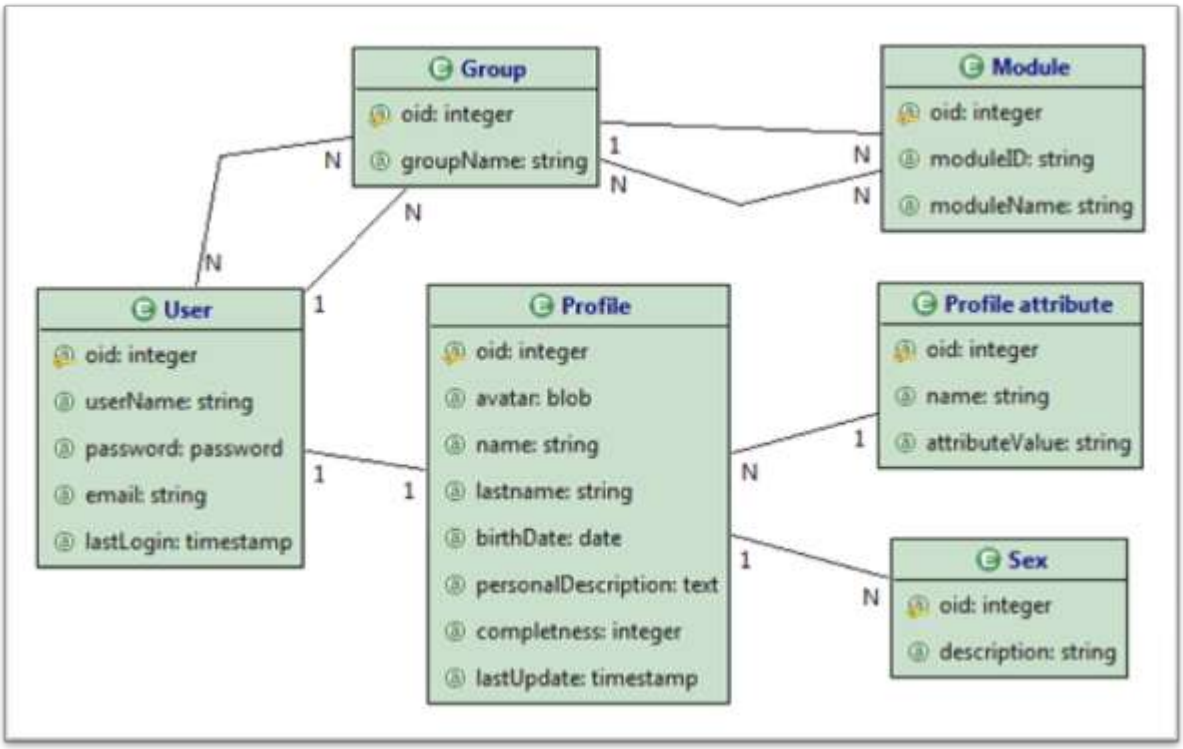
A titolo esemplificativo ho immaginato una ipotetica comunità di scrittori che vogliano condividere le loro opere all'interno di un sito web comunitario.

Il racconto (o poesia o scritto in genere) è il contenuto che viene condiviso dai membri. Ogni racconto può essere categorizzato usando i patterns relativi ai tags, può essere valutato usando i patterns di valutazione e recensione, può essere contestualizzato nel tempo e nello spazio usando i tag calendario e geotagging.

*La testata della comunità virtuale usata come esempio*

# Il modello dei dati

## Il profilo




*Il modello dei dati relativo al profilo utente*

Il modello rappresentato in figura prevede una struttura generale per memorizzare il profilo di un utente.

Le entità Utente *(User)*, Gruppo *(Group)* e Modulo *(Module)* sono standard in Webratio e permettono di gestire l'utente, l'accesso al sistema ed i permessi relativi.

### *Entità profilo dell'utente*

La struttura Profilo *(Profile)* memorizza le basi per il profilo degli utenti.

#### Dati personali

- **Name** (string), **LastName** (string)
  Nome e Cognome dell'utente.
- **Avatar** (blob)
  Contiene l'immagine del profilo.
- **BirthDate** (date)
  Data di nascita.

### Elementi di partecipazione alla comunità

- **PersonalDescription** (text)
  Testo descrittivo dove l'utente può presentare se stesso, i suoi interessi e le sue aspirazioni come membro attivo della comunità.
- **Completness** (integer)
  Percentuale di completamento del profilo.
- **LastUpdate** (timestamp)
  Data di ultima modifica del profilo. Utile per mostrare la data dell'ultimo aggiornamento ma importante anche in collegamento con la percentuale di completamento ma anche come indice di reputazione.

### *Entità attributi del profilo*

E' una entità collegata al profilo e permette di gestire come coppia chiave/valore attributi non previsti inizialmente nello schema del profilo. Usando questa tabella non è necessario modificare il modello dati e il database.

### *Entità sesso*

Piccola entità che lista i possibili valori del campo sesso nel profilo secondo il codice standard internazionale ISO 5218 che definisce una rappresentazione dei sessi attraverso un linguaggio codificato a carattere unico.

I quattro codici specificati nell'ISO 5218 sono:

0 = sconosciuto

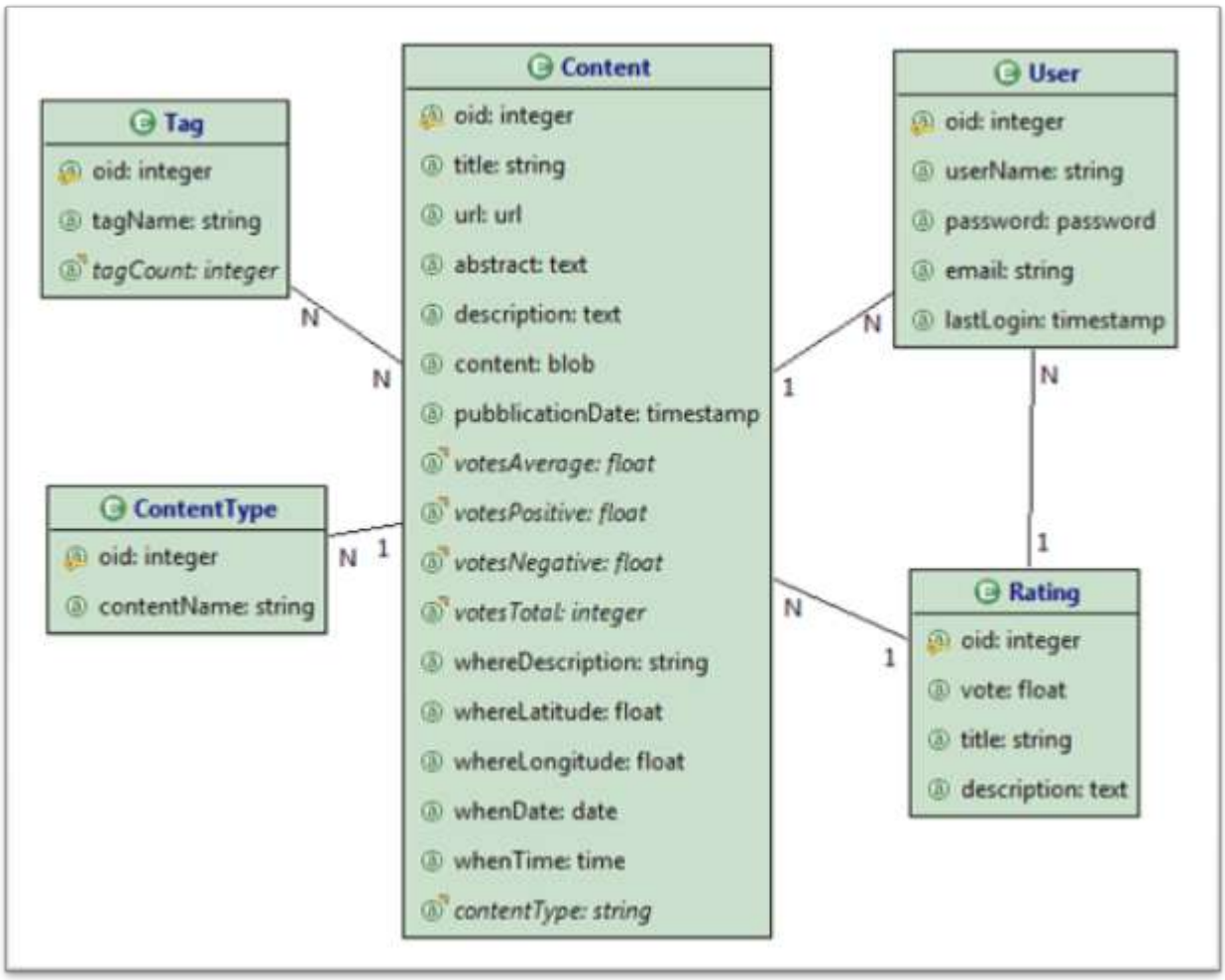
1 = maschio

2 = femmina

9 = non specificato

- **oid** (integer)
  Codice ISO 5218
- **description** (string)
  Etichetta ISO 5218

# Il contenuto




*Il modello dei dati relativo al contenuto.*

Il modello rappresentato in figura prevede una struttura generale per memorizzare il contenuto di una comunità.

### *Entità Contenuto*

La struttura *Content* memorizza le basi per il contenuto pubblicato dagli utenti.

#### Elementi propri del contenuto

- **Title** (string)
  Titolo o nome del contenuto.
- **Url** (url)
  Link a risorsa (pagina web, video o immagine o altro contenuto disponibile su web).
- **Abstract** (text)
  Descrizione breve del contenuto.

- **Description** (text)
  Descrizione testuale completa del contenuto, ammette un numero anche molto grande di caratteri (può essere un intero articolo o racconto).
- **Content** (blob)
  Campo che permette la memorizzazione nel database di contenuto binario (immagini o altro).

### Elementi legati all'autore ed alla data di creazione:

- **Author** (chiave esterna su users in relazione N-1)
  Autore del contenuto.
- **PubblicationDate** (timestamp)
  Data di pubblicazione.

### Classificazione dell'autore

- **WhereLatitude** (float), **WhereLongitude** (float)
  Rappresenta una descrizione ed una coppia di coordinate indicanti un luogo utile per rappresentare l'oggetto in una mappa collegata con relazione 1-N.
  Scenari di utilizzo potrebbero essere delle foto o video turistici a cui abbinare il luogo in cui il video è stato girato o è stata scattata la foto.
  Altro utilizzo potrebbe essere la localizzazione di manifestazioni o eventi.
  La relazione inversa multipla permette di aggregare più oggetti legati alla stessa località. Per rendere le cose semplici ogni contenuto può avere una sola località.
  L'aggregazione per località si presuppone vada chiamata con la clausola DISTINCT.
- **WhenDate** (date), **WhenTime** (time)
  Rappresenta una data ed un ora legata al contenuto.
  Scenari possibili sono organizzazioni di eventi o appuntamenti legati al contenuto oppure la contestualizzazione temporale dello scatto di una foto o di una ripresa video.
  Ho preferito tenerli distinti in modo che si possano aggregare contenuti anche soltanto per data. L'aggregazione in calendari o simili si presuppone vada chiamata con la clausola DISTINCT.
- **Tag_Content** (tag)
  Elenco dei tag collegati (vedi descrizione entità tags)

### Classificazione della comunità

- **VotesAverage** (float)
  La media dei voti conseguiti dal contenuto collegato
- **VotesPositive** (float), **VotesNegative** (float)
  Rappresenta la somma dei voti positivi o negativi da utilizzare con un sistema di votazioni "Mi piace" / "Non mi piace".
- **VotesTotal** (integer)
  Il numero di voti espressi per il contenuto.

- **Ratings_Content** (ratings)
  Elenco dei voti collegati (vedi descrizione entità valutazioni).

### *Entità Tag*

La classificazione per tags o categorie è memorizzata in una apposita entità in relazione 1-N con l'entità contenuto.

#### Classificazione dell'autore per categorie o tags

- **TagName** (string)
  Nome del tag, normalizzato
- **TagCount** (int)
  Conteggio del numero di contenuto collegati, necessario all'implementazione del pattern "Nuvola di Tags".

### *Entità Valutazioni*

La classificazione della comunità è memorizzata in una apposita entità che gestisce sia le votazioni che le recensioni. Tale entità è in relazione 1-N.

La votazione nella tabella esterna è un valore assoluto intero positivo (per votazioni con stelle) oppure +/-1 (per votazioni tipo "Mi piace" / "Non mi piace").

#### Classificazione della comunità

- **Vote** (float)
  La valutazione numerica espressa da un utente relativa al il contenuto.
- **Title** (string)
  Titolo di una eventuale recensione.
- **Description** (text)
  Testo di una eventuale recensione.
- **Rating_User** (user)
  Utente che ha espresso la valutazione. Viene anche usato per impedire votazione multiple se richiesto.

### *Entità Tipo contenuto*

Piccola entità che lista i possibili valori del campo tipo contenuto della entità contenuto.

- **oid** (integer)
  Chiave numerica
- **ContentName** (string)
  Descrizione del contenuto

# Modellazione in Webratio

## Descrizione dei modelli

Ogni implementazione pratica di modello qui descritta segue la medesima struttura per essere di facile riferimento.

### *Dati utilizzati*

Nella parte relativa ai dati ho descritto tutti i parametri relativi al modello, divisi in tre gruppi. Per ogni parametro ho specificato il nome, il tipo di dato e una breve descrizione.

Un esempio è visibile nella riga seguente:

- **Parametro** (string|float):
  breve descrizione indicante la funzione del parametro.

Per convenzione ho mantenuto la lingua inglese nel nominare i parametri, scelta naturale dal momento che Webratio è in inglese così come la maggior parte delle biblioteche di funzioni aggiuntive utilizzate.

#### Parametri statici

I parametri statici sono impostati dal progettista al momento della aggiunta dell'unità in Webratio, vengono generati e compilati con il progetto e non cambiano più.

#### Parametri dinamici in ingresso

I parametri dinamici in ingresso dipendono dal contesto e dalle unità collegate. Tipicamente variano in base al variare di dati presenti nel database. Il progettista in Webratio decide soltanto da dove questi dati arrivano (da una entità oppure da un input inserito da un utente) e questi verranno poi dinamicamente calcolati in fase di esecuzione.

#### Parametri dinamici in uscita

I parametri dinamici in uscita, come quelli di ingresso, dipendono dal contesto e dalle operazioni che l'utente fa sulla unità interessata (inserendo dati in input, facendo clic con il mouse o attivando un link). Il valore risultante viene esposto sui link di uscita dell'unità (link normali o di trasporto) e reso disponibile ad altre unità che possono, ad esempio, memorizzarlo su database.

### *L'implementazione WEBML*

Nella parte relativa all'implementazione descrivo una pagina tipo in linguaggio WebMl come si presenta in Webratio, descrivendo le unità coinvolte nel modello e la loro interazione.

### *La visualizzazione sulla pagina*

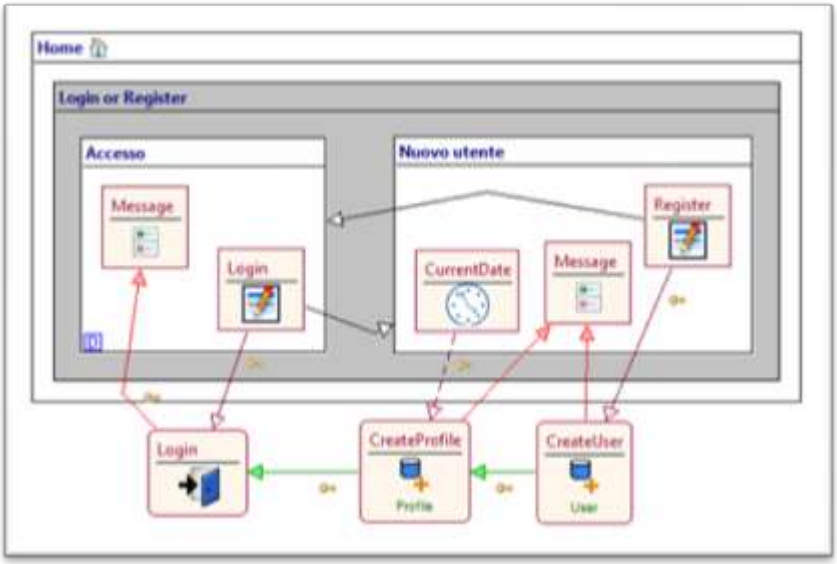
La visualizzazione sulla pagina presenta un possibile utilizzo del modello all'interno di una vera applicazione Webratio. La grafica è volutamente semplice e pulita perché lo scopo del lavoro è pret-

tamente funzionale e legato al modello dei dati ed alla dinamica dell'interazione con l'utente, cercando di evidenziare le interazioni che con una grafica più ricercata sarebbero potute apparire poco evidenti.

