bastet

Manuale del Sistemista
v. 1.1

Bastet Balestra/Manzo Roma marco@crossbow.it, nando@crossbow.it

Introduzione

Bastet è un'applicazione concepita per la gestione di un sistema di trasmissione delle informazioni su pannelli video.

La metafora

Il sistema è stato ideato facendo ricorso alla metafora televisiva. Esiste un sistema di trasmissione (*Broadcast System*), su cui diverse **Reti** (*Network*) trasmettono i propri **Canali**. Le trasmissioni sono costituite di **Contenuti** (specifica proprietà di ogni Rete), ovvero dell'insieme degli **Outline** (aggregazione di **Programmi**).

Un monitor non ha dunque una programmazione propria: si sintonizza su un canale e trasmette la programmazione che è in corso su quel canale in quel momento.

A differenza di quanto accade nel caso del broadcast televisivo, però, i Monitor collegati vengono assegnati alle Reti dagli amministratori di sistema. *Un pannello si identifica come monitor dal nome scelto*; specificato il nome monitor non esiste poi alcuna scelta che compete al monitor. Solo a livello centralizzato è possibile cambiare il canale, o assegnare un diverso outline al canale.

Eventuale Monitor non censiti che accedono al server si sintonizzano in automatico sul **Canale prefedinito** (di *default*), sulla Rete che diventa così predefinita.

Risulta quindi intuitiva anche la strutturazione di quelli che sono gli *operatori* della rete, suddivisi tra **Direttori di Rete**, **Direttori di Canale** ed **Autori**, mentre le funzioni tecniche inerenti al sistema di trasmissione nel suo complesso saranno affidate a figure di **Amministratori**.

Notazione e convenzioni

Per la descrizione delle funzionalità del sistema, si farà uso di un accesso di tipo amministrativo. Questo garantirà la completezza della trattazione delle diverse funzionalità del sistema, nonostante la maggior parte degli utenti avranno, invece, accesso solo ad un subset delle funzionalità di Bastet.

Descrizione	Significato
Grassetto	Indica i termini che costituiscono comandi ed etichette presenti in Bastet
<descrizione>	Rappresentano elementi da sostituire con valori *attuali*, specificati da "descrizione"
`Code`	Convenzione tipografica per path, URL, comandi da console.

Notifica dei problemi

Ogni problema, richiesta di modifica, corredata da:

1. Indicazione dell'eventuale messaggio di errore presente
2. Indicazione dell'entità su cui il messaggio si è verificato
3. Parametri di connessione

Potrà essere inviato a bastet@crossbow.it.

Installazione e requisiti hardware e software

Bastet è distribuito in tre versioni, ognuno per le diverse piattaforme: Linux, Mac, Windows.

In questa sezione descriveremo le procedure di installazione. Bastet condivide con Mozilla Firefox la quasi totalità dei requisiti software/hardware necessari al suo corretto funzionamento. Per la visualizzazione dei video, è necessario assicurarsi che sia stato installato Quicktime (per Windows e Mac) o Mplayer (per Linux).

Generalità sul Client

L'ultima versione del client è disponibile per il download (protocollo http) all'URL `http://<server>/<directory_web>/client`, dove i dati sono quelli per il server.

Piattaforma	Operazioni
Mac	I. Effettuare il download della disk Image II. Montare il disco III. Copiare l'applicazione **bastet.app**
Linux	I. Effettuare il download dell'archivio in formato tar compresso II. Espandere l'archivio nella posizione desiderata: apparirà la directory Bastet III. Eseguire **bastet**
Windows	I. Effettuare il download dell'archivio in formato ZIP II. Espandere l'archivio nel percorso desiderato: apparirà un folder Bastet, contenete l'eseguibile *bastet.exe* III. Eseguire **bastet.exe**

Client modalità Display

Per avviare la trasmissione delle informazioni, seguire le seguenti istruzioni:

1. Nel pannello **Display**, inserire o selezionare il **Nome Pannello** desiderato;
2. Nel pannello **Server**, inserire i parametri di connessione, verificarli e salvarli;
3. Selezionare ancora una volta il pannello **Display**: la trasmissione apparirà automaticamente.

Client modalità Authoring/Amministrazione

1. Nel pannello **Display**, disabilitare l'autostart;
2. Nel pannello **Server**, inserire i parametri di connessione, validarli e salvarli;
3. Selezionare il pannello **Authoring**: inserire le credenziali fornite dall'amministratore del sistema.

Server

Il server di Bastet è fornito come macchina virtuale VM Ware.

Contenuto del documento

Dopo questo capitolo introduttivo, saranno descritte le caratteristiche lato Server di Bastet che possono essere modificate da un Sistemista.

Prerequisiti

Per poter seguire questa guida sono necessarie conoscenze di XHTML, javascript, CSS ed XSLT (il che ovviamente implica XML ed XPath).

Interventi su file di *outline* richiedono la conoscenza di XUL, "XML User-interface Language", ulteriori info: **https://developer.mozilla.org/en/XUL**

Per tutti gli interventi sul server è richiesta una minima familiarità con l'ambiente Linux ed i suoi strumenti base: bash o altra shell, SSH, SFTP, ecc.

Applicabilità

Le informazioni che seguono sono applicabili alla versione 1.1 di Bastet. Per versioni successive potrebbe essere necessaria una integrazione relativa a nuove funzionalità.

Esclusione

Le modalità di utilizzo del programma sono descritte nel Manuale Utente di Bastet.

Panoramica di base del Server

In questa sezione del manuale saranno descritte le caratteristiche del server applicativo di Bastet, e la disposizione dei file e delle directory che compongono il server applicativo.

La macchina virtuale

Il server di Bastet viene distribuito sotto forma di una macchina virtuale Vmware, con USB disattivata, senza installazione dei *Vmware tools* (non necessari), con ethernet in bridged mode.

Viene distribuita con 1GB di RAM assegnato.

Distribuzione e versioni

Il server è una distribuzione stable di `Debian Etch`, con le seguenti caratteristiche:

Pacchetto/modulo	Versione
Kernel	2.6.18-6-686
Apache 2	2.2.3
MySQL	5.0.32
Perl	5.8.8

Le librerie Perl utilizzate sono state ottimizzate tramite ricompilazione durante l'installazione da CPAN, ed utilizzano librerie native wrappate (moduli XS) dove le performances diventano particolarmente sensibili (ad es. per `XML::LibXML` e `XML::LibXSLT`, che si appoggiano ai parser nativi Gnome di `LibXML` e `LibXSLT`).

Qualunque modifica della distribuzione o versione di Linux, Perl, dei moduli Perl utilizzati, di Apache, dei moduli di Apache, di MySQL o di ognuna delle altre componenti di Bastet porrà il Server al di fuori della piattaforma testata, e quindi escluderà automaticamente qualunque tipo di supporto relativo alla piattaforma.

Credenziali

Le credenziali predefinite sono per gli utenti "root"e "bastet", con password identica al login name.

Utente	Password
root	root
bastet	bastet

Se cambia il MAC Address

Quando la macchina viene copiata viene richiesto se il server è stato copiato o spostato; lo scopo di questa domanda da parte di Vmware è decidere se sia o meno necessario generare un nuovo MAC Address da associare al server.

Nella maggior parte dei casi un solo server reggerà anche reti multiple, quindi non dovendo tenerne duplicati potremo rispondere che la macchina è stata spostata.

Nel caso in cui venga generato un nuovo MAC Address potrebbe essere necessario, come per ogni sistema Debian, aggiornare l'associazione del MAC Address con il nome device perché punti nuovamente ad `eth0`.

Questa associazione è definita in `/etc/udev/rules.d/z25_persistent-net.rules`

Il MAC Address predefinito per eth0 è `00:0c:29:d7:84:b8`

Servizi attivi di default

Oltre ad Apache e MySQL, sono attivi di default i servizi di SSH, e quindi SFTP.

Samba, FTP ed altri protocolli o servizi non sono stati attivati.

Le directory

Tutta l'installazione applicativa di Bastet si trova sotto la home dell'utente, "bastet", ovvero in `/home/bastet`. Qui distinguiamo tre directory principali:

/home/bastet/client

In questa directory trovano posto:

- Il client Linux (in formato ".tgz")
- Il client Mac (in formato ".dmg", per Mac Intel e PPC)
- Il client Windows (in formato ".zip")
- Il PDF del manuale utente
- Questo PDF
- i file `header.html` e `footer.html` che descrivono la versione dei client disponibili per il download.

/home/bastet/perl

Tutte le librerie Perl che costituiscono il Server di Bastet sono racchiuse in questa directory. La libreria specifica di Bastet è "Bt", ed è così strutturata:

- `Bt` - un modulo vuoto che funge da wrapper per le chiamate più comuni, come quelle relative ai path.
- `Bt::api` - raccoglie le API che possono venire invocate dalle CGI che usano Bastet
- `Bt::api::cmd` - i comandi disponibili per le API
- `Bt::api::pan` - i comandi per le API accessibili dalle CGI non autenticate in sicurezza (ovvero dai pannelli, non dagli autori).
- `Bt::lib` - le librerie interne di Bastet, utilizzate dalle API.
- `Bt::lib::priv` - i metodi ad uso delle librerie, non chiamati dalle API.

Questa gerarchia è l'unica ad essere stata sviluppata specificatamente per Bastet. A sua volta si appoggia ad altre librerie, sempre opera degli stessi autori:

- `F` - "Framework", il meta-framework applicativo disegnato per applicazioni client-server con interazioni AJAX (con una controparte nel client).
- `M` - "MyBase", una sorta di BIOS con le funzioni comuni di IO.
- `db` - La libreria di interfaccia con il database (tramite con `DBI` e `DBD`)
- `CMRA` - "Chimera", Centralized Multiple Re-Authenticator, modulo di riautenticazione utilizzato per svincolare il processo di autenticazione dell'utente dall'effettiva assegnazione dei privilegi. *Chimera è stato anche presentato all'Università di Roma TRE come progetto pilota nel corso del 2008*.

Tutte le librerie Perl presenti in `/home/bastet/perl` sono disponibili system-wide grazie ai link simbolici che sono stati creati in `/usr/local/lib/site_perl/`:

- `Bt -> /home/bastet/perl/Bt`
- `Bt.pm -> /home/bastet/perl/Bt.pm`
- `CMRA.pm -> /home/bastet/perl/CMRA/CMRA.pm`
- `db.pm -> /home/bastet/perl/db.pm`
- `F -> /home/bastet/perl/F`
- `F.pm -> /home/bastet/perl/F.pm`
- `M -> /home/bastet/perl/M`
- `M.pm -> /home/bastet/perl/M.pm`

/home/bastet/site

È la root del server web di default definito in Apache2.

Al di sotto di questa directory trovano posto:

- La `favicon.ico`
- Il file `index.html` e le directory con le pagine per il default del sito
- La directory `misc`, che contiene elementi di configurazione (non è accessibile dal web, interdetta dal suo file `.htaccess`)
- La directory `styles`, che contiene i template HTML per le CGI.
- **La root applicativa di bastet**, ovvero la directory che corrisponde ad ognuno dei sistemi di broadcast serviti. Il nome di questa directory è il nome del corrispondente sistema di broadcast, ed è questo che dobbiamo inserire sul client insieme all'indirizzo IP ed alla porta per decidere quale sia il sistema di broadcast a cui accedere tra i tanti che potenzialmente un server bastet può gestire in parallelo.
  Il default fornito è "`bastet`".

Vedremo ora nel dettaglio le directory principali al di sotto della root applicativa di bastet.

/home/bastet/site/<*bastet*>

Contiene le tre CGI attraverso cui passa tutto il traffico tra pannelli, sistema di authoring e server web:

- `index.cgi` - deputata allo smistamento di tutto il traffico AJAX autenticato del sistema di authoring, tramite `Bt::api`
- `login.cgi` - sottende alla sola autenticazione del modulo di authoring, tramite il modulo Chimera
- `panel.cgi` - gestisce il traffico non autenticato, ovvero le interazioni dei pannelli con il sistema di gestione.

Solo la determinazione del canale di appartenenza passa per una CGI, tutto il resto del traffico dei pannelli avviene tramite il solo server Apache (si tratta di file statici).

Questo permette al server di Bastet di supportare carichi anche massicci di pannelli, con pochissimo sforzo.

/home/bastet/site/<*bastet*>/client

La directory `/home/bastet/site/<bastet>` contiene anche la directory da cui prelevare il client, che può essere un semplice link simbolico al client generale o contenere una versione dedicata per lo specifico sistema di broadcast.

/home/bastet/site/<*bastet*>/log

Una directory specializzata per l'accesso via browser ai log mantenuti nel database.

È costituita da una sola CGI, che si avvale dei templates HTML definiti nella directory `/home/bastet/site/styles`

/home/bastet/site/<*bastet*>/version.xml

Questo file statico è quello la cui esistenza (come XML well-formed) conferma al client la validità dei parametri immessi per contattare il Server.

/home/bastet/site/<*bastet*>/feeder

Questa directory contiene due CGI di servizio dedicate alla raccolta dei feed esterni per la loro inclusione nei programmi di Bastet che prevedono tale feed.

Sebbene normalmente sia sufficiente limitarsi a duplicarne il contenuto, la possibilità di effettuare customizzazioni sul modulo di *grab* di dati esterni ci ha spinto a racchiudere queste CGI (e la loro directory) al di sotto della gerarchia del sistema di broadcast (ovvero la directory "bastet" di primo livello del sito).

Lo *schedule* della raccolta di feed esterni viene gestito tramite il `cron` dell'utente bastet, che richiama la CGI `http://localhost/<bastet>/feeder/feed.cgi` la quale, a sua volta, richiama più volte come CGI `feed1.cgi`

Abbiamo scelto di chiamare questi script tramite CGI al fine di avvalerci di Apache come application server, cosa che sa fare benissimo.

Più avanti il processo di raccolta dati da feed esterno verrà discusso nel dettaglio.

/home/bastet/site/<*bastet*>/static

Questa è la directory che contiene tutti i file statici XML (ed XSLT, anche loro XML) utilizzati in Bastet.

Alcuni sono personalizzabili, altri sono cambiati dinamicamente dal sistema, ed ora vedremo nel dettaglio il significato delle sottodirectory ed il loro contenuto.