## Modello accesso e registrazione utenti

### *L'implementazione WEBML*

La *Site view* pubblica è costituita da una sola pagina che, a sua volta contiene due pagine a scelta mutuamente esclusiva che permettono di accedere al sistema, per gli utenti già in possesso di credenziali valide, oppure di registrarsi, per utenti al loro primo accesso. Le due pagine sono collegate tra loro da opportuni links.




*Il primo accesso al sistema: autenticazione e registrazione*

Nel caso di una nuova registrazione il procedimento è ridotto al minimo. Contestualmente al nuovo utente viene creato anche un nuovo profilo ed aggiornata data ed ora correte (importante: la Time unit fornisce l'ora corrente solo se è posizionata all'interno di una pagina).

### *Visualizzazione sulla pagina dell'accesso utenti registrati*

*La schermata di autenticazione che accoglie gli utenti.*

Tutto il contenuto del sito è protetto ed è richiesta la registrazione dei nuovi utenti.

### *Visualizzazione sulla pagina della registrazione nuovi utenti*

Il numero delle informazioni richieste è ridotto la minimo, l'utente potrà aggiungere altre informazioni successivamente.

*La maschera di registrazione nuovo utente.*

Espandere con:

descrizione, collegamento a patterns, accettazione privacy

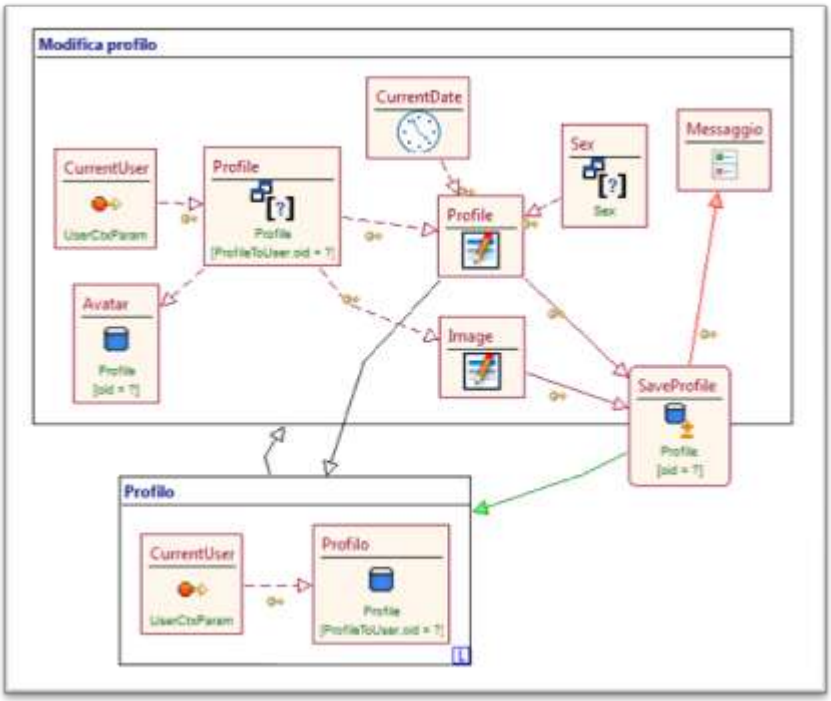
## Modello profilo utenti

Nella pagina profilo l'utente ha la possibilità di modificare i propri dati personali.

### *L'implementazione WEBML*

La modifica del profilo è gestita attraverso una Entry unit. Il valore di sesso, collegato all'entità Sex, è inserito come selettore a discesa. Una unità Time aggiorna la data dell'ultima modifica.

Ho dovuto separare l'invio dell'immagine dalla modifica dei dati testuali. Se non avessi fatto in questo modo ogni salvataggio dei dati avrebbe richiesto anche un nuovo invio dell'immagine. In questo modo l'immagine è visualizzata, usando una data unit, e, usando una Entry unit, l'utente ha la possibilità di modificarla ogni volta che lo desidera.




*Il diagramma WebMl relativo alla visualizzazione ed alla modifica del profilo utente*

Aggiungere: modifica password

Aggiungere: visualizzazione profilo altri utenti

### *La visualizzazione sulla pagina*

*La visualizzazione del profilo utente, il link a sinistra permette di modificare il proprio profilo.*

*La modifica del profilo*

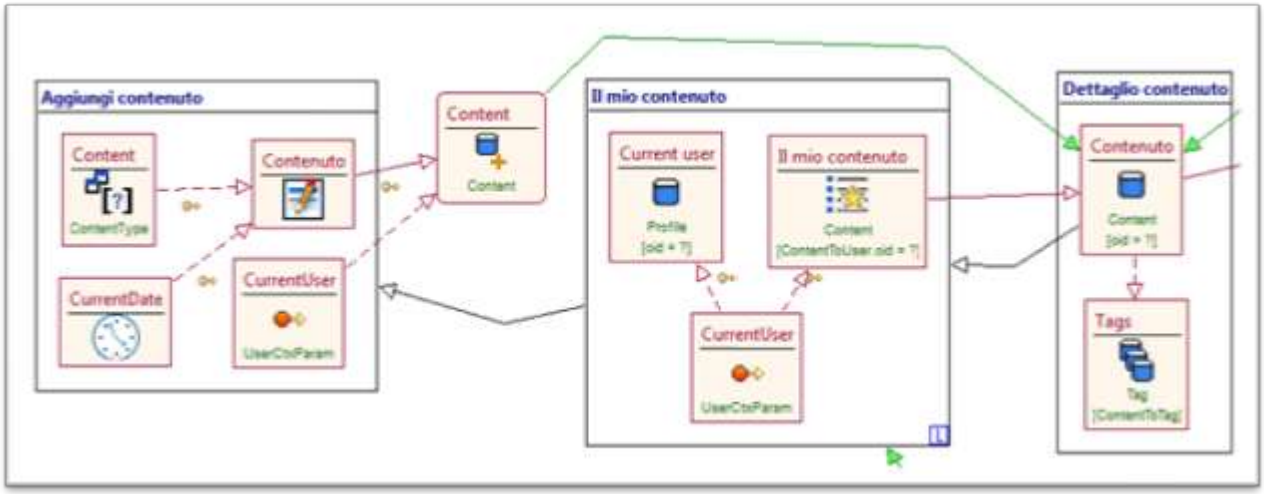
## Modello gestione contenuto

### *L'implementazione WEBML*

La gestione contenuto è composta da quattro pagine.

- Nella pagina "Il mio contenuto" è presente l'elenco di tutto il contenuto prodotto dall'utente corrente.
- La pagina "Aggiungi contenuto" permette di aggiungere nuovi elementi.

- La pagina “Dettaglio contenuto” permette di visualizzare gli elementi esistenti.




*Elenco contenuti personali e aggiunta nuovo contenuto*

## *La visualizzazione sulla pagina*

*L’elenco dei contenuti personali di ogni utente*

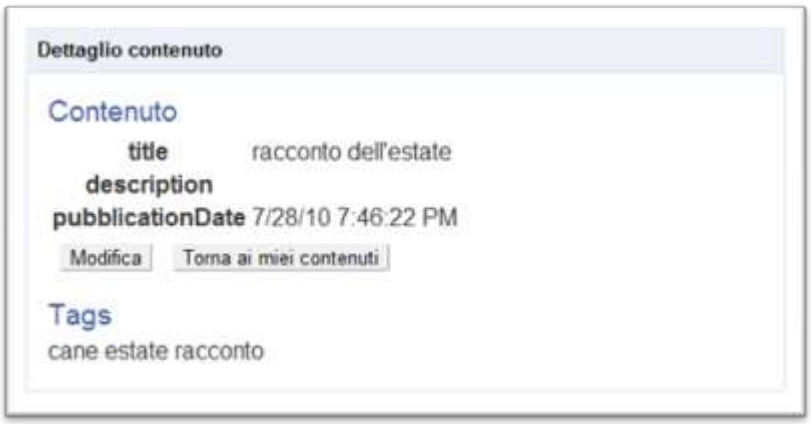
La scritta “Il contenuto di Emmanuele De Andreis” è ricavata utilizzando un apposito template grafico di unità che permette di stampare un messaggio ed un elenco di attributi senza soluzione di continuità.

La parte rilevante del template è la seguente. Notare la condizione di stampa del messaggio, definito come attributo personalizzato di layout.

```
<wr:Frame>
     [% if (useUnitMessage == "true") { printJSPTagValue(UnitMessage); } %]
     <wr:Iterate var="attr" context="unit" select="layout:Attribute">
            <wr:Visible>
                   <span class="<wr:StyleClass/> value[%= attr["type"]%]"><wr:Value/></span>
            </wr:Visible>
     </wr:Iterate>
</wr:Frame>
```

*Aggiungi contenuto*

Aggiungi contenuto permette di aggiungere un numero minimo di informazioni e soprattutto di specificare il tipo di contenuto, informazione che non sarà più possibile in seguito modificare e potrebbe essere usata per indirizzare su differenti pagine di modifica, ognuna specifica per il singolo contenuto.




*La pagina di visualizzazione contenuto*

Anche per la pagina di visualizzazione contenuto ho creato un layout personalizzato per una multi data unit, con l'unica differenza che per una multi data unit è necessario un ciclo ulteriore sulle varie entità.

## Modello modifica contenuto e assegna tag

Nella applicazione demo è presente una pagina che permette di modificare il contenuto ed assegnare contestualmente i tags, come descritto nel pattern "Assegna un tag ad un oggetto".

### *Dati utilizzati*

#### Parametri dinamici in ingresso

Parametri in ingresso sono la lista attuale di tags assegnati all'oggetto ed il relativo oid di filtro.

- **Oid (id del contenuto)** (integer):
  presente all'ingresso del selettore dei tags, serve a filtrare la lista in modo che contenga soltanto i tags assegnati all'oggetto in corso di modifica.
- **TagName** (array of string):
  La lista dei tags correntemente assegnati all'oggetto in corso di modifica.

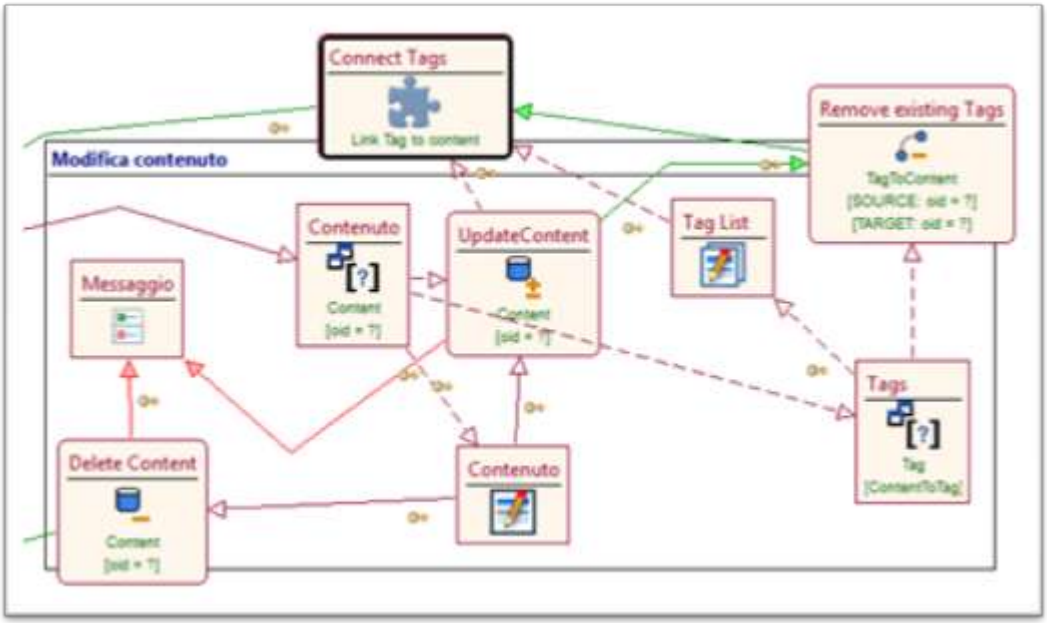
#### Parametri dinamici in uscita

- **TagName** (array of string):
  La lista dei tags modificati dall'utente all'oggetto in corso di modifica. Contiene la lista completa dei tags, quelli esistenti, quelli appena aggiunti e ovviamente non contiene quelli rimossi

### *L'implementazione WEBML*

Per implementare il pattern "Assegna Tag" non è necessario creare una unità personalizzata ma possiamo utilizzare gli strumenti standard messi a disposizione da webratio.

#### Prima fase: input dei dati

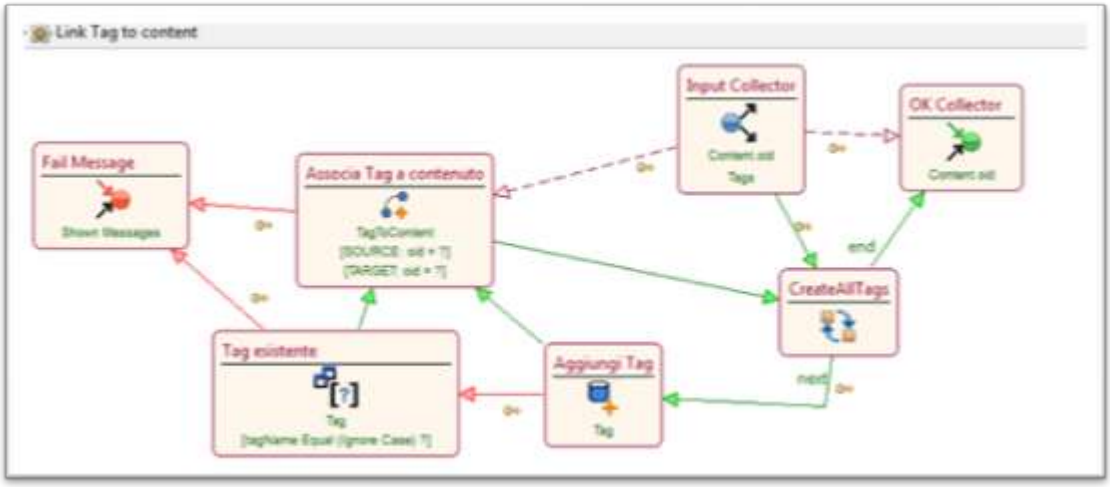



*Il diagramma WebMl di salvataggio del contenuto e la relative assegnazione dei tags*

I tags vengono inseriti tramite una “Multi entry unit” che gestisce autonomamente la rimozione e l’aggiunta di nuovi elementi.

### Seconda fase: memorizzazione e assegnazione dei tags

La logica di assegnazione non è semplicissima per cui ho preferito includere la parte più complessa (ma anche quella più riutilizzabile) in un modulo dedicato che si dedica al lavoro di assegnazione vero e proprio.