Una nota: questa directory può essere acceduta dal web utilizzando l'URL:

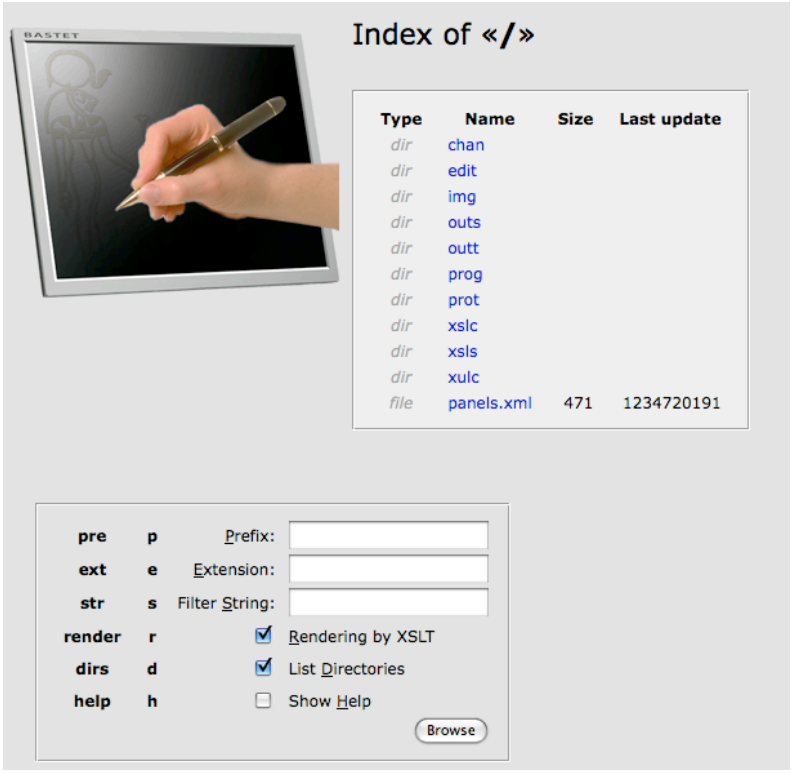
`http://<ip o nome server>/<bastet o nome broadcast>/static`

L'index di questa directory è una CGI che risponde in XML, e questo permette al client di stabilire quali file siano cambiati (e siano quindi da scaricare di nuovo) tramite una transazione AJAX che elenca il contenuto di ogni directory.

Se si desidera sfogliare la directory è utile sapere che esistono alcuni parametri aggiuntici che la CGI riconosce:

Parametro	Valore	Significato
"h" o "help"	non falso (i.e.: 1)	Visualizza l'aiuto
"r" o "render"	non falso (i.e.: 1)	Utilizza XSLT per il rendering in HTML
"d" o "dirs"	non falso (i.e.: 1)	Mostra le directory
"p" o "pre"	stringa alfanumerica	Filtra i file per prefisso
"e" o "ext"	stringa alfanumerica	Filtra i file per estensione
"s" o "str"	stringa alfanumerica	Filtra sia i file che le directory per il nome che contenga il valore

Utilizzando `http://<ip>/<bastet>/static/?r=1&d=1` si ottiene:



In questo modo è possibile sfogliare comodamente la directory dei file statici, con hyperlink, per visualizzarne i contenuti.

L'XSLT responsabile di questa trasformazione è `.dir.xsl`, nella directory `static`. La trasformazione viene effettuata dal browser stesso, grazie alla direttiva che viene acclusa all'XML tornato ("processing-instruction").

Per convenzione le directory presenti che andiamo a dettagliare hanno un nome di quattro lettere, ricordando il codice tipo/creatore a 4 byte del vecchio System Mac.

/home/bastet/site/<*bastet*>/static/chan

Si tratta di una directory ad aggiornamento automatico (a cura di Bastet).

Contiene i file XML dei canali in trasmissione. Ogni file ha come nome l'ID del canale, e come contenuto una copia dell'Outline che il canale sta trasmettendo.

/home/bastet/site/<*bastet*>/static/edit

Una directory non aggiornata (o ad aggiornamento manuale).

Contiene i file XLST che costituiscono gli *editor* dei programmi. Si tratta di template XSL estremamente complessi, che rispondono all'antico problema di generazione automatica della form sulla base della definizione dei dati.

Si consiglia di usare estrema cautela nel manipolare questi file, e di modificarli solo se si è assolutamente certi di quel che si sta facendo.

Utilizzare sempre e solo editor di testo con pieno supporto di UTF-8; in (S)FTP trasferire i file in modalità binaria, ed effettuare sempre un backup.

Ogni tipo di editor è definito da una coppia di file, un "-xul.xsl" che definisce lo XUL dell'editor sulla base del programma, ed il "-js.xsl" che ne definisce il codice javascript sulla base del programma.

Il nome di ogni file è composto come:

```
<editor type>-<versione>-<"xul" o "js">.xsl
```

La versione ("001") è scolpita nel client, questo permette di sperimentare nuove versioni di editor sullo stesso server e sugli stessi programmi senza per questo creare problemi ai client in produzione, utilizzando un client in beta.

Il tipo di editor (attualmente "simple" o "compound") è definito nel *Program Template* ("prot", lo vedremo in seguito).

/home/bastet/site/<*bastet*>/static/outs

Si tratta di una directory ad aggiornamento automatico (a cura di Bastet).

Contiene i file XML degli outline definiti. Ogni file ha come nome l'ID dell'outline.

/home/bastet/site/<*bastet*>/static/outt

Una directory non aggiornata (o ad aggiornamento manuale).

Contiene la definizione XML degli "Outline Type". Vedremo più avanti come definire nuovi *Outline Type*, e quali XSLT sono collegati.

/home/bastet/site/<*bastet*>/static/prog

Si tratta di una directory ad aggiornamento automatico (a cura di Bastet).

Contiene i file XML dei programmi definiti. Ogni file ha come nome l'ID del programma.

/home/bastet/site/<*bastet*>/static/prot

Una directory non aggiornata (o ad aggiornamento manuale).

Contiene la definizione XML dei "Program Type". Vedremo più avanti come definire nuovi *Program Type*, e quali XSLT sono collegati.

/home/bastet/site/<*bastet*>/static/xslc

Una directory non aggiornata (o ad aggiornamento manuale).

Contiene i file XSL ad uso del Client, ovvero quelli referenziati sia dall'Outline (come definito nel corrispondente Outline Type, per creare la "gabbia" XUL dei diversi oggetti browser) che dal Program (come definito nel corrispondente Program Type, per creare l'XHTML da renderizzare nel singolo oggetto browser).

/home/bastet/site/<*bastet*>/static/xsls

Una directory non aggiornata (o ad aggiornamento manuale).

Contiene i file XSL as uso del Server. In particolare contiene i file che trasformano i diversi tipi di feed esterno in una struttura di programma a blocchi di testo, ovvero `<feed type>2blocks.xsl`, e il file `rssify.xsl` che armonizza i diversi tipi di feed news (RSS, Atom ed RDF) in una versione RSS adatta a `rss2blocks.xsl`.

Questi file vengono eseguiti sul Server tramite il parser XSL `LibXSLT`, e possono quindi avvalersi delle estensioni EXSLT supportate da LibXSLT (vedi: http://exslt.org).

/home/bastet/site/<*bastet*>/static/xulc

Una directory non aggiornata (o ad aggiornamento manuale).

Contiene il file XUL associato ad ogni Outline Type, da trasformare con l'XSLT specificato nella definizione dell'Outline Type stesso.

Il database

Un database per ogni broadcast system

Ricordiamo che Bastet server può gestire più broadcast system distinti, sebbene per default venga fornito con il solo broadcast system predefinito "bastet".

Ogni broadcast system dispone di un proprio database, le cui coordinate per l'accesso sono definite nel file `/home/bastet/site/misc/dbini.xml`.

Qui sotto il nodo `root` troviamo un nodo per ogni sistema di broadcast definito, ad esempio per "`bastet`":

```
<?xml version="1.0" encoding="utf-8"?>
<root>
  <bastet>
       <driver>DBI:mysql</driver>
       <db>bastet</db>
       <host>localhost</host>
       <additional>mysql_enable_utf8=1</additional>
       <user>bastet</user>
       <pass>fg45VhDf</pass>
       <AutoCommit>0</AutoCommit>
       <RaiseError>0</RaiseError>
  </bastet>
</root>
```

Utilizzare un server database remoto

Il database può anche risiedere su un server distinto, è sufficiente che le coordinate di puntamento al database vengano aggiornate per il nodo del broadcast system.

Potremmo definire un nodo `bastetTest` per un nuovo sistema:

```
  <bastetTest>
       <driver>DBI:mysql</driver>
       <db>bastetTestDatabase</db>
       <host>81.82.83.84</host>
       <additional>mysql_enable_utf8=1</additional>
       <user>bastet</user>
       <pass>xgghfjjjrui777</pass>
       <AutoCommit>0</AutoCommit>
       <RaiseError>0</RaiseError>
  </bastetTest>
```

Qualunque sia il server, gli statement SQL attualmente implementati sono ottimizzati per il database server MySQL versione 5.

Cambiare tipologia di server Database

Sebbene non sia parte di un intervento sistemistico ordinario, è comunque possibile con uno sforzo di sviluppo applicativo modesto.

Il codice SQL non è integrato nel Perl, anzi è tutto completamente racchiuso in template in un file XML: `/home/bastet/site/misc/query.xml`.

Un sistema completamente automatizzato di costruzione (realizzato dai moduli Perl `F` e `db`) si avvale di questi template per costruire dinamicamente le query a prova di *SQL injection* basandosi sulle regole definite nel file XML *core* della gestione del database: `/home/bastet/site/misc/dbstructure.xml`.

La modifica di questo file è riservata agli sviluppatori di Bastet, ma il meccanismo è da citare perché implementa una astrazione delle regole di database rispetto al database effettivo.

È quindi teoricamente possibile cambiare motore di database, ad esempio passando da MySQL a PostgreSQL o Oracle, ridefinendo le Stored Procedures e adattando le maschere SQL definite in `query.xml`, e con un ritocco su `db.pm`.

Un simile intervento riguarda comunque lo sviluppo core dell'applicazione, ed esula da un intervento di carattere sistemistico.

Qualunque modifica dei moduli Perl forniti o dei file di configurazione diversi da db.ini porrà il Server al di fuori della piattaforma testata, e quindi escluderà automaticamente qualunque tipo di supporto relativo alla piattaforma.

Il client

La *User Chrome*

Il client di Bastet è basato su XULRunner. È una *applicazione Mozilla*, come lo sono ad esempio Firefox o Thunderbird, basata sulla distribuzione multipiattaforma di *XULRunner*.

Ulteriori informazioni: https://developer.mozilla.org/en/XULRunner

Come accade per ogni applicazione Mozilla, al primo avvio viene creata un profilo utente, e per questo motivo il primo avvio (più lento dei successivi) avviene due volte: il programma si avvia, si accorge che non esiste un profilo utente, lo crea, e si riavvia di nuovo.

La *User Chrome* si trova all'interno del profilo utente, il quale si trova a sua volta nella cartella di supporto applicazione. A seconda delle diverse piattaforme (Linux, Mac, Windows) questa si trova sotto una gerarchia leggermente diversa.

Il profilo utente su Linux

Viene creata una cartella nella home dell'utente con il nome `.<vendor>`, il "." iniziale serve a non renderla visibile, al suo interno una cartella con il nome applicativo e lì, finalmente, i profili che contengono la User Chrome.

Per Bastet il path che porta alla cartella dei profili è `~/.www.crossbow.it/Bastet/`

Cancellando questa cartella (o direttamente `~/.www.crossbow.it`, se non sono state installate altre applicazioni dello stesso *vendor*) sarà come lanciare Bastet per la prima volta.

Il profilo utente su Mac OS X

Per Mac OS X il *vendor* viene ignorato, e la cartella dei profili si trova più semplicemente in `~/Library/Application Support/Bastet/Profiles`

Il profilo utente su Windows

Su Windows si trova nella cartella dei Dati applicazione dell'utente, che potrebbe avere un path come: `C:\Documents and Settings\<utente>\Dati applicazioni` oppure `C:\Documents and Settings\<utente>\Application Data`.

Al suo interno troviamo la cartella `www.crossbow.it\Bastet\Profiles`.

Ancora una volta cancellando la cartella `Bastet` (o direttamente `www.crossbow.it`, se non sono state installate altre applicazioni dello stesso *vendor*) sarà come lanciare Bastet per la prima volta.

La cache di lookup

Il client di authoring effettuano un caching dei valori di lookup, e i tipi di outline disponibili rientrano nei valori che sono in questa cache.

Ogni nuovo client (o nuovo utente che usa il client per la prima volta) scarica e mantiene una copia aggiornata di questa cache.

Per forzare il refresh di questa cache su tutti i client occorre agire sul server web, incrementando il *numero di lookup* che è normalmente definito come la data in di ultima modifica seguita da un numero sequenziale a tre cifre (YYYYMMDDnnn) nell'attributo `/root/@lookup` del file `/home/bastet/site/<bastet>/version.xml`

Quando al login il client si accorge che il *numero di lookup* è maggiore di quello associato alla propria *cache di lookup*, considera scaduta la propria cache ed acquisisce nuovamente i valori di lookup da cachare.

A seguito di un ricaricamento di *Outline Type* e *Program Type* il *numero di lookup* viene automaticamente incrementato.

Un simile evento può rendersi necessario solo in seguito ad un intervento di carattere sistemistico, di conseguenza non è stata realizzata alcuna altre interfaccia di controllo.

Gli Outline

Gli outline sono raccolte di sequenze di programmi che vengono eseguiti per il tempo noto dentro ogni singolo frame.

La modifica di un outline esistente richiede conoscenze di XUL, "XML User-interface Language": https://developer.mozilla.org/en/XUL

La definizione di nuovi Outline richiede un intervento su database MySQL, e quindi la conoscenza di SQL o di tools appropriati.

outt - Outline Type

Un esempio di Outline Type, `/home/bastet/site/bastet/static/outt/1.xml`:

```
<root id="1" refresh="120" xsl="chancopy" label="Outline Base" >
  <frame id="F4" label="Header"/>
  <frame id="F1" label="Body"/>
  <frame id="F3" label="Sidebar"/>
  <frame id="F2" label="Footer"/>
</root>
```

Tutti i file XML riportano come attributo di root "id" ("/root/@id") il proprio nome, ovvero l'ID numerico definito nel database.