*Il modulo di collegamento tra contenuto e categorie*

L’assegnazione viene eseguita in tre passaggi (di cui gli ultimi due, ciclici, eseguiti per ogni tag da assegnare):

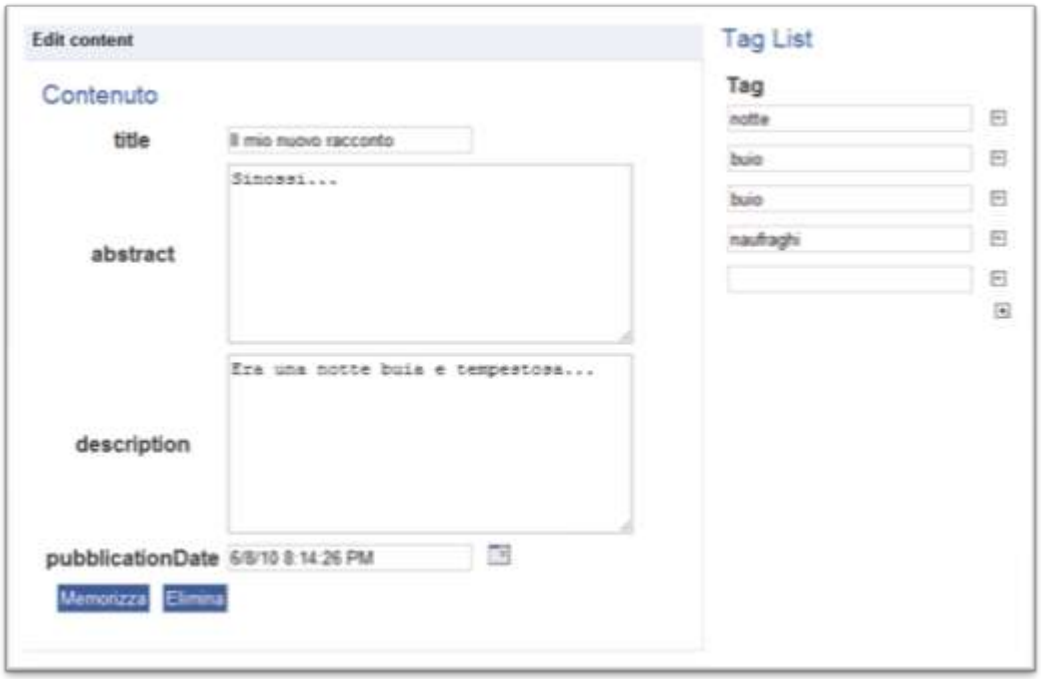
1. Scollegamento di tutti i tag eventualmente presenti (per evitare doppie assegnazioni e gestire i tag rimossi)
2. Creazione del tag (se non esiste)
3. Collegamento del tag al contenuto

L’aggiornamento del contenuto è la prima operazione (Update Content), dopo di che vengono rimossi tutti i tags in precedenza assegnati (Remove existings tags) che sono poi riconnessi utilizzano un modulo appositamente progettato (Connect Tags). La rimozione è necessaria per rimuovere correttamente eventuali tag cancellati.

La progettazione del modulo è interessante perché è un buon esempio di come gestire le condizioni di successo e di fallimento tramite appositi ok e ko collectors. Dimostra anche come utilizzare una loop unit per eseguire un processo ripetitivo come la creazione di tutti i tag richiesti.

Condizione necessaria per il buon funzionamento del modulo è che il nome del tag sia associato ad un indice univoco sul database, questo permette di gestire il pattern “aggiungi se non esiste”: se l’aggiunta fallisce vuole dire che il tag già esiste e quindi lo recuperiamo con un selector, se non esiste ancora la create unit darà esito positivo e sarà necessario soltanto collegare il tag appena creato.

### *La visualizzazione sulla pagina*




*A sinistra un contenuto in modalità modifica, a destra la lista dei tag assegnati*

In figura vediamo come si presenta all'utente l'interfaccia di assegnazione.

Con il pulsante [-] è possibile rimuovere un tag assegnato in precedenza, con il pulsante [+] posto alla fine è possibile aggiungere un numero teoricamente illimitato di nuovi tags.

Eventuali duplicati assegnati allo stesso oggetto saranno automaticamente rimossi.

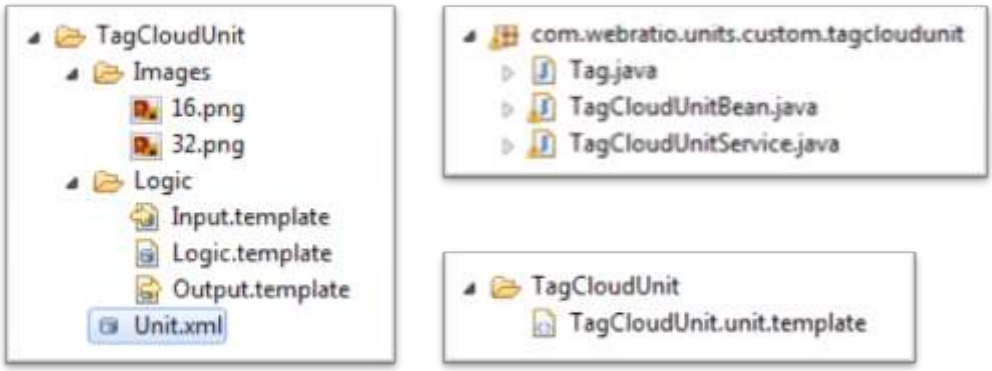
Il pulsante di memorizzazione del contenuto attiva anche la procedura di memorizzazione dei tag.

## Modello nuvola di tags

Implementa una nuvola di tag secondo il pattern "naviga per tag". La dimensione relativa dei font è calcolata in base ad un algoritmo che permette di visualizzare in dimensione relativa più grande i tags usati da un numero maggiore di contenuti.

### *Descrizione dell'implementazione*

Per implementare questo modello ho creato una unità personalizzata che incapsula la maggior parte della logica di creazione e attivazione del modello.




*Le varie parti che compongono l'unità personalizzata "Nuvola di Tags" (TagCloudUnit)*

## *Dati utilizzati*

### Parametri statici

- **minFontSize**: (integer, default: 10)
  Dimensione massima del font (raggiunta dalla frequenza massima)
- **maxFontSize**: (integer, default 36)
  Dimensione minima del font (raggiunta dalla frequenza minima)

### Parametri dinamici in ingresso

In ingresso abbiamo una collezione di tre parametri sotto forma di array (che devono necessariamente essere della stessa dimensione) che rappresentano rispettivamente id, nome e conteggio tag.

- **oid (da Tag)**: (array of integer)
  Chiave numerica del tag
- **Tag Name (da Tag)**: (array of string)
  Valore testuale del tag
- **Count (da Tag)**: (array of integer)
  Numero di occorrenze del tag

### Parametri dinamici in uscita

In uscita viene esporto l'id del tag cliccato che può essere usato per visualizzare il dettaglio del tag usando una data unit.

- **oid (da Tag)**: (integer)
  Chiave numerica del tag selezionato

### L'algoritmo di rendering

In linea di principio, la dimensione del carattere di un tag in una tag cloud è determinata dalla sua incidenza. Per una nuvola di parole che rappresenta le categorie di un blog, ad esempio, la frequenza d'uso corrisponde al numero di blog che sono assegnati ad una categoria. Per le frequenze

piccole è sufficiente usare la frequenza stessa come dimensione, fino ad una dimensione massima prestabilita. Per valori più grandi è necessario un ridimensionamento.

E' assegnata una grandezza minima del font e una massima. Esiste un numero minimo e uno massimo di occorrenze di tag.

L'algoritmo di normalizzazione più semplice è quello lineare: si pone la grandezza del font sulle ascissa e il numero di occorrenze sull'ordinata, unendo i due punti di minimo e di massimo si ottiene una retta, ovvero una funzione che dato il numero di occorrenze ritorna la grandezza del font corrispondente.

$$s_i = \left[ \frac{(f_{\max} - f_{\min}) \cdot (t_i - t_{\min})}{t_{\max} - t_{\min}} \right] + f_{\min}$$

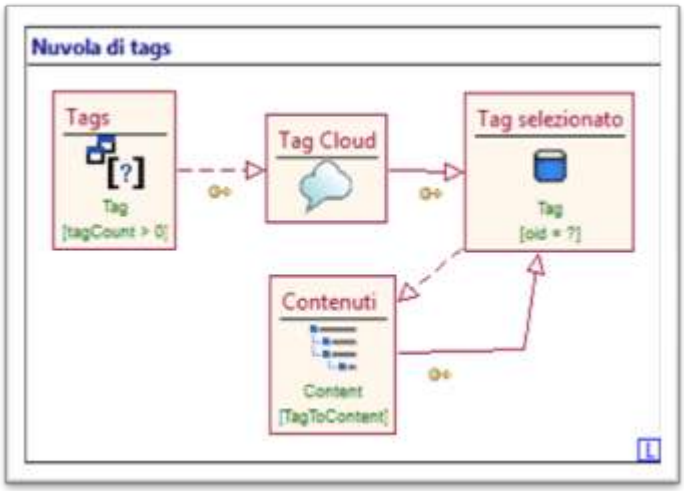
- $s_i$: Dimensione del font risultante
- $f_{max}$: Dimensione massima del font (raggiunta dalla frequenza massima)
- $f_{min}$: Dimensione minima del font (raggiunta dalla frequenza minima)
- $t_i$: numero di occorrenze del tag in oggetto ($t_{max} < t_i < t_{min}$)
- $t_{min}$: numero minimo di occorrenze di ogni tag (predefinito: 1)
- $t_{max}$: numero massimo di occorrenze di ogni tag (predefinito: 1)

Lo stesso algoritmo si presenta nel modo seguente in codice java:

```
public Integer getFontSize() {
        return ((maxFontSize - minFontSize) * (tagCount - minTagCount) /
                        (maxTagCount - minTagCount)) + minFontSize;
}
```

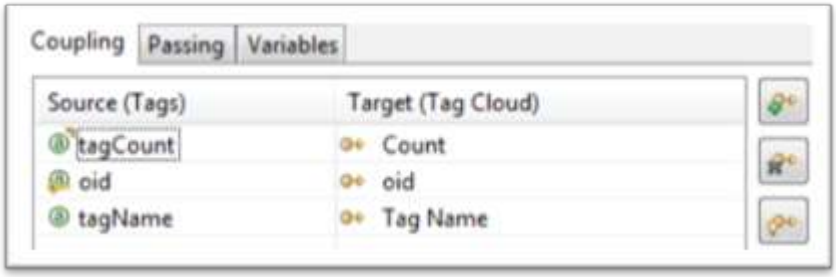
Poiché il numero di articoli indicizzati per descrittore di solito è distribuito secondo una legge di potenza, per intervalli di valori più grandi potrebbe aver senso una rappresentazione logaritmica.

## *L'implementazione WEBML*




*La pagina di esempio contenente l'unità personalizzata nuvola di tags*

L'unità "nuvola di tags" posizionata nella pagina deve essere collegata ad una sorgente dati (nell'esempio ho usato una unità selettore) che supporti un entità di tipo tag che contenga almeno i tre campi indicati nei parametri dinamici in ingresso (oid, tagName e tagCount). Attraverso un link in ingresso la nuvola raccoglie in input i tre valori (tutti obbligatori). E' raccomandato l'ordine alfabetico per i tags in modo che sia facile per gli utenti trovare i tag di proprio interesse.




*L'accoppiamento dei parametri*

Nella fase di rendering, della quale è responsabile il template predefinito, vengono scritti i vari tag sotto forma di links usando il peso di font relativo come descritto dall'algoritmo.

Ogni nome è un link che cliccato presenta il suo id come output e lo rende disponibile alle altre unità. In questo esempio ho collegato una data unit che visualizza il nome del tag e il numero di occorrenze presenti.

### *La visualizzazione sulla pagina*

*Nell'immagine vediamo al centro la nuvola, a destra il contenuto collegato in una unità gerarchica (contenuto/tags). A sinistra il tag correntemente selezionato.*

A titolo di esempio ho creato una pagina che dimostra il funzionamento della unità in un contesto operativo. I contenuti associati al tag selezionato sono elencati nella colonna di destra usando una unità gerarchica che al primo livello ha il contenuto ed al livello più interno i tags corrispondenti.

A sinistra sono visibili i dettagli del tag selezionato.

## Modello geomapping (Google Maps)

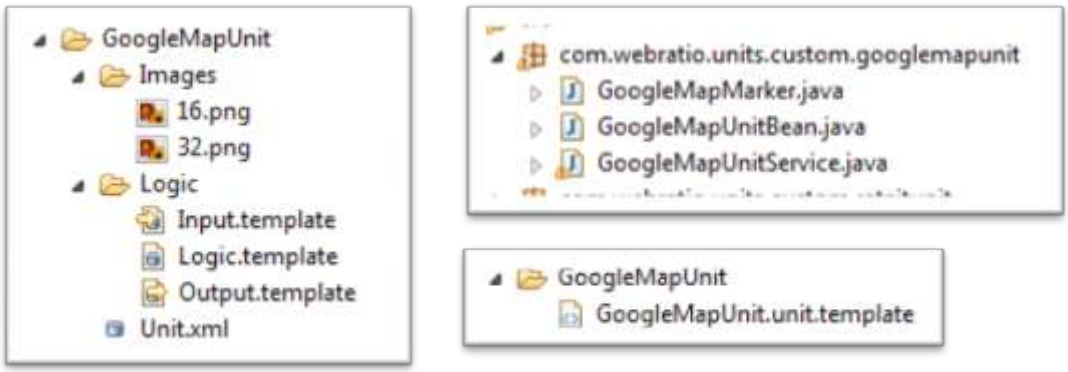
L'unità geomapping si avvale di Google Maps [18] (GM in seguito), un'applicazione di mappatura liberamente disponibile sul Web, che permette la ricerca e la visualizzazione di informazioni geografiche, interagendo con un'interfaccia utente piacevole e funzionale.

Alcuni dei vantaggi più importanti provenienti dall'adozione di GM sono l'aggiornamento costante dei dati geografici (ad es.: strade) e l'interfaccia utente accattivante, che sfrutta le più avanzate tecnologie Web [52]. Un altro vantaggio deriva dalla diffusione di GM. In realtà, sua adozione diffusa dagli utenti di Internet ha reso GM lo standard de facto per la presentazione di informazioni geografiche.

Questa diffusione è anche a causa di Google politica di licenze che consente agli sviluppatori di integrare liberamente GM nelle loro applicazioni, attraverso le API Google Maps [50]. Queste API forniscono una serie di utilità per la manipolazione delle mappe e aggiunta di contenuti a loro attraverso una varietà di servizi. La versione utilizzata dalla unit custom è la Google Maps JavaScript API V3.

### *Descrizione dell'implementazione*

Per implementare questo modello ho creato una unità personalizzata che incapsula la maggior parte della logica di creazione e attivazione del modello.




*Le varie parti che compongono l'unità personalizzata "Google Maps" (GoogleMapUnit)*

### *Dati utilizzati*

#### Parametri statici

Il progettista identifica i dati relativi alla posizione della mappa, al fattore di ingrandimento e ad una serie di opzioni accessorie relative alla visualizzazione della mappa.