/root

L'outline type riporta anche l'attributo "refresh" che propone l'intervallo di default (in secondi) dopo il quale un client verifica aggiornamenti sul canale, e l'attributo "xsl" specifica il nome del file in `static/xslc` che si occuperà di trasformare il template XUL associato nel DOM XUL effettivamente utilizzato nel pannello di proiezione. Il file `static/xulc/chancopy.xsl` fa esattamente quel che il nome suggerisce: una copia dell'XML di canale che riceve, trasparente.

L'attributo `label` è il nome dell'*Outline Type*, auspicabilmente univoco.

/root/frame

L'outline type riporta un elenco di *frames*, che verranno riportati anche nell'ambiente di authoring.

Ogni frame è dotato di un attributo "id" che lo collega al nodo browser corrispondente in `static/xulc/</root/@id>.xul`, e di un attributo "label" che identifica quel frame nell'interfaccia di authoring del client.

Definire un nuovo Outline Type

Per definire un nuovo Outline Type occorre definire:

- Il file XML di outline type in `static/outt`

- Il file XUL di modello omonimo in `static/xulc`
- Il file XSLT di trasformazione XUL in `static/xslc`, a meno che non si usi un file XSLT di trasformazione già esistente (come `chancopy.xsl`)

Una volta completati questi passi occorre ricaricare i template.

outs - La struttura di un Outline

Un outline è un file XML statico, la cui struttura è del tipo:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<root id="1" refresh="120" xsl="chancopy" outlinetype="1">
  <frame id="F4" label="Header">
       <program id="17" duration="30" xsl="5"/>
  </frame>
  <frame id="F1" label="Body">
       <program id="1" duration="60" xsl="1"/>
       <program id="19" duration="90" xsl="1"/>
       <program id="25" duration="300" xsl="1"/>
  </frame>
  <frame id="F3" label="Sidebar">
       <program id="15" duration="30" xsl="1"/>
  </frame>
  <frame id="F2" label="Footer">
       <program id="2" duration="30" xsl="2"/>
  </frame>
</root>
```

/root

Come recita l'ID `/root/@id`, si tratta dell'outline 1 (`static/outs/1.xml`).

La struttura di qualunque outline è basata sul corrispondente *Outline Type*, il cui ID è referenziato dall'omonimo attributo `/root/@outlinetype` (`static/outt/1.xml`).

Gli attributi `/root/@xsl` e `/root/@refresh` hanno il significato già discusso in merito agli Outline Type.

/root/frame

Questi nodi sono la duplicazione di quelli discussi in Outline Type, ma se lì sono vuoti qui vengono popolati con i nodi `program` che indicano la sequenza di programmazione.

Non esiste una relazione diretta tra la nidificazione degli oggetti browser e la loro rappresentazione nell'XML di Outline, il legame è rappresentato dall'id.

Una considerazione sul rendering: **quando il client effettua il rendering dei diversi frame un frame privo di programmi** (o popolato con programmi che non possono essere scaricati o utilizzati) **collassa automaticamente, senza bisogno di ulteriori istruzioni**.

/root/frame/program

Questi nodi vengono popolati automaticamente da Bastet, sono l'effetto dei *link* dei programmi all'interno dei frame di un Outline che viene effettuato nell'ambiente di Authoring, in Drag&Drop.

Ognuno di questi nodi riporta una durata in secondi che potrebbe variare rispetto al programma originale a cui è riferito, e duplica l'informazione dell'XSL client (`xslc`) associato per rendere più semplice ed efficiente il lavoro di *pre-caching* che ogni client effettua prima di iniziare la proiezione.

I metadati e la sicurezza associata agli outline

I metadati (proprietario e rete) e la sicurezza associata all'Outline sono gestiti dal database, per semplificare e rendere più efficiente l'aggregazione degli stessi, nella tabella `btouline`.

La sicurezza è gestita sul modello Unix ottale `rwxrwxrwx`, dove le tre cifre ottali rappresentano il proprietario dell'outline (`u`), il suo network di appartenenza (`g`) ed infine tutti gli altri (`o`). Il significato dei bit `r` e `w` rimane inalterato (permessi di lettura e scrittura), mentre il bit di esecuzione `x` indica la possibilità di utilizzare l'outline in trasmissione, di eseguirlo ovvero di trasmetterlo.

Questo permette la creazione di un workflow abbastanza elementare, impedendo la pubblicazione di un outline ancora in lavorazione tramite la sua *cmask*.

I valori ottali della sicurezza sono riportati nella colonna `cmask`, nella rappresentazione a stringa di 4 caratteri della forma ottale (ad esempio `0752` significa `rwxr-xr--`).

Ambito della sicurezza applicativa e ulteriori restrizioni

È importante ricordare che la sicurezza (in lettura) è riferita esclusivamente all'ambiente di authoring del client: tutti i file XML sotto `static` sono accessibili direttamente dal server web Apache da qualunque macchina che possa raggiungere la porta 80 del server.

Eventuali limitazioni a questa apertura possono essere definite nei modi classici:

- regole di routing di un eventuale router esterno alla LAN del server
- regole del firewall integrato in Linux (il server linux è già dotato dell'utility `firestarter`, da eseguire tramite X11 su tunnel ssh)
- utilizzo di file `.htaccess` di Apache2 (il server web di default è già impostato con la direttiva `AllowOverride All`).

I programmi

Il *programma* rappresenta il mattone della programmazione dei pannelli, l'elemento più basso ed al tempo stesso quello attorno a cui si concentra la maggior parte della tecnologia utilizzata e dello sviluppo accessorio.

Come accade per gli Outline, distinguiamo tra i *Program Template* (`static/prot`) ed i *Programmi* veri e propri (`static/prog`).

La comprensione del programma passa per la comprensione del Program Template associato: sotto il nodo `/root` il `prot` presenta il template dei parametri `/root/defaults` che diventeranno nel `prog` i parametri del programma `/root/params`, ed il template dei dati `/root/program` che diventeranno i dati del programma `/root/data`.

Il program Template - prot

Nel program template è registrata la definizione del programma tesso, dei suoi parametri, dei suoi dati e di tutti gli elementi di interfaccia che ne costituiranno l'interfaccia di editing tramite la doppia trasformazione degli XSLT di `static/edit`.

In effetti per l'edit l'effettiva trasformazione avviene basandosi sul *combo* di `prog` e `prot` corrispondente, ovvero la miscela dei nodi `/root/defaults` e `/root/program` sotto il nodo `/root` del programma che si va ad editare.

Ma i parametri ed i campi che ci interessano maggiormente risiedono tutti nel prot, mentre i campi del prog (che vedremo in seguito) sono interessanti ai fini della trasformazione sul client per la proiezione del programma, ovvero il rendering finale in XHTML.

/root/@xsl

L'attributo `xsl` del nodo di root del `prot` identifica il file XSLT (in `static/xslc`) che verrà utilizzato per il rendering finale.

Non verrà riportato nel `prog` corrispondente, ma in ogni *program link* (ovvero in ogni nodo `program` presente all'interno di un Outline).

Programmi diversi potranno utilizzare lo stesso XSLT, ad esempio esiste un solo XSL predefinitito (`static/xslc/1.xsl`) per i programmi con blocchi a scorrimento verticale.

/root/defaults/param - Form per i parametri

I parametri predefiniti di un program template sono elencati come nodi `param` sotto il nodo `/root/default`. Ogni nodo definisce un parametro del programma che verrà passato all'XSLT di rendering, ed i suoi attributi definiscono la form che sarà associata al parametro se verrà reso editabile.

I parametri non editabili vengono passati trasparentemente dal `prot` al `prog` corrispondente quando il programma viene salvato.

Segue un elenco di attributi possibili per ogni nodo **`param`**:

- **`name`** (Obbligatorio)
  Stringa alfanumerica priva di spazi che inizia con una lettera, case sensitive. Diventerà il nome del parametro del programma (attributo `name`), ed il nome dei parametri che il client passerà all'XSLT per il rendering in XHTML.
  Il valore di questo attributo non è mai modificabile in edit.
- **`value`** (Obbligatorio)
  Il valore di default del parametro, qualunque esso sia.
- **`editable`**
  Se questo attributo esiste con valore "`yes`" allora il parametro sarà editabile tra i parametri del programma.
- **`type`** (Obbligatorio per i parametri *editable*)
  Definisce il tipo di interfaccia che effettuerà l'edit del parametro. I tipi attualmente previsti sono "`choose`" (scelta da menu), "`rgbcolor`" (scelta da palette colore), "`text`" (campo di testo semplice), "`textarea`" (campo di testo multilinea), ed "`html`" (editor HTML WYSIWYG).
  In genere i parametri editabili (a differenza dei contenuti) sono di tipo `choose`.
- **`label`**
  È il nome del parametro nella form di edit.
  Se manca viene utilizzato il valore dell'attributo `name`.
- **`values`** (obbligatorio per i valori *editable* di tipo `choose`)
  Una lista separata da *pipe* ("|") dei possibili valori del menu di scelta. Uno dei valori deve essere quello riportato nell'attributo di default `value`.
- **`labels`** (opzionale per i valori *editable* di tipo `choose`)
  Una lista separata da *pipe* ("|") e della stessa lunghezza di quella di `values`, riporta le etichette da associare ai valori presenti in `values`.
  Se manca vengono utilizzati i valori di `values`.
- **`tooltip`**
  L'eventuale tooltip text (il callout al rollover) che si vuole associare al campo.

Ulteriori attributi eventualmente presenti sono sperimentali, non utilizzati.