- **Google key**: (string)
  Chiave alfanumerica necessaria per usare la API su un url diverso da localhost [51]

- **Width**: (integer, default: 400px) **Height**: (integer, default: 400px)
  Larghezza e altezza della mappa
- **Latitude**: (float, default: 42) e **Longitude**: (float, default: 13)
  Latitudine e longitudine del centro della mappa
- **Zoom factor**: (integer, default: 13)
  Fattore di zoom
- **Center on markers**: (boolean, default: true)
  Definisce se la mappa va centrata in modo che tutti i markers siano visibili invece che ai valori di latitudine e longitudine predefiniti.
- **Draggable**: (boolean, default: true)
  Definisce se la mappa è trascinabile o è fissa
- **Scroll when zoom**: (boolean, default: true)
  Regola la possibilità di spostare la mappa
- **Scale**: (boolean, default: true)
  Determina se visualizzare o meno la scala graduata
- **Overview**: (boolean, default: true)
  Determina se visualizzare o meno l'anteprima ridotta
- **Map control**: (selettore, default: Small):
  **Off**: disattivato, **Small**: piccolo, **Large**: grande
- **Map Type**: (string, uno o più dei seguenti valori: map;satellite;hibrid;)
  Lista delle visualizzazioni possibili

### Parametri dinamici in ingresso

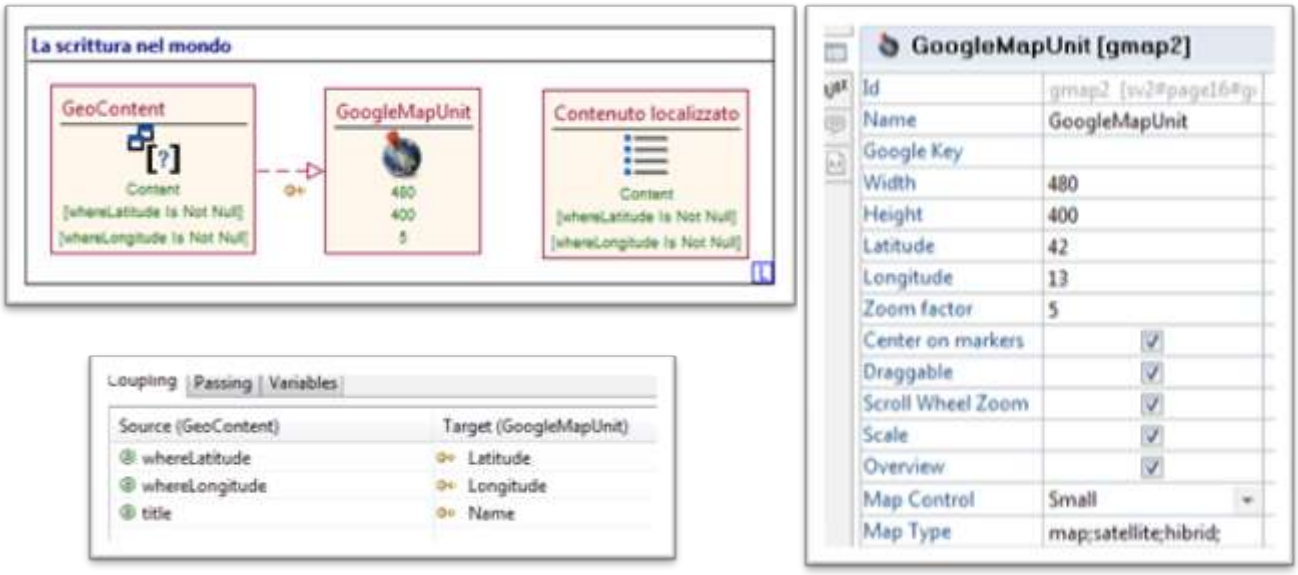
In ingresso viene accettata una lista di punti, che possono essere forniti ad esempio da un selector, formati da una coppia di coordinate (latitudine e longitudine) e da un testo descrittivo.

- **Latitude**: (array of double) e **Longitude**: (array of double)
  Latitudine e longitudine del marker da aggiungere
- **Name**: (string)
  Testo descrittivo visualizzato al passaggio del mouse

## *L'implementazione WEBML*

La visualizzazione proposta comprende una mappa singola di cui il modellatore ha specificato in Webratio lo zoom, il centro ed altri parametri caratteristici (come ad esempio le dimensioni). Tali valori sono visibili anche ne diagramma, sotto l'icona.

La mappa accetta in input un vettore di coordinate e relative descrizione e disegna una serie di markers in corrispondenza dei singoli punti.




*A sinistra il diagramma webml del selettore collegato alla unità google map*
*In basso a sinistra l'accoppiamento dei parametri per i marker.*
*A destra i parametri statici dell'unità.*

## *La visualizzazione sulla pagina*

Il risultato finale nella nostra applicazione di esempio. Vediamo la mappa rende rizzata con un paio di punti. Passando con il mouse sopra i punti viene evidenziato il nome del punto corrispondente.

*Un esempio di mappa con due contenuti localizzati. L'elenco completo è visibile sulla destra.*

## Modello valutare un oggetto

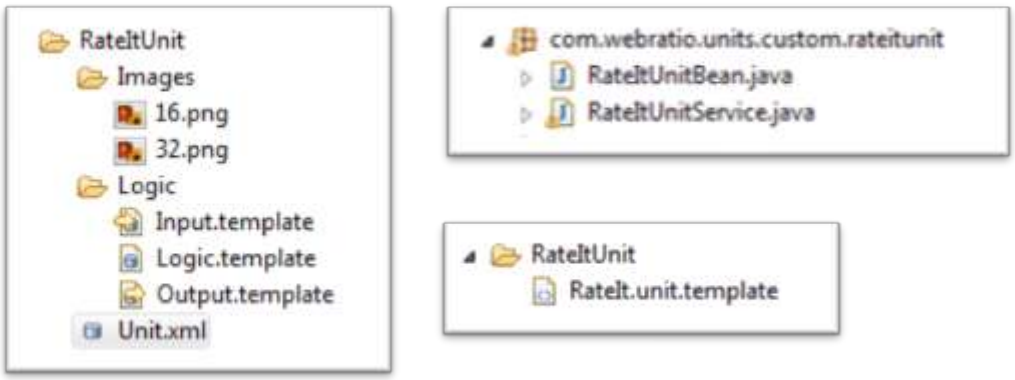
L'implementazione il pattern di valutazione (che può facilmente essere esteso anche a mi piace/non mi piace, e recensioni) identifica una serie di oggetti cliccabili (stelle) che si illuminano al passaggio del mouse ed invitano l'utente a valutare il contenuto (ad esempio, "Vota!").

Per l'implementazione ho utilizzato la biblioteca di funzioni Starbox [49] che permette di creare facilmente diverse rtipologie di votazione utilizzando solo una immagine PNG. La libreria è costruita per utilizzare Prototype, che è la biblioteca standard di estensione javascript in Webratio. Per alcuni effetti aggiuntivi è possibile utlizzare anche la biblioteca Scriptaculous.

*La biblioteca di funzioni Starbox*

### *Descrizione dell'implementazione*

Per implementare questo modello ho creato una unità personalizzata che incapsula la maggior parte della logica di creazione e attivazione del modello.




*Le varie parti che compongono l'unità personalizzata "Valutazioni" (RateItUnit)*

### *Dati utilizzati*

I parametri utilizzati sono un sottoinsieme di quelli disponibili in starbox, quelli essenziali per mantenere semplice l'utilizzo.

#### Parametri statici

Unico parametro statico è il numero di stelle che identifica anche il voto massimo ammissibile. Valore predefinito è 5, e può essere aumentato o diminuito a piacere.

- **Stars:** (intero, default: 5)
  Numero di stelle visualizzate

#### Parametri dinamici

I parametri dinamici sono necessari per visualizzare le votazioni già effettuate.

- **Average**: (float)
  Votazione media corrente da visualizzare
- **Rated:** (boolean):
  Valore sì/no indicante se l'utente corrente ha già votato (e non può quindi effettuare una nuova votazione)
- **Total**: (integer)
  Numero di voti espressi, visualizzato accanto alla votazione media corrente.

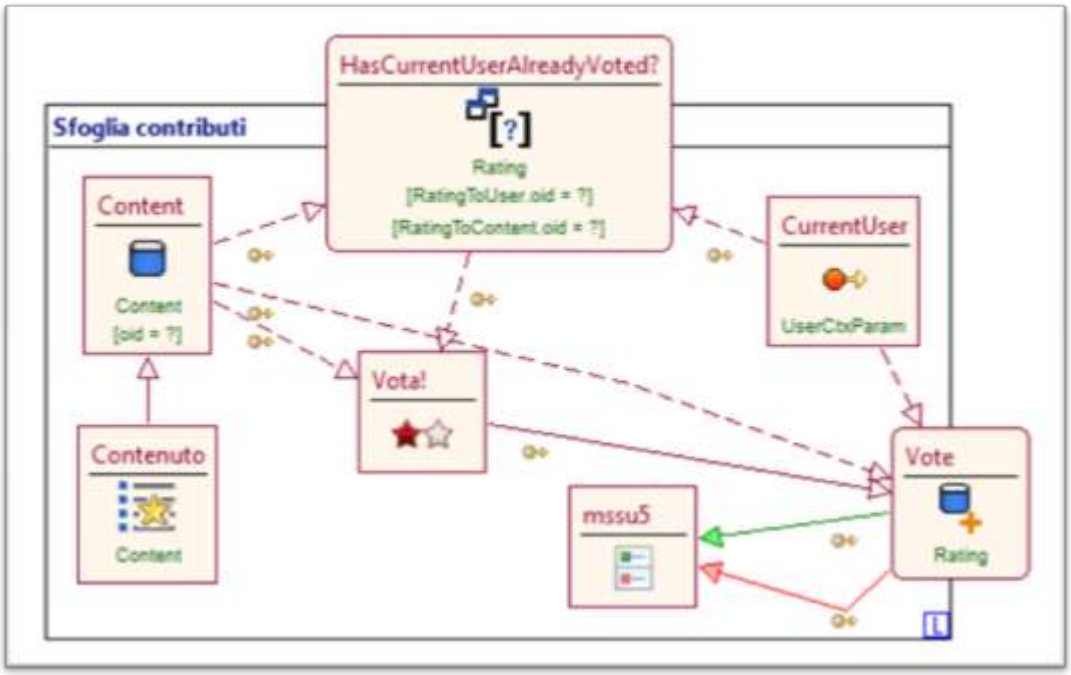
#### Parametri di output

Unico parametro di uscita è il voto espresso dall'utente.

- **Vote** (integer)
  Voto espresso

### *L'implementazione WEBML*

Se l'utente ha diritto può esprimere un voto. Passando con il mouse sulle stelle queste cambiano colore dando un feedback immediato della possibile scelta di votazione.




*In diagramma WebMl contenente l'unità di valutazioni ed il suo scenario di utilizzo*

### *La visualizzazione sulla pagina*

Ho creato una Index Unit che visualizza una lista di contenuti, una Data Unit che visualizza un singolo contenuto su cui l'utente può esprimere la propria preferenza.

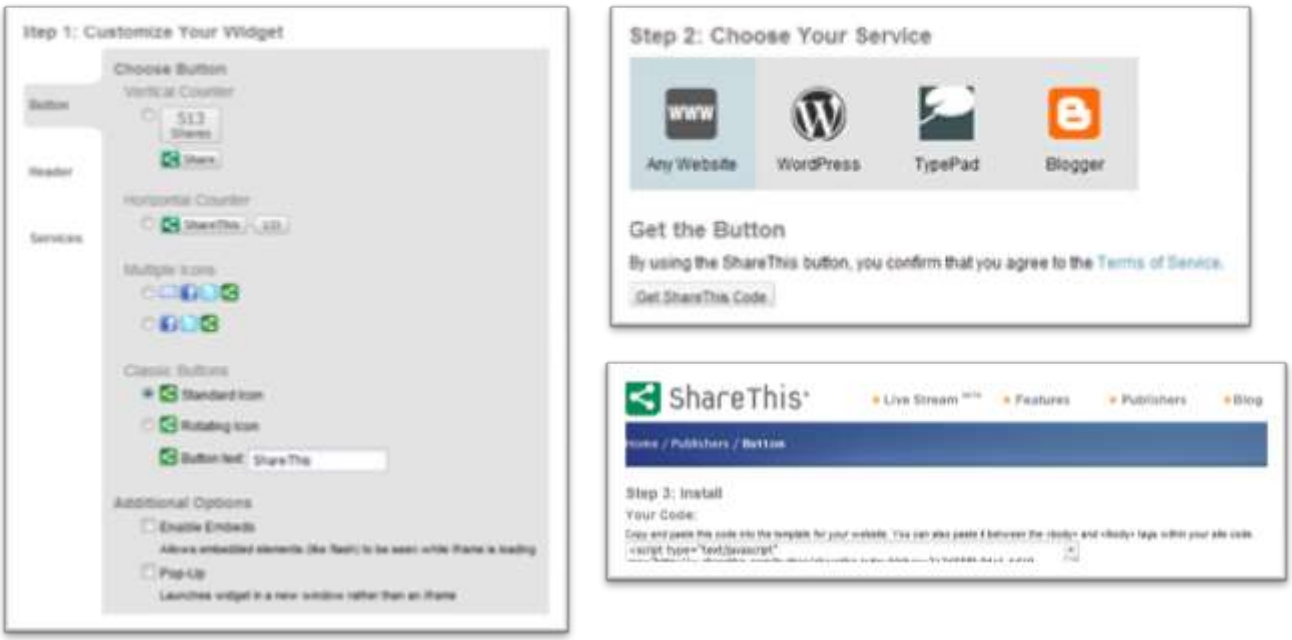
All'attivazione della unità di votazione viene creata una nuova entità di tipo rating combinando l'utente corrente, il contenuto che si sta votando e il voto espresso in un unica preferenza.

*A destra l'elenco contenuti, a sinistra il dettaglio con lo strumento di votazione*

## Modello condividi

L'implementazione del modello condividi è molto semplice perché è possibile incorporare uno dei tanti servizi disponibili che forniscono un collegamento ai network più diffusi.

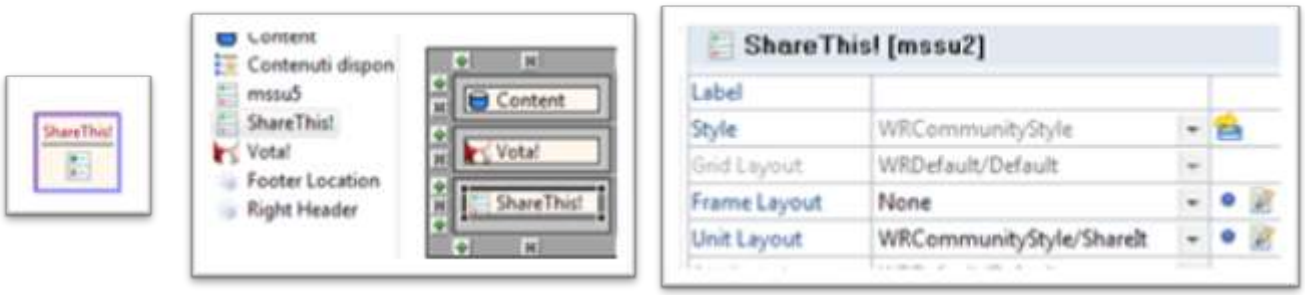
Nel mio caso ho registrato un account sul servizio ShareThis [46].




*I tre semplici passaggi per ottenere il codice da incorporare nel proprio sito*

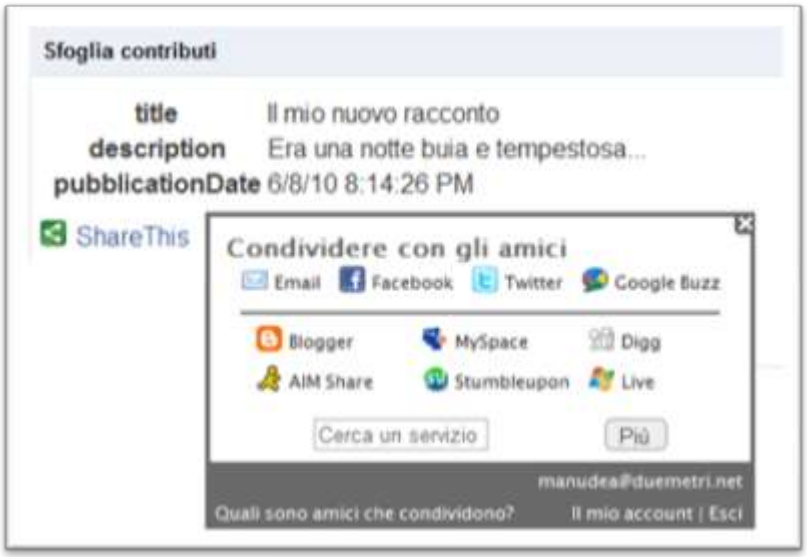
### *L'implementazione WEBML*

Per utilizzare il servizio in webratio è sufficiente create un nuovo template personalizzato per l'unità più semplice che esiste: la Multi Message Unit.




*Come incorporare semplicemente un codice html in webratio*

## *La visualizzazione sulla pagina*




*Il risultato finale, il link ShareThis come appare al click del mouse*

# Relazione tra pattern, modello WebMl e modello dati

| Pattern | Modello WebMl | Modello dati |
|---|---|---|
| Scrivi recensione | Modello valutare un oggetto | Entità **Rating**<br>Title: titolo recensione<br>Description: testo recensione<br>Vote: voto numerico |
| Valutare un oggetto (stelline) | Modello valutare un oggetto | Entità **Rating**<br>Vote: voto numerico (1-5)<br>Entità **Content**:<br>votesAverage: valutazione media |
| Valutare un oggetto (mi piace/non mi piace) | Modello valutare un oggetto | Entità **Rating:**<br>Vote: voto numerico (+1 / -1)<br>Entità **Content**:<br>votesPositive: mi piace<br>votesNegative: non mi piace |
| Assegna Tag | Modello assegna tag | Entità **Tag**<br>TagName: Nome tag<br>Entità **Content** |
| Nuvola di tag | Modello nuvola di tags | Entità **Tag**<br>TagName: Nome tag<br>TagCount: conteggio relativo |
| Geo-Mashing | Modello Geomapping | Entità **Content**:<br>whereLatitude: Latitudine<br>whereLongitude: Longitudine |

# Conclusioni e futuri sviluppi

In questo lavoro ho raccolto diversi pattern ed ho dimostrato come sia possibile implementarli in Webratio in modo che sia semplice realizzare una applicazione web che implementi i pattern sociali.

Nessun progetto software può dirsi mai veramente concluso. Non appena lo si riprende in mano vengono subito alla mente migliorie, aggiunte, prefezionamenti.

Partendo da quanto ho realizzato è possibile estendere il progetto in modo da inglobare nuove unità personalizzate e soprattutto indicare nuovi scenari e modi d'uso per quelle presentate in questa pagine. Ogni pattern ingloba nella sua definizione diverse pratiche che sono emerse nel tempo in vari siti web quindi è facile immaginare come ogni implementazione possa essere più ricca e flessibile di quanto si possa immaginare a prima vista.

## Un progetto open source

Per rilanciare questa idea in modo concreto e mettere quindi le mie conclusioni a disposizione di tutti ho deciso di pubblicare il codice sorgente delle unità che ho creato e la relativa applicazione di esempio che ho realizzato per dimostrarne il funzionamento.

Il progetto è disponibile all'indirizzo: http://code.google.com/p/webratio-community-patterns/

con una sommaria descrizione in lingua inglese, per abbracciare un pubblico più vasto.

La licenza che ho scelto è la licenza MIT, creata ed utilizzata in ambito universitario dal Massachusetts Institute of Technology [53]. E' una licenza estremamente permissiva per incoraggiare la massima diffusione di questo lavoro.

Dalla stessa pagina è possibile accedere all'archivio di controllo codice sorgente direttamente dall'interno dell'ambiente di Webratio usando l'apposito plugin SVN disponibile per Eclipse [58].

Auspico dunque che altri vogliano partire da dove io ho terminato per proporre le proprie idee e i propri casi d'uso con la consapevolezza di non partire da zero.

*Emmanuele De Andreis*

*Luglio 2010*

# Appendice A
# Realizzare unità personalizzate in WebRatio

## Le unità personalizzate (custom unit)

In questa parte del lavoro descriverò alcune implementazioni pratiche dei pattern visti nella sezione precedente. In particolare non descriverò soltanto il risultato finale, ma cercherò di descrivere anche il procedimento per creare nuove unità personalizzate che posso essere utile a chi dopo di me vorrà cimentarsi nella creazione di nuove unità [21].

Sebbene ispirato alla documentazione disponibile cercherò qui di omettere particolari non essenziali, rimandando alla guida per gli approfondimenti, e di approfondire invece quelli che ritengo gli aspetti più importanti, non dando per scontato la conoscenza di Eclipse e di WebRatio in modo da poter fornire informazioni, spero utili, a chi si cimenta per la prima volta nella creazione di una unità personalizzata. Molte delle informazioni che ho raccolto in queste pagine non sono presenti nella documentazione e negli esempi ufficiali, derivano invece da risposte a domande, dubbi e difficoltà riscontrate dagli utilizzatori di WebRatio e discusse nei forums di supporto, molto vivi ed attivi[1].

La documentazione esistente è molto completa da un punto di vista formale, ed elenca con precisione tutte le funzioni disponibili. Non è però facile per un principiante capire tutti i concetti presentati e soprattutto comprendere come legarli tra loro per ottenere il risultato voluto. Questo è un peccato perché le unità personalizzate sono un elemento fondamentale per integrare funzionalità particolari in WebRatio e, una volta padroneggiate le basi, non è così complesso costruire unità personalizzate in poco tempo.

L'idea iniziale di questo scritto è nata nel tentativo personale di cercare di fare chiarezza nella documentazione esistente, spesso troppo dettagliata nel descrivere i singoli componenti da perdere la visione di insieme e l'aspetto pratico. I pochi esempi disponibili rendono a volte difficile adattare la propria idea ad una nuova unità.

In queste pagine cercherò di presentare i concetti che ho sperimentato nella maniera più semplice possibile focalizzandomi più che altro sul risultato desiderato e proponendo uno solo dei molteplici modi per realizzarlo, quello più semplice, quello più immediato. Compresi i passaggi base sarà poi molto semplice approfondire tutte le sfumature e le potenzialità del linguaggio consultando la guida e le risorse disponibili.

---

[1] Tutte le schermate e le procedure presentate sono state realizzate e testate usando la versione 5.1.1 di WebRatio

## Introduzione

WebML, il linguaggio di modellazione di WebRatio, è costruito su pochi elementi altamente componibili, che possono essere utilizzati per assemblare complesse applicazioni Web.

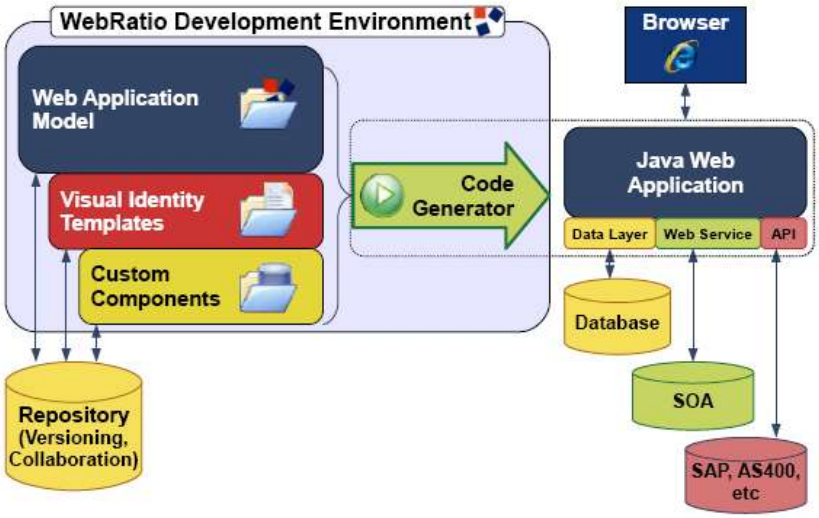
L'aspetto chiave di WebML è la capacità di definire un modello costituito da pagine Web, unità di contenuto, e le unità di gestione, legati tra loro. Tuttavia, l'unità di base fornite in WebML potrebbe non essere sufficiente per coprire l'intero spettro di requisiti applicativi, soprattutto nel caso in cui fosse necessario usare componenti software esterni nello sviluppo delle proprie applicazioni web. A questo scopo, WebRatio comprende il concetto di unità personalizzate (chiamate anche unità plug-in). Un'unità personalizzata (*custom unit*) è un'unità di tipo contenuto (*content unit*) o di tipo operazionale (*operation unit*) definita interamente dallo sviluppatore. E' possibile inserire in un diagramma ipertestuale tutte le unità personalizzate, collegarle ad altre unità e definirne le proprietà; se l'unità è un'unità di contenuto, è anche possibile disporla sulla griglia, e scegliere un modello (*template*) per visualizzare il contenuto. È possibile implementare le unità per fare praticamente qualsiasi cosa, da inviare email a pubblicazione di documenti XML, per interagire con i servizi web.

## Per cominciare

L'architettura generale di WebRatio distingue due componenti principali:

- L'ambiente di sviluppo WebRatio, che supporta la fase di progettazione, sviluppo e generazione dell'applicazione
- WebRatio Runtime, che offre tutte le funzionalità necessarie all'esecuzione di applicazioni Java Web generate da WebRatio

Costruire una unità personalizzata non è una operazione semplicissima perché richiede dunque di padroneggiare diversi concetti che riguardano non soltanto la programmazione java, e soprattutto la programmazione java per il web (javabeans e jsp), ma anche soprattutto una profonda conoscenza di WebRatio e del suo funzionamento interno.

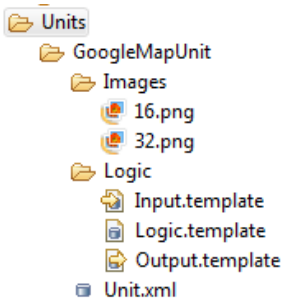



*L'architettura interna di un progetto realizzato con Webratio (figura da [21])*

L'unità personalizzata che creeremo è un po' più complessa del tutorial "Hello World" presente all'interno della guida in linea dell'applicativo alla quale rimando per la descrizione più dettagliata delle operazioni elementari.

Tutte le unità personalizzate che l'utilizzatore di WR può creare devono essere incluse in un progetto di tipo speciale ("Untis Project").

Si parte aprendo Webratio e creano un nuovo progetto. Il tipo di progetto è di tipo "Units Project". All'interno del progetto va creata una nuova Unità (New → Unit) all'interno della cartella Units.




*La struttura dei files di una unità personalizzata all'interno dell'albero di WebRatio*

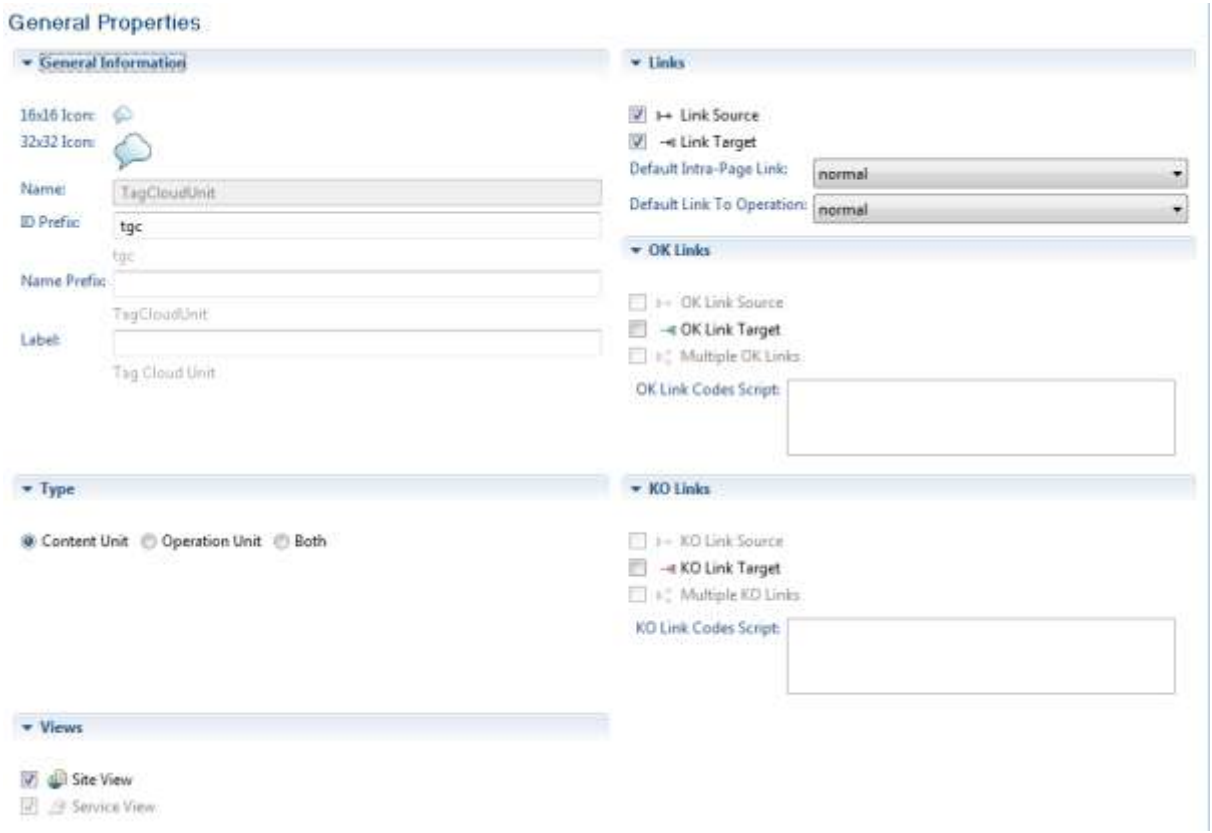
Quando un progetto di tipo Unit è aperto in WR tutte le Unità valide presenti in esso sono automaticamente disponibili in ogni progetto WR ed immediatamente utilizzabili.

La creazione di una nuova Unità inserisce diversi nuovi files nel progetto, ognuno dei quali ha un preciso significato che è importante comprendere per realizzare delle Unità personalizzate potenti ed efficaci.

## Unit.xml

Questo è il file più importante della nostra nuova unità e gestisce tutti i parametri principali. E' un file xml, ma Webratio mette a disposizione una piacevole interfaccia, suddivisa in diverse sottopagine, che ci guida nella scelta delle varie opzioni.

### *Pagina generale (General page)*




*La pagina generale*

La pagina generale ha un duplice scopo: contiene una sezione per impostare le informazioni generali dell'unità come il nome, il tipo, e le viste (Views) dove può essere inserita e specifica se l'unità può essere di origine e di destinazione di uno o più collegamenti.

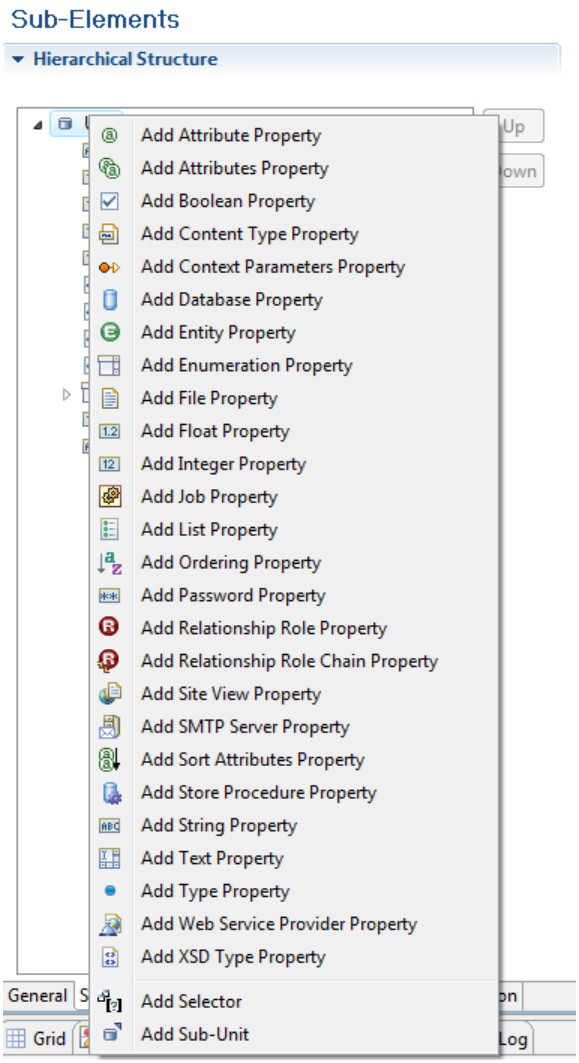
Le immagini che rappresentano l'unità nelle varie visualizzazioni vanno inserite nella cartella "Images" e devono chiamarsi 16.png (formato 16x16) e 32.png (32x32).

### *Pagina sotto elemento (Sub-elements)*

Permette di definire tutte le proprietà che saranno visibili nella finestra delle proprietà dell'unità. Si possono scegliere diverse tipologie di proprietà, dai valori più semplici (testo (string), vero/falso (boolean), numerico intero (integer) o a virgola mobile (float)), quelli che richiamano i moduli HTML (elenco valori (enumeration), password, file) fino a quelli più complessi e tipici di WebRatio

che permettono di interagire con le entità ed il database (entità (entità), campi di entità (attribute), relazioni (relationship) e database).

Da questo punto è anche possibile definire sotto unità (che vanno oltre gli scopi di questa trattazione e che quindi non considereremo, fare riferimento alla guida se interessati).




*Il menu di inserimento proprietà nella pagina dei sottoelementi.*

Va ricordato però che tutti i sottoelementi che sono definiti in questa pagina sono case-sensitive, e fanno quindi distinzione tra maiuscole e minuscole.

In caso contrario si può ottenere un errore del tipo:

```
com.webratio.rtx.RTXException: Error creating service: gmapM2 - WEB-INF/descr/gmapM2.descr
…
Caused by: com.webratio.rtx.RTXException: The attribute 'latitude' is missing for the element
GoogleMapMarkerUnit
     at com.webratio.rtx.core.DescriptorHelper.getAttribute(DescriptorHelper.java:209)
```

E' bene pertanto seguire una convenzione che ci aiuti in seguito a non sbagliare riferimenti (ad esempio decidere che tutte le variabili sono scritte in minuscolo)

I sub-elementi hanno alcune proprietà comuni:

- *Etichetta*: definisce l'etichetta utilizzata per visualizzare le proprietà nella visualizzazione delle proprietà.
- *Ordine in figura*: Se presente permette di visualizzare l'attributo sotto l'icona nel modello WR.

Oltre alle proprietà comuni ogni elemento possiede delle sue proprietà che sono per lo più intuitive (per un elenco completo consultare la guida in linea di WebRatio).

### *Logica (Logic)*

Permette la definizione di script logica. Gli script sono significativi solo per le unità di contenuto, perché gli script permettono di influenzare l'ordine in cui vengono computate le unità sulla pagina e di definire le dipendenze con le altre unità.

Il metodo *computeParameterValue* viene invocato dal runtime di WebRatio in base a come i parametri devono essere propagati all'interno della pagina. La logica con cui i parametri devono essere propagati dipende dal tipo di unità: l'unità stessa dice, attraverso uno dei suoi script di configurazione, di che tipo è:

- *Context-free* (l'unità non ha bisogno di nessun parametro per potersi computare)
- *History* (l'unità mantiene la storia delle scelte fatte dall'utente).
- *Multicondition* (l'unità ha una o più condizioni che le servono per potersi computare)

Se l'unità non ha definito di che tipo è, il runtime non prova nemmeno a propagarne i parametri.

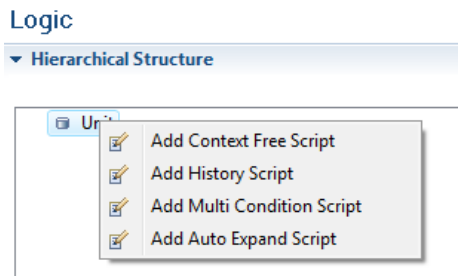
Bisogna aprire il file di configurazione, andare nel tab Logic e aggiungere gli script per definirne il tipo (Context Free, History, Multicondition). Come prima prova è sufficiente aggiungere lo script Context Free e scrivere come valore semplicemente

```
return true
```

A questo punto, dopo aver rigenerato il runtime dovrebbe invocare il metodo computeParameterValue.

Altro valore comune è:

```
return getIncomingLinksAllTypes(unit).empty
```

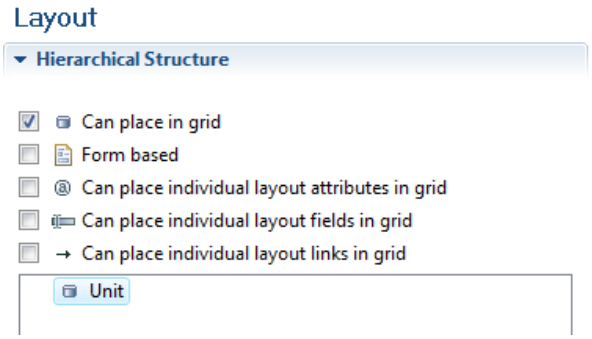



*La pagina logica*

E' un argomento abbastanza avanzato per cui rimando alla guida in linea per approfondimenti.

### *Disposizione (Layout)*

Le opzioni in questa pagina permettono di definire il comportamento degli elementi dell'unità ed ha senso solo per unità di contenuto.




*La pagina layout*

Le opzioni più importanti qui sono:

- Può essere posizionata sulla griglia (*Can place in grid?*) che decide se l'unità può essere posizionata usando la griglia.
- Basata su modulo (Form based) che regola la possibilità o meno di accettare input da parte dell'utente.

Altre opzioni sono disponibili (consultare la guida in linea a proposito).

### *Documentazione (Documentation)*

Permette di documentare tutte le funzioni che abbiamo deciso di aggiungere alla nostra unità e spiegare l'utilizzo previsto.

### *Versione (Version)*

Permette di inserire alcune informazioni relative alla versione della unità che stiamo sviluppando.

## La cartella logic

L'input e l'output dinamico prodotto dale istanze delle unità richiede una coppia di templates scritti in Groovy chiamati Input.template e Output.template memorizzati nella sotto cartella Logic della cartella della unità. Lo scopo di questi templates è quello di produrre due frammenti XML che contengono rispettivamente gli elementi <InputParameters> e <OutputParameters>.

### *Il file Input.template*

Il file Input.template è situato nella sottocartella Logic della cartella dell'Unità e specifica i parametri di input.

Ogni parametro di input ha un nome (name), un tipo (tipo) e una etichetta (label).
L'attributo tipo è molto importante perché, se usato correttamente, permette di abilitare la funzione di agganciamento automatico (automatic coupling) quando si passano le informazioni su un link in entrata all'unità personalizzata. La logica utilizzata nell'agganciamento automatico è tale che l'attributo tipo del parametro di ingresso della fonte dell'unità di collegamento e il parametro di output del target unità di collegamento abbiano lo stesso tipo di dato, il nome non viene controllato anche se a volte sarebbe utile (se ci sono due parametri con lo stesso tipo di dato vengono collegati automaticamente entrambi al primo dei due anziché ognuno al suo corrispondente).

```
#?delimiters <%,%>,<%=,%>
<%
     setXMLOutput()
     def unitId = unit.valueOf("@id")
%>

<InputParameters>
     <InputParameter name="<%= unitId %>.zoom" type="integer" label="Zoom"/>
     <InputParameter name="<%= unitId %>.latitude" type="double" label="Latitude"/>
     <InputParameter name="<%= unitId %>.longitude" type="double" label="Longitude"/>
</InputParameters>
```

### *Il file Output.template*

Il file di output è assolutamente speculare al file di input, valgono le stesse considerazioni già fatte, cambia solo il nome.

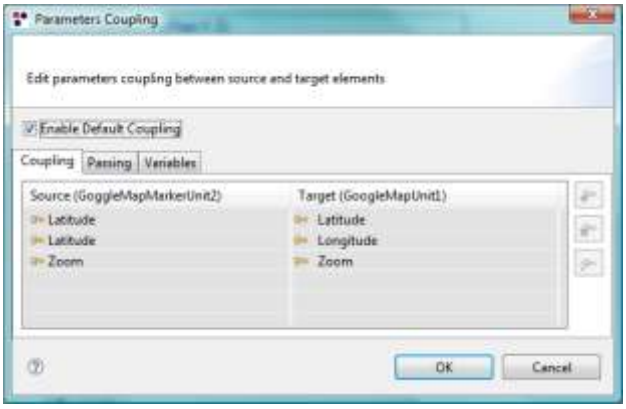
Se il bean ha un metodo corrispondente (es: Zoom ha come corrispondenza un getZoom() nel bean) allora il type che avevamo messo in input nell'output non serve.

In generale solamente gli input parameter name vanno qualificati con l'id della unit per evitare collisione di nomi nella URL.

A titolo di esempio possiamo definire in un'unità il seguente output, che combinerà perfettamente con l'input dell'esempio precedente, risultato che vediamo nella successiva figura.

```
#?delimiters <%,%>,<%=,%>
<%
     setXMLOutput()
     def unitId = unit.valueOf("@id")
%>

<OutputParameters>
     <OutputParameter name="<%= unitId %>.zoom" label="Zoom"/>
     <OutputParameter name="<%= unitId %>.latitude" label="Latitude"/>
     <OutputParameter name="<%= unitId %>.longitude" label="Longitude"/>
</OutputParameters>
```




*Il risultato dell'accoppiamento automatico in WebRatio dopo aver definito parametri compatibili.*

## *Il file Logic.template*

Il file Logic.template, è situato nella sottocartella Logic della cartella dell'Unità e specifica le proprietà di lavoro.

Il file contiene un elemento <Descriptor> che descrive il comportamento e le diverse proprietà aspetto. Gli elementi figlio di <Descriptor> specificano le proprietà che saranno disponibili nel quadro dell'unità di proprietà.

Tipicamente le proprietà sono già state definite nella pagina dei sottoelementi del file unit.xml, il seguente codice (che viene creato automaticamente quando creiamo la nostra nuova unità) non fa altro che serializzare le proprietà definite, quindi non abbiamo bisogno di scrivere altro.

```
#?delimiters <%,%>,<%=,%>
<Descriptor service="com.webratio.units.custom.googlemapunit.GoogleMapUnitService">
     <% printRaw(serializeXML(unit, false)) %>
</Descriptor>
```

E' importante notare qui il nome della classe java presente nell'attributo service dell'elemento Descriptor, che vedremo nel paragrafo successivo.

Nel caso fosse necessario è possibile aggiungere elementi personalizzati o con una struttura diversa (vedere la documentazione se si vuole seguire questa strada).

## *Il file WebModel.template*

Il file WebModel.template, è situato nella sottocartella Logic della cartella dell'Unità e permette allo sviluppatore di controllare delle condizioni di errore ed avvisare tramite errori o warning in fase di compilazione che l'utilizzo della unità non è corretto o mancano parametri. La definizione di questo file non è obbligatoria, ma può essere molto utile se avete intenzione di sviluppare un'unità personalizzata complessa.

## Il codice Java

La creazione di una nuova unità richiede anche la creazione del servizio java che gestisce la logica del componente. In particolare durante il processo di creazione della nuova unità viene aggiunto nel package di default (com.webratio.units.custom.nomeunit se non specificato diversamente). La classe referenziata nel file Logic.template viene creata automaticamente nella directory "src" del progetto Units.

Non tutti i files sono visibili usando la perspective predefinita, per vedere i file java di codice occorre aprire il "Package Explorer" scegliendolo dal menu "Window → Show Wiew". Così facendo sono visibili anche i files della cartella "src" (che contiene il codice sorgente java e javabean).

### *Il codice che contiene lo stato (Unit Java bean)*

Prima di definire il nostro service però è consigliabile creare una classe Java Bean che useremo per gestire tutte le proprietà associate allo stato della nostra unità.

Nel caso di esempio, per semplicità, immaginiamo di avere una sola proprietà di tipo numerico intero chiamata zoom.

Nello stesso package del service creiamo una classe java che ha lo stesso nome della nostra unità, ma con suffisso Bean anziché Service che estende la classe base BasicContentUnitBean.

La classe BasicContentUnitBean è una classe da usare con le unità di tipo contenuto direttamente, oppure estendendola in modo opportuno. Ha due proprietà:

- **dataSize** (intero) corrispondente alla dimensione dei dati presenti.
- **data** (object) I dati da visualizzare nella unità.

In questa classe aggiungiamo poi una variabile di tipo intero per memorizzare il nostro dato e definiamo i due metodi di get e set.

```
public class GoogleMapUnitBean extends BasicContentUnitBean {

     private int zoom;

     public int getZoom() {
            return zoom;
     }

     public void setZoom(int zoom) {
            this.zoom = zoom;
     }
}
```

### *Il codice che definisce la logica (Unit Java Service)*

Ora possiamo tornare al nostro service.

Nel costruttore della classe leggiamo i valori dal descriptor.

Il descriptor che andremo a leggere è simile a quello seguente:

```
<GoogleMapUnit xmlns:gr="http://www.webratio.com/2006/WebML/Graph" gr:x="0" gr:y="0" id="gmap1"
name="GoogleMapUnit1" zoomFactor="5" customDescriptor="false"/>
```

Si può visualizzare l'XML selezionando un'unità all'interno del modello WR e cliccando sul bottone *'Show XML'* nel pannello XML delle proprietà dell'unità.

Tutti i descrittori xml sono rilasciati insieme all'applicazione generata, nella sottocartella *WEB-INF/descr* nei rispettivi file *<id della unit>.descr*

### *Lettura dei parametri statici*

I parametri statici sono definiti in fase di realizzazione del progetto e sono inseriti direttamente all'interno dell'unità usando la finestra di proprietà.

La lettura dei parametri statici avviene tipicamente nel costruttore della classe service dell'unità.

Per accedere a questi valori per prima cosa accediamo all'elemento base dell'unità (quello che è stato creato dall'istruzione *printRaw(serializeXML(unit, false))* nel *Logic.template*) che si trova nel *descriptor*, passato alla classe service nella variabile *descr*. Per leggere il *descriptor* usiamo le classi di lettura file xml presenti nel package *org.dom4j.Element*

```
    Element currentUnit = descr.element("GoogleMapUnit");
```

Una volta catturato l'elemento possiamo accedere all'attributo e memorizzarlo in una variabile privata della classe, come nel seguente esempio:

```
    Object zoomFactor = DescriptorHelper.getAttribute(currentUnit, "zoomFactor", true, this);
```

### *Lettura dei parametri dinamici*

I parametri dinamici non sono noti al momento della progettazione del modello, ma sono forniti all'esecuzione da altre unità collegate, che a loro volta possono averli caricati da database, o catturando l'input dell'utente.

Il modello delle unità personalizzate sgancia il comportamento dell'unità dalla sua sorgente dati usando un meccanismo di accoppiamento (*binding*), l'unica cosa che veramente importante è che il tipo dei dati fornito ed atteso corrisponda. In questo modo qualunque unità, almeno in teoria, può agganciarsi a qualunque altra che fornisca dati che lei sia in grado di gestire.

## Parametri d'ingresso

Ogni unità può essere sorgente dati (o *source*), come definito nel file *Output.template*, oppure ricevere dati (o *target*), come definito nel file di *Input.template*.

Per recuperare i dati passati in ingresso è possibile utilizzare, nella classe service, il metodo *asString* dell'oggetto *BeanHelper*, il quale prende in input il nome dell'Input Parameter in questione, recuperato dall'operationContext.

Quindi se per esempio il parametro di ingresso è stato definito in questa forma

```
<InputParameter name="<%= unitId%>.uri" type="string" label="URI"/>
```

allora si può recuperare il valore del parametro in questo modo:

```
String paramString = BeanHelper.asString(operationContext.get(getId() + ".uri"));
```

Il metodo BeanHelper.asString (obj) ritorna il valore stringa di un oggetto. Se l'oggetto di input è una stringa, viene restituito. Se l'oggetto in ingresso è un vettore di stringhe, il primo elemento (se esiste) viene restituito. Se l'oggetto in ingresso è un vettore di oggetti, il metodo *toString()* viene richiamato sul primo elemento (se presente). Infine, il metodo *toString()* viene richiamato sopra l'oggetto d'ingresso. Un valore null viene restituito se l'oggetto di input è null.

Una IndexUnit ritorna un oggetto singolo, se si è interessati ad avere un array di oggetti bisogna usare una MultiDataUnit.

Alternativamente si può usare il metodo *BeanHelper.asStringArray (obj)* che ritorna un vettore di stringhe da un oggetto. Se l'oggetto in ingresso è un vettore di stringhe, è restituito un clone. Se l'oggetto in ingresso è una stringa, un vettore di stringhe composto da un solo elemento è costruito e ritornato. Se l'oggetto di input è un vettore di oggetti, il metodo toString () viene richiamato per di ogni elemento per costruire un vettore di stringhe.

```
String[] names = BeanHelper.asStringArray(input)
```

La lettura dei parametri d'ingresso normalmente avviene nel metodo *execute* della classe service dell'unità.

### Parametri di uscita

La lettura dei parametri di uscita avviene nel metodo *computeParameterValue (String outputParamName, Map pageContext, Map sessionContext)* della classe service.

Il metodo è chiamato quando le unità collegate hanno necessità di recuperare un valore dall'unità e la stessa ha dichiarato in che modo chiamarlo negli script Logic di Unit.xml, se non è stato dichiarato nulla nella logica, il metodo non sarà chiamato.

Il nome del parametro richiesto è presente come stringa in *outputParamName*, con prefisso dipendente dall'unità, che si può sempre leggere utilizzando la funzione *getId()*.

Il metodo può essere chiamato più volte, tante quante sono i parametri collegati.

Ponendo per esempio che il parametro di output abbia questa forma

```
<OutputParameter name="vote" label="Vote"/>
```

Si controlla che il nome corrisponda e si ritorna il valore corrispondente o null se non c'è corrispondenza con i parametri noti, in questo modo:

```
public Object computeParameterValue(String outputParamName, Map pageContext, Map sessionCon-
text)
     throws RTXException {
     if(outputParamName.equals(getId() + ".resultObject")) return resultObject;
     return null;
 }
```

Nel codice di esempio abbiamo ipotizzato che esistesse una variabile di classe *resultObject* e che tale variabile contenesse il valore associato al parametro di output omonimo.

Il metodo computeParameterValue(...) viene comunque invocato solo durante la propagazione automatica di pagina (link automatici o transport) . Per link normal entra in gioco la reflection java:

se ad esempio l'Output.Template dichiara il parametro di output dal name="vote" dal bean della unit viene estratta come conseguenza la proprietà "vote", cioè viene invocato il metodo "getVote()".

### *Esecuzione*

All'esecuzione del metodo *execute*, che è chiamato quando l'unità viene posizionata sulla pagina, possiamo istanziare la classe *bean* che abbiamo appena creato (che contiene lo stato dell'unità) e valorizzarla con i valori inseriti.

```
public class GoogleMapUnitService extends AbstractService implements
            RTXContentUnitService {

     private int zoomFactor;

     public GoogleMapUnitService(String id, RTXManager mgr, Element descr) throws RTXException
{
            super(id, mgr, descr);

            // Selezioniamo l'elemento che contiene i valori dell'unità corrente
            Element currentUnit = descr.element("GoogleMapUnit");

            // valori predefiniti
            zoomFactor = 5;

            try {
                   zoomFactor = Integer.parseInt(DescriptorHelper.getAttribute(
                                  currentUnit, "zoomFactor", true, this));
            } catch (NumberFormatException ex) { }
     }

     public Object execute(Map pageContext, Map sessionContext)
                   throws RTXException {

            GoogleMapUnitBean map = new GoogleMapUnitBean();
            map.setZoom(zoomFactor);
            return map;
     }
```

## La view: Lo stile, ovvero gestire l'output (Layout template)

Lo stile di una unità gestisce il modo con cui l'html viene posizionato sulla pagina e la struttura della relativa jsp.

Gli stili stanno in un progetto di tipo particolare ("Style project"). Se non avete un progetto di tipo stile nella vostra soluzione, createne uno.

Ora dobbiamo creare uno stile per la nostra unità, scegliendo Nuovo → "Layout template" nella cartella Units/Nome unità (se la cartella non esiste, createla).

Quello che viene inserito in questo file verrà renderizzato ed inserito nel file html risultante.

Esiste una sintassi speciale per accedere ai valori definiti nelle classi java che abbiamo creato in precedenza. In particolare per accedere al valore di zoom definito nella classe bean useremo la sintassi:

```
<c:out value="${<wr:UnitId/>.zoom}"/>
```

Questo esempio è una applicazione dalla JavaServer Pages Standard Tag Library, una biblioteca standard id funzioni java sotto forma di tag che semplificano lo sviluppo di pagine web [56].

JavaServer Pages Standard Tag Library (JSTL) incapsula le funzionalità di base comuni a molte applicazioni JSP, come semplici tag. JSTL ha il supporto per attività comuni quali l'iterazione e istruzioni condizionali, tag per la manipolazione di documenti XML, internazionalizzazione e tag di formattazione di impostazioni locali, tag SQL e strutturali. Introduce anche un nuovo linguaggio di espressione per semplificare lo sviluppo di pagine, e fornisce un'API per gli sviluppatori per lo sviluppo di tag personalizzati conformi alle convenzioni JSTL.

Nell'esempio qui proposto il valore di zoom viene renderizzato in un elemento HTML di tipo <p>

```
#?delimiters [%, %], [%=, %]
[%setHTMLOutput()%]
<p><c:out value="${<wr:UnitId/>.zoom}"/></p>
```

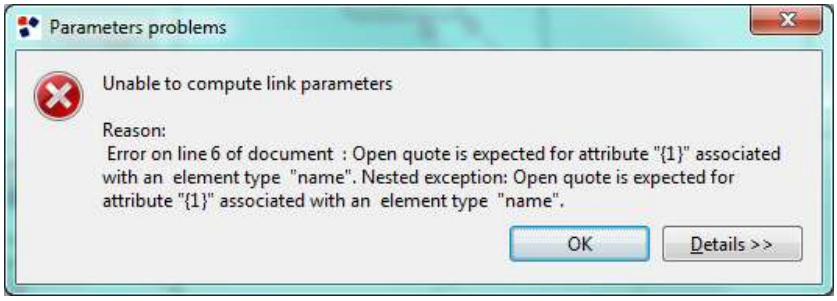
E' possibile definire in una custom unit un layout predefinito in modo che sia automaticamente selezionato ogni volta che l'unità viene posizionata sulla pagina senza richiedere all'utilizzatore di sceglierlo manualmente ogni volta: nel progetto di stile associato al tuo modello c'è un file xml, che si chiama Layout.xml, che permette per ogni content unit di definire i template da applicare di default (unit, frame, link, attribute/field) e per ognuno di essi i layout parameters.

Per fare in modo che le nuove unit prendano il layout predefinito bisogna fare la stessa cosa anche nelle proprietà progetto webratio.

## Quando qualcosa va storto

Non è sempre semplice capire cosa non va quando si verificano degli errori.

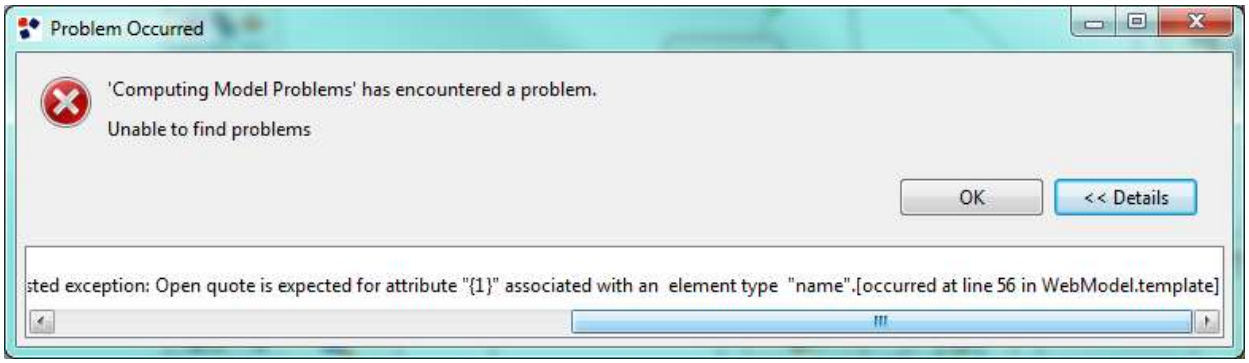
Un errore possibile che può avvenire quando si tenta di definire l'accoppiamento dei parameri è il seguente, ed indica un errore nel file input.template dell'unità personalizzata (nel nostro coso dipendenti dal troppo zelo di word nel convertire le virgolette semplici usate in programmazione con le virgolette tipografiche, più eleganti ma incomprensibili per il parser).




*Errore nella accoppiamento dei parametri.*

Lo stesso errore può essere ancora più misterioso se occorre quando si cerca di avviare una validazione del modello con "Find Model Problems".

L'errore fa riferimento al Model.Template, che non c'entra nulla…




*Errore in un file xml, non sempre il messaggio di errore riporta il file corretto…*

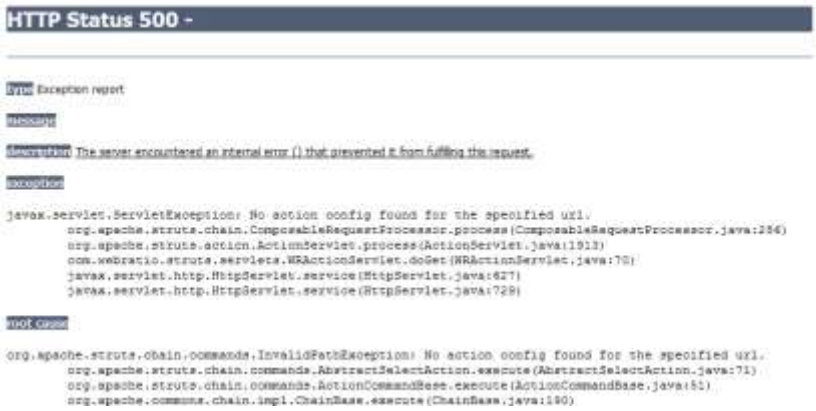
Un errore possibile cercando di aprire una pagina generata è che la pagina stessa non venga trovata.

*Errore http 404: pagina non trovata.*

Se la pagina non è stata cancellata dal modello è probabile che tomcat non sia in esecuzione o comunque non sia in grado di rispondere correttamente, a volte capita quando si modificano le

unità, richiedendo quindi un continuo ricaricamento del contesto. La prima cosa da fare è provare a rigenerare il progetto completo (*"Generate full web project"*) e riavviare tomcat.

Un altro errore piuttosto comune è: *"No action config found for the specified url"*, significa che la pagina richiesta non è stata generata.

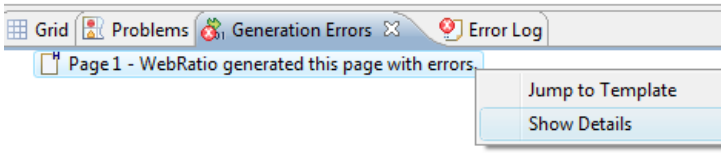



*No action config found for the specified url.*

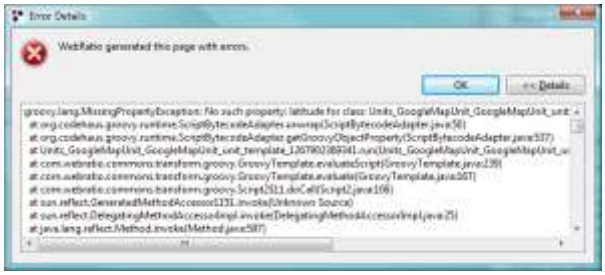
Altre volte WR riesce ad essere più esplicito e preciso. Se ci sono errori nella generazione l'applicazione non viene nemmeno avviata.

*Errori durante la generazione*

Per avere lumi sugli errori di generazione bisogna dare un'occhiata agli errori di generazione che troviamo nell'ambiente di WR (oppure visualizzare il sorgente della pagina generata che contiene il dettaglio dello stack con tutto il percorso)

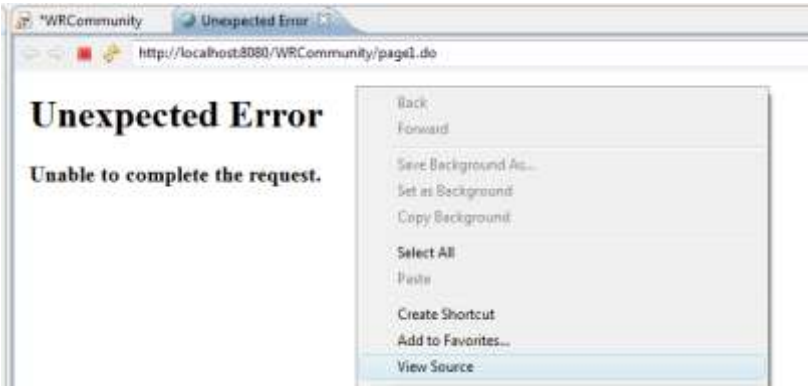



*Per saperne di più possiamo scegliere "Show Details"*




*Il dettaglio dell'errore con l'intero stack trace*

Quando invece si verifica un errore sulla pagina web è possibile dare una occhiata al codice, che contiene il messaggio di errore completo, commentato e reso invisibile (il suggerimento sembrerà banale ma l'ho trovato scavando nei gruppi di supporto, non ci avevo pensato).




*Un errore inaspettato*

*Così è più chiaro, avendo a disposizione l'intero stack trace si capisce cosa non va…*

Nel caso di un errore di generazione che è possibile imputare ai layout per trovare la vera causa dell'errore può essere utile usare la funzione "Find layout problems", in generale, se le unit sono state fatte bene, con le validazioni necessarie, è facile capire il dato mancante che ha causato l'errore di generazione.

# Debug

Può essere estremamente complicato realizzare una unità personalizzata senza avere la possibilità di fermare ed analizzare l'esecuzione del codice. Fortunatamente il procedimento è semplice e richiede solo alcuni passaggi.

## *Attivazione tomcat in modalità debug*

Bisogna spiegare a tomcat che deve attivare il debug. Per fare questo bisogna aprire il file catalina.bat (che si trova nella directory di Tomcat: C:\WebRatio\[Webratio DIR]\Tomcat\bin) ed aggiungere la riga seguente all'inizio del file (subito sotto quella esistente che inizia nello stesso modo: set CATALINA_OPTS=)

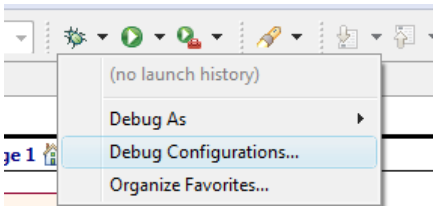
```
set CATALINA_OPTS=-Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=8088
```

Bisogna poi far ripartire tomcat. All'inizio apparirà una riga del tipo:

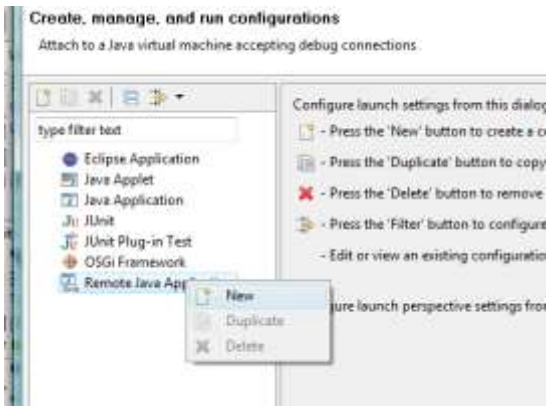
*Tomcat in debug mode*

## *Attivazione WebRatio in modalità debug*

Selezionare in WebRatio il progetto contenente le unit su cui eseguire debug e scegliere "debug configurations…" dal menu debug e creare una nuova configurazione di debug per una "Remote Java Application".

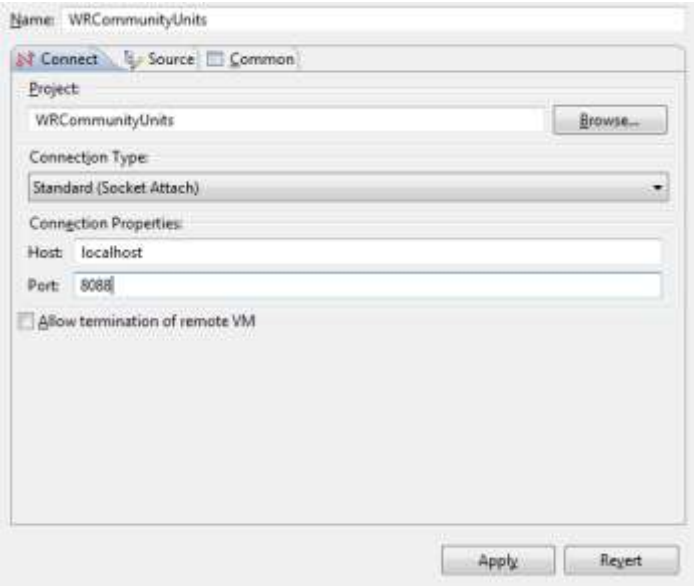



*Debug configurations…*



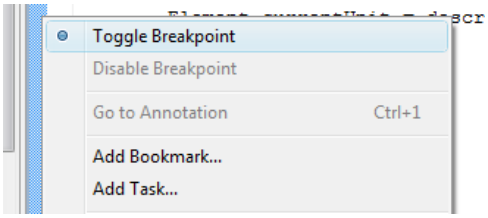

*… new Remote Java Application*

La porta di debug è la 8088, modalità "Standard (Socket Attach)"



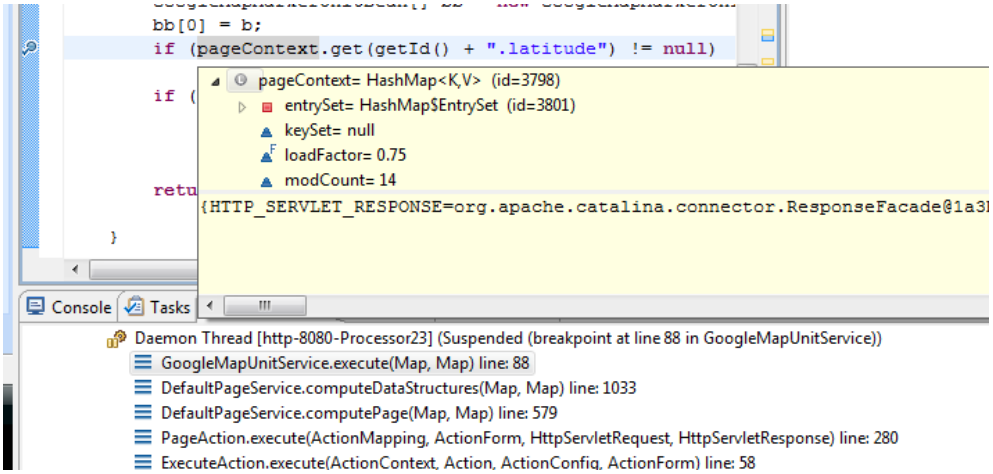

*Standard Debug on port 8088*

A questo punto basta impostare un breakpoint nel punto dove si desidera che l'esecuzione si interrompa.




*Impostazione breakpoint*

Ora basta generare il progetto e visualizzare una pagina. L'esecuzione si interromperà nel punto desiderato. Per una gestione migliore è consigliabile attivare le debug perspective di eclipse, opzione per altro consigliata dallo stesso ide alla prima esecuzione.




*Il debugger in azione*

A volte il debugger sembra non funzionare. Per testare se il debugger funziona si può aprire una finestra comando e provare a vedere se con telnet funziona:

```
C:\ >telnet localhost 8088
Connecting To localhost...
Could not open connection to the host, on port 8088: Connect failed
```

Se la connessione non funziona la cosa più semplice è provare a riavviare tomcat, a volte basta.

## Problemi comuni, soluzioni comuni

### *Evitare una doppia chiamata al metodo execute*

Il numero di volte in cui una unità viene calcolata dipende dall'algoritmo di propagazione. L'algoritmo di propagazione è un algoritmo "compile time" che opera sul grafo delle unit che compongono una pagina che dipende strettamente dalla struttura della pagina e definisce l'ordine con cui i parametri si devono propagare e le unit si devono calcolare. L'algoritmo di propagazione permette di stabilire la logica con cui verrà propagato il contesto all'interno di una pagina nel momento in cui viene invocata. Una unit può comparire più di una volta nell'elenco di propagazione ed è quindi il codice dell'unità stessa che "decide" se ricalcolarsi oppure no in base al fatto che l'unità si sia già calcolata almeno una volta e quindi il suo bean sia già presente nel contesto di pagina. È l'algoritmo di propagazione che decide quante volte una unit deve essere computata.

Detto questo, tutte le unit standard implementano nel loro servizio Java una logica per cui, se il bean dell'unità è già presente nel contesto di pagina, viene restituito direttamente questo bean senza eseguire nuovamente tutto il servizio della unit. Se nelle vostre unit custom, manca l'implementazione di questo meccanismo è possibile che il bean venga processato due volte.

Potreste implementare questo meccanismo facendo in modo che il metodo execute() dell'unità non contenga tutto il codice necessario per calcolare il bean, ma richiami una funzione protetta che determina se calcolare il bean o ritornare quello già memorizzato nel page context. Questa

funzione protetta verrà anche utilizzata dal metodo computeParameterValue, che viene invocato ogni volta che l'unità deve propagare del contesto ad altre unit presente nella pagina.

Qui di seguito un esempio di codice

```
    public Object computeParameterValue(String outputParamName, Map pageContext, Map session-
Context)
                        throws RTXException {
        Object unitBean = getUnitBean(pageContext, sessionContext);
        if (unitBean == null) {
            return null;
        }
      ....
   }
```

```
    public Object execute(Map pageContext, Map sessionContext) throws
RTXException {
        return getUnitBean(pageContext, sessionContext);
    }
```

```
    protected Object getUnitBean(Map pageContext, Map sessionContext) throws RTXException {
        Object unitBean = pageContext.get('_' + getId());
        if (unitBean == null) {
            unitBean = createUnitBean(pageContext, sessionContext);
        } else {
            int dataSize = ((Integer) BeanHelper.getBeanProperty
                            (unitBean, "dataSize", this)).intValue();
            if (dataSize == 0) {
                unitBean = createUnitBean(pageContext, sessionContext);
            }
        }
        pageContext.put('_' + getId(), unitBean);
        return unitBean;
    }
```

Come si può notare, è il metodo createUnitBean che conterrà la logica per calcolare il bean dell'unità ex novo.

```
protected Object createUnitBean(Map pageContext, Map sessionContext)
throws RTXException {
   ........................
   BasicContentUnitBean bean = new BasicContentUnitBean();
   return bean;
}
```

Utilizzando questo meccanismo, la prima volta che viene computata la unit, effettivamente viene invocato il metodo createUnitBean che calcola il bean della unit. Le volte successive che viene richiesto il calcolo dell'unità dall'algoritmo di propagazione, non verrà eseguito questo metodo, ma verrà estratto il bean direttamente dal contesto di pagina.

# Riferimenti

## Bibliografia


[1] Rittel, Horst, and Melvin Webber; "**Dilemmas in a General Theory of Planning**" pp. 155–169, Policy Sciences, Vol. 4, Elsevier Scientific Publishing Company, Inc., Amsterdam, 1973.

[2] C. Alexander, S. Ishikawa, M. Silverstein: **A Pattern Language: Towns, Buildings, Construction**. Oxford University Press (1977)

[3] A. Bellucci, S. Levialdi, A. Malizia: **Visual Tagging Through Social Collaboration: A Concept Paper**. In: Heidelberg, S.B. (ed.): INTERACT, Vol. 4663/2008, Rio de Janeiro (2007) pp. 268-271

[4] C. Crumlish, E. Malone: **Designing Social Interfaces: Principles, Patterns, and Practices for Improving the User Experience**. O'Reilly (2009)

[5] S. Farrell, T. Lau, S. Nusser: **Building Communities with People-Tags**. In: Heidelberg, S.B. (ed.): INTERACT, Vol. 4663/2008, Rio de Janeiro (2007) pp. 357-360

[6] E. Gamma, R.Helm, R. Johnson, J. Vlissides: **Design Patterns: Elements of Reusable Object-Oriented Software**. Addyson Wesley (1998)

[7] N. Koch, M. Pigerl, G. Zhang, T. Morozova: **Patterns for the Model-Based Development of RIAs** (2009)

[8] T. Oreilly: **What is Web 2.0? Design Patterns and Business Models for the Next Generation of Software**. (2007) - (Versione italiana: http://www.bitmama.it/articles/14-Cos-Web-2-0 )

[9] M. V. Welie, H. Traetteberg: **Interaction Patterns in User Interfaces. 7th. Pattern Languages of Programs Conference**, Illinois, USA (2000)

[10] P. Fraternali, M. Tisi, M. Silva, L. Frattini: **Building community-based web applications with a model-driven approach and design patterns**, Politecnico di Milano

[11] H. Rheingold: **The Virtual Community: Homesteading on the Electronic Frontier**, London: MIT Press. (ISBN 0262681218) (2000)

[12] A. Bozzon, T. Iofciu, W. Nejdl, S. Tonnies: **Integrating databases, search engines and Web applications: a model-driven approach**
(Code at : http://skydev.l3s.uni-hannover.de/gf/project/yasef/)

[13] L. Frattini, M. Silva: **Methodology and patterns for developing community based web applications with a model driven approach** (2007)

[14] A. Bozzon: **Model-driven development of Search Based Web Applications** (2009)

[15] J. C. Preciado, M. Linaje, S. Comai, F. Sánchez-Figueroa: **Designing Rich Internet Applications with Web Engineering Methodologies**

[16] A. Bozzon, M. Brambilla, P. Fraternali: **Conceptual Modeling of Multimedia Search Applications Using Rich Process Models**, Politecnico di Milano

[17] D. Sinnig, P. Forbrig, A. Seffah: **Patterns in Model-Based Development**

[18] G. Avagliano, S. Di Martino, F. Ferrucci, L. Paolino, M. Sebillo, G. Tortora, G. Vitiello: **Embedding Google Maps APIs into WebRatio for the Automatic Generation of Web GIS Applications**

[19] S. Ceri, P. Fraternali, M. Brambilla, A. Bongio, S. Comai, M. Matera: **Designing Data-Intensive Web Applications.** Morgan Kaufmann (2002).

[20] J. Tidwell - **Designing Interfaces Patterns For Effective Interaction Design** - O'Reilly (2005)

# Linkografia

Last visit: March 2010


[21] Definizione di **Wicked problem**
http://en.wikipedia.org/wiki/Wicked_problem

[22] Definizione di **Comunità virtuale**
http://it.wikipedia.org/wiki/Comunit%C3%A0_virtuale
http://en.wikipedia.org/wiki/Virtual_community

[23] Definizione di **Servizio di social network**
http://en.wikipedia.org/wiki/Social_network_service
http://it.wikipedia.org/wiki/Servizio_di_rete_sociale

[24] La **lista delle comunità virtuali** più attive
http://en.wikipedia.org/wiki/List_of_social_networking_websites

[25] **The Open Source Way, Creating and nurturing communities of contributors** Red Hat Community Architecture team 2009
http://www.theopensourceway.org/book/

[26] **Design patterns**
http://www.deyalexander.com.au/resources/uxd/design-patterns.html

[27] **UI Patterns library**: http://ui-patterns.com

[28] **Yahoo design pattern library**: http://developer.yahoo.com/ypatterns

[29] **Definizione di tassonomia** http://it.wikipedia.org/wiki/Tassonomia

[30] **UI Patterns library - Tag an Object**:
http://developer.yahoo.com/ypatterns/social/objects/collecting/tag/object.html

[31] **UI Patterns library - Tag Cloud**:
http://developer.yahoo.com/ypatterns/social/objects/collecting/tag/cloud.html

[32] **Wikipedia definition of TagCloud** - http://en.wikipedia.org/wiki/Tag_cloud

[33] **Automated Tag Clustering: Improving search and exploration in the tag space**
http://www.pui.ch/phred/automated_tag_clustering/

[34] **Designing Your Reputation System** - Bryce Glass
http://www.slideshare.net/soldierant/designing-your-reputation-system

[35] **I Love My Chicken Wire Mommy** by Ben Brown
http://xoxco.com/clickable/i-love-my-chicken-wire-mommy

[36] **Social Design Patterns for Reputation Systems**: Interview with Yahoo's Bryce Glass by Joshua Porter.
http://bokardo.com/archives/social-design-patterns-for-reputation-systems-one/
http://bokardo.com/archives/social-design-patterns-for-reputation-systems-two/

[37] **Social Gaming Summit panel writeups** by Jeremy Liew.
http://lsvp.wordpress.com/category/social-gaming

[38] **How to design a reputation system for your social media site or social game** by Jeremy Liew. http://lsvp.wordpress.com/2008/06/

[39] **The Competitive Spectrum** from the Yahoo Design Pattern Libary http://developer.yahoo.com/ypatterns/social/people/reputation/competitive.html

[40] **Identifying Labels** http://developer.yahoo.com/ypatterns/social/people/reputation/identifyinglabels.html

[41] **Labels** http://www.designingsocialinterfaces.com/patterns.wiki/index.php?title=Labels

[42] **Addicted to Achievements?** from the GamerScoreBlog.com http://gamerscoreblog.com/team/archive/2007/02/01/540575.aspx

[43] **Collectible Achievements** from the Yahoo Design Pattern Libary. http://developer.yahoo.com/ypatterns/social/people/reputation/achievements.html

[44] **Collectible Achievements** on UI. Pattern http://ui-patterns.com/patterns/CollectibleAchievements

[45] **Pattern Share this** http://www.designingsocialinterfaces.com/patterns/Share_This

[46] **Sharing made simple**: http://sharethis.com

[47] **Rating an Object** from the Yahoo Design Pattern Library http://developer.yahoo.com/ypatterns/social/objects/feedback/rating.html

[48] **Rate Content** on UI. Pattern http://ui-patterns.com/patterns/RateContent

[49] **StartBox**, rating star for prototype http://www.nickstakenburg.com/projects/starbox/

[50] **Google Maps APIs** website, http://code.google.com/intl/it-IT/apis/maps/documentation/javascript/

[51] **Google Maps APIs** signup for a key, http://code.google.com/intl/it-IT/apis/maps/signup.html

[52] **Google Maps** website, http://maps.google.com

[53] **MIT open source licence** http://www.opensource.org/licenses/mit-license.php

[54] **WebRatio, Custom Unit Guide** http://wiki.webratio.com/index.php/Custom_Unit_Guide

[55] **WebRatio, How to debug the code of a custom unit inside the Web application:** http://wiki.webratio.com/index.php/How_to_debug_the_code_of_a_custom_unit_inside_the_Web_application

[56] **JavaServer Pages Standard Tag Library** documentation http://java.sun.com/products/jsp/jstl/reference/docs/index.html

[57] **Parse XML with dom4j**: http://www.ibm.com/developerworks/library/x-dom4j.html

[58] **SVN integration in Eclipse/Webratio:** http://www.ibm.com/developerworks/opensource/library/os-ecl-subversion/