Nell'inserire i valori tenere presente che si tratta di file XML encodati in UTF-8: utilizzare direttamente i caratteri accentati o estesi (anche cinese, arabo, giapponese ecc.), non utilizzare le *entities* dell'XHTML ma solo quelle dell'XML ove necessario (ad esempio per `&quot;`, `&amp;`, `&lt;` o `&gt;`), in forma decimale (`&#32;`) o esadecimale (`&#x20;`).

Il file XML deve essere *well-formed*, o bloccherà l'edit e la gestione di tutti i programmi associati al *program type* corrotto, con conseguenze anche globali.

/root/program

Sotto questo nodo viene definita la struttura dati risultante del programma, ed il tipo di editor.

/root/program/editor

Definisce il tipo di editor associato al tipo di programma, ed il nome con cui il tipo di programma viene identificato durante la fase di authoring. I suoi attributi sono:

- **`type`** (Obbligatorio)
  I possibili valori sono quelli disponibili in `static/edit`, con bastet vengono forniti un editor di tipo "`compound`" ed un editor di tipo "`simple`", che sono i due possibili valori dell'attributo.
  L'editor `compound` supporta una struttura dati di uno o più nodi `block` che possono contenere uno o più nodi `chunk` (entrambi definiti una sola volta).
  L'editor `simple` supporta una struttura apparentemente più piatta, di uno o più nodi `block` contenenti un numero noto e fisso di elementi `chunk`, ma sebbene il nodo `block` sia definito comunque una sola volta con l'editor `simple` è possibile definire un numero finito di nodi `chunk` uno distinto dall'altro.
- **`program`** (Obbligatorio)
  Indica il nome del *program type* come visualizzato nell'editor e riportato nel database, ed è auspicabilmente univoco.

Form per i dati

`/root/program/block` e `/root/program/chunk` sono i due nodi destinati a contenere i dati, con il nodo `block` contenitore ed il nodo `chunk` contenuto finale.

Qui dobbiamo descrivere la form di questa struttura `block/chunk` risultante, con attributi di ogni nodo e contenuto (nodo testuale di valore del nodo).

/root/program/block - attributi

L'unico attributo che un nodo `/root/program/block` di `prot` può avere è l'attributo **`compound="yes"`**. La presenza di questo attributo indica che il blocco non è univoco, ma il programma potrà contenere *n* blocchi. Vale sia per editor `simple` che `compound`.

/root/program/chunk - attributi

Anche il nodo `chunk` ha come unico possibile attributo **`compound="yes"`**, e solo per per l'editor di tipo `compound`. L'editor `simple` non supporta `chunk` compositi (*compound*), in compenso permette di definire puntualmente uno o più nodi `chunk` con attributi e form diversi.

/root/program/block e /root/program/chunk

Entrambi questi nodi possono avere dei nodi che definiscono il tipo di dato (e la form associata) di ogni attributo e del contenuto del rispettivo nodo che verrà posto nei dati.

Per definire gli attributi useranno un nodo di nome `attribute`, che dovrà definire nome e valore di ogni attributo di `block` o `chunk`.

I possibili attributi per il nodo **<attribute />** sono sostanzialmente quelle che abbiamo elencato per i nodi `defaults/param`, ovvero:

- **name** (Obbligatorio)
  Stringa alfanumerica priva di spazi che inizia con una lettera, case sensitive. Diventerà il nome dell'attributo.
- **type** (Obbligatorio)
  Definisce il tipo di interfaccia che effettuerà l'edit del parametro. I tipi attualmente previsti sono quelli visti: "`choose`", "`rgbcolor`" o "`text`", con l'aggiunta del tipo "`hidden`" (form non visualizzata, valore definito dall'attributo `value`).
- **value** (solo per `type="hidden"`)
  Il valore dell'attributo, obbligatorio solo per attributi di tipo `hidden`.
- **label**
  È il nome del parametro nella form di edit.
  Se manca viene utilizzato il valore dell'attributo `name`.
- **values** (obbligatorio, solo per `type="choose"`)
  Una lista separata da *pipe* ("|") dei possibili valori del menu di scelta.
- **labels** (opzionale, solo per `type="choose"`)
  Una lista separata da *pipe* ("|") e della stessa lunghezza di quella di `values`, riporta le etichette da associare ai valori presenti in `values`.
  Se manca vengono utilizzati i valori di `values`.
- **tooltip**
  L'eventuale tooltip text (il callout al rollover) che si vuole associare al campo.

Per l'eventuale definizione del contenuto del nodo (il valore del testo nel nodo XML) si utilizza l'elemento **<content />**. Ancora una volta questo nodo ha attributi di definizione analoghi, inevitabilmente con qualche leggera differenza (ad esempio non ha un `name`):

- **type** (Obbligatorio)
  Definisce il tipo di interfaccia che effettuerà l'edit del parametro. I tipi attualmente previsti sono quelli visti: "`choose`", "`rgbcolor`", "`text`", "`textarea`" o "`html`".
- **label**
  Il nome dell'elemento nella form di edit.
- **values** (obbligatorio per i valori *editable* di tipo `choose`)
  Una lista separata da *pipe* ("|") dei possibili valori del menu di scelta.
- **labels** (opzionale per i valori *editable* di tipo `choose`)
  Una lista separata da *pipe* ("|") che riporta le etichette da associare ai valori presenti in `values`. Se manca vengono utilizzati i valori di `values`.
- **tooltip**
  L'eventuale tooltip text (il callout al rollover) che si vuole associare al campo.

Chiaramente il contenuto di un nodo privo di figli (tipicamente -non necessariamente- un `chunk`) si presta particolarmente bene ad ospitare dati di tipo "`html`" (con editor WYSIWYG).

I programmi con feed esterno

Alcuni programmi possono attingere dati da una sorgente esterna, purché la sorgente esterna sia un XML raggiungibile dal server in `http`.

Le CGI definite in *feeder* si occupano proprio di questo: da tutti i programmi attivi (linkati ad un outline che viene trasmesso) vengono selezionati solo quelli che contengono un parametro di nome `feeduri`. Come il nome suggerisce, il valore di questo attributo deve contenere la URI che porta al file XML esterno da cui prelevare le informazioni.

Prelevato l'XML da Internet, il comportamento dipende dal valore del parametro `feedtype`:

- se `feedtype="brss"`, ovvero un RSS potenzialmente *broken*: i nodi che possono rompere lo schema XML vengono racchiusi in un gruppo `<![CDATA[… ]]>` con regular expressions, quindi il `feedtype` viene considerato come "`rss`" e si procede con l'elaborazione ai punti successivi
- se `feedtype="rss"` questo include i feed di tipo RSS 0.98, 0.99 ecc, nonché i feed di tipo Atom e RDF. Per armonizzare i diversi tipi ogni feed RSS viene passato attraverso l'XSL `static/xsls/rssify.xsl`, che lo riporta ad una versione 0.99 di RSS ben nota, poi l'elaborazione prosegue con il passo successivo.
- L'XML viene passato attraverso un XSLT preso da `static/xsls` per trasformarlo in una struttura `block/chunk` di dati. Il nome del template XSL deputato alla trasformazione è `<feedtype>2blocks.xsl`

Se arrivati a questo punto abbiamo almeno un blocco di risultato, la struttura `block/chunk` ottenuta viene appesa sotto al nodo `/root/data` del programma corrispondente, e l'attributo `/root/data/@lastupdate` viene aggiornato con il time (unix) corrente.

Il *program* ottenuto sarà strutturalmente identico a quello ottenibile con un programma a blocchi verticali con editor di tipo `compound`, basato su un `prot` come `1.xsl`, e con la stessa resa a schermo (scorrimento verticale).

Definire un nuovo tipo di feed esterno

Definire un nuovo tipo di feed esterno richiede la scrittura dell'XSL che trasformi il feed esterno in una struttura `block/chunk`, e la modifica dei valori per il parametro `feedtype` del program type (`prot`) dei feed esterni.

Eventuali pre-trattamenti necessari (filtri puntuali, come per il `feedtype="mrbs"` che costruisce un URL basato sui parametri e sulla data/ora) dovranno essere implementati nelle CGI specifiche di `/home/bastet/site/<bastet>/feeder/`.

Per default in Bastet il program type per feed esterno è destinato allo scorrimento verticale. Se si desidera un rendering a schermo diverso dallo scorrimento verticale sarà necessario definire un nuovo program type con caratteristiche analoghe a quelle del program type esistente, ma XSL di rendering (*program->XHTML*) diverso.

Controllo periodico dei feed

Di questa operazione si occupa il demone `cron`, nativamente disponibile nei sistemi **nix*.

Come detto il processo viene richiamato tramite CGI, come si può verificare accedendo in SSH al server di Bastet:

```
bastet:~# crontab -lu bastet
# m h  dom mon dow   command
*/2 * * * * wget -q -O - http://localhost/bastet/feeder/feed.cgi
```

Il tempo di default impostato per il controllo è ogni due minuti, e i tempi possono essere variati secondo le modalità tradizionali di `cron`.

È sconsigliabile stringere troppo l'intervallo, si rischia di far partire un nuovo controllo mentre il precedente è ancora in corso d'opera. D'altra parte è possibile limitare i giorni, disponendo controlli ogni due minuti dal lunedì al venerdì ed ogni ora nel fine settimana.

Controllare uno specifico feed

È possibile controllare uno specifico feed nello stesso identico modo in cui il controllo periodico lo invoca, ovvero come viene controllato dal client premendo il pulsante "Ricarica feed". Come detto si tratta di una cgi, `feed1.cgi`, che prevece il parametro `prog` con l'ID numerico del programma da verificare:

`http://<ip>/<bastet>/feeder/feed1.cgi?prog=1`

La CGI risponde in `text/plain con` il numero dei blocchi scaricati.

Con il parametro opzionale `out=xml` la CGI risponde un XML:

`http://<ip>/<bastet>/feeder/feed1.cgi?prog=1&out=xml`

In entrambi i casi viene aggiornato l'XML del programma.

La CGI `feed.cgi` può essere ristretta con un file `.htaccess` per prevenirne l'accesso da IP diversi da *localhost*. La CGI `feed1.cgi` viene invece invocata dai client quando si preme il pulsante "Aggiorna feed", e non può quindi essere limitata a *localhost*.

Definire un nuovo Program Type

Per definire un nuovo *Program Type* occorre definire:

- Il file XML di program type in `static/prot`
- Il file XSLT di trasformazione in XHTML in `static/xslc`, a meno che non si usi un template XSLT esistente (ad esempio `1.xsl` per i programmi con `block` o `block/chunk` per lo scorrimento verticale)

Una volta completati questi passi occorre ricaricare i template.

Il programma - prog

Il programma è un file XML definito sulla base delle informazioni presenti nel `prot` corrispondente. La manutenzione di questo file è interamente a carico di Bastet.

Rendering del programma - xslc

Come descritto sopra, il rendering verso l'XHTML avviene ad opera del client, avendo preventivamente scaricato sia l'XML del programma che l'XML del file XSLT che lo trasformerà. La struttura dati che andiamo a trasformate è del tipo `block` o `block/chunk`, ed i contenuti e gli attributi dei nodi sono stati decisi dal *program type* associato.

Il programma trasformato gira all'interno di un oggetto *browser*, la sua visibilità e la sua sicurezza sono regolate dalle normali misure di sicurezza di una pagina web: non può accedere al *Chrome* del programma, e non ha altra visibilità che non l'oggetto *window* in cui gira (ovvero il box del *browser*).

Questa trasformazione avviene ad opera del client, gestita dal parser XSLT integrato in XULRunner (*"TransforMiiX"*, implementa correttamente XSLT v.1.0 standard sulle diverse piattaforme ed indipendentemente dalla piattaforma su cui gira il client). È Bastet a gestire il processo di trasformazione, popolando l'oggetto *browser* con la serializzazione del *document Object* che è il risultato dalla trasformazione stessa. Poiché è possibile che il risultato contenga XHTML passato dell'editor programmi in forma "escaped" (come "`<br />`" invece di "`<br />`"), dopo la trasformazione Bastet di occupa anche di effettuare un *unescaping* di tutti gli elementi del DOM risultante che abbiano attributo *id* o *class* dal valore *"toBeUnescapedAfterXform"*.

I parametri XSL di default

Al programma verranno inviati come parametri XSLT tutti i parametri del programma definiti in `/root/defaults/param` (del `prot`) e registrati in `/root/params/param` (nel `prog`).

Oltre a questi un set predefinito di parametri viene comunque valorizzato per default per ogni programma dal client, con valori di default che possono eventualmente essere sovrascritti da parametri omonimi.

In qualunque XSLT di `static/xlsc` per il rendering dei programmi è possibile dichiarare ed utilizzare alcuni parametri XSL disponibili di default, parametriche vengono valorizzati dal client ad ogni trasformazione (possono essere sovrascritti da parametri omonimi definiti nel programma):

- **`bastetServerURI`**
  Stringa. Viene valorizzato con l'URL «http://<*ip*>/<*bastet*>/»
- **`broadcastSystem`**
  Stringa. Il nome del sistema di brodcast di appartenenza, corrispondente al nome della directory web <*bastet*> definita nel client.
- **`channelID`**
  Intero. L'ID numerico del canale in trasmissione

- **`channelName`**
  Stringa. Il nome del canale in trasmissione
- **`networkID`**
  Intero. L'ID numerico del network a cui appartiene il canale
- **`networkName`**
  Stringa. Il nome del network a cui appartiene il canale
- **`panelName`**
  Stringa. Il nome con cui è stato registrato il pannello.
- **`platform`**
  Stringa. Vale "L" per Linux, "M" per Mac OS X, "W" per Windows.
- **`platformLong`**
  Stringa. Riporta il nome completo della piattaforma su cui gira il client, ad esempio "Win32", "Linux i686", "MacPPC", "MacIntel", o altro.
- **`platformName`**
  Stringa. Riporta il nome con cui il Sistema Operativo del client identifica sé stesso, ad es. "Windows NT 6.0", o "Intel Mac OS X 10.5".
- **`platformPlugins`**
  Stringa. Riporta un elenco delimitato da "|" (*pipe*) di tutti i plug-in per il web disponibili sulla piattaforma in uso, ad esempio: "`QuickTime Plug-in 7.6|Shockwave Flash|Shockwave for Director|VLC Multimedia Plug-in`".
- **`screenDepth`**
  Intero. Il numero di bit di colore di profondità dello schermo.
- **`screenOrient`**
  Stringa. Vale "H" o "V", *orizzontale* o *verticale*, dipendentemente dal rapporto altezza/larghezza dello schermo. Questa proprietà vale per tutto lo schermo, il singolo oggetto `window` del programma corrente potrà essere stretto e lungo (ad esempio un *header*) anche su un pannello disposto verticalmente
- **`screenHeight`**
  Intero. Il numero di pixel di altezza disponibile dello schermo.
  All'interno di ogni singolo browser l'altezza dell'oggetto `window` sarà determinata dalla proprietà javascript `window.availHeight`
- **`screenHeightMax`**
  Intero. Il numero di pixel di altezza totale dello schermo. Se il pannello è correttamente a pieno schermo questo valore coincide con `screenHeight`
- **`screenWidth`**
  Intero. Il numero di pixel di larghezza disponibile dello schermo.
  All'interno di ogni singolo browser la larghezza dell'oggetto `window` sarà determinata dalla proprietà javascript `window.availWidth`
- **`screenWidthMax`**
  Intero. Il numero di pixel di larghezza totale dello schermo. Se il pannello è correttamente a pieno schermo questo valore coincide con `screenWidth`

Ricaricare outt e/o prot

A seguito di uno di questi eventi:

- Variazione del nome di un *Outline Type*
- Variazione del nome di un *Program Type*
- Aggiunta di un *Outline Type*
- Aggiunta di un *Program Type*

si rende necessario ridefinire nel database i dati corrispondenti nella tabella `bttype`.

CGI di ri-caricamento dei Templates

A tale scopo è stata predisposta una CGI per uso prettamente sistemistico:

`http://<ip>/<bastet>/reloadTemplates.cgi?pass=BastetAdmin`

Eseguendo questa CGI tutti i record della tabella `bttype` vengono cancellati, e successivamente popolati estraendo i valori (ID e nome) dai corrispondenti file XML nelle directory `static/prot` e `static/outt`.

Alla fine del ricaricamento dei *program type* ed *outline type*, se lo stesso ha avuto successo, la CGI incrementa il *numero di lookup*.

Caveat

- È consigliabile editare il file `reloadTemplates.cgi` per modificare la *password* predefinita "BastetAdmin", onde evitare accessi indesiderati.
- Se uno dei file XML dei Template è broken il caricamento fallisce, e la tabella resta immutata (un errore viene visualizzato nel browser)
- Cancellare un *Outline Type* o un *Program Type* è una operazione estremamente rischiosa. Di norma non dovrebbe servire, ma se si decide di farlo occorre aver prima accuratamente verificato che l'elemento cancellato non sia utilizzato da alcun programma o outline.

Appendici

Appendice A - *quick reference*

URLs

- **http://**<*ip*>**/**<*bastet*>**/static/**
  Visualizzare in XML il contenuto della directory `static`
- **http://**<*ip*>**/**<*bastet*>**/feeder/feed.cgi**
  Trigger del caricamento dei feed dei programmi attivi
- **http://**<*ip*>**/**<*bastet*>**/feeder/feed1.cgi?prog=***n***
  Trigger del caricamento del feed di un programma con ID *n*
- **http://**<*ip*>**/**<*bastet*>**/log/**
  Interfaccia di consultazione dei log
- **http://**<*ip*>**/**<*bastet*>**/reloadTemplates.cgi?pass=***BastetAdmin***
  CGI per il ricaricamento dei templates (`prot` ed `outt`)

Visualizzare “static” sul web

- **h** o **help**
  Valore non falso (i.e.: `h=1` o `help=1`), visualizza l’aiuto
- **r** o **render**
  Valore non falso (i.e.: `r=1` o `render=1`), utilizza XSLT per il rendering in HTML
- **d** o **dirs**
  Valore non falso (i.e.: `d=1` o `dirs=1`), mostra le directory
- **p** o **pre**
  Stringa alfanumerica, filtra i file per prefisso
- **e** o **ext**
  Stringa alfanumerica, filtra i file per estensione
- **s** o **str**
  Stringa alfanumerica, filtra sia i file che le directory per nome

prot: /root/defaults/param/@*

- **name** (Obbligatorio)
  Stringa alfanumerica priva di spazi che inizia con una lettera, case sensitive: nome del parametro del `prog` (attributo `name`), e nome del parametro XSL
- **value** (Obbligatorio)
  Il valore di default del parametro, qualunque esso sia.
- **editable**
  Se esiste con valore “`yes`” allora il parametro sarà *editable*.
- **type** (Obbligatorio per i parametri *editable*)
  Tipo di interfaccia: “`choose`”, “`rgbcolor`”, “`text`”, “`textarea`” o “`html`”.

- **label**
  Alternativa al nome del parametro nella form di edit.
- **values** (obbligatorio per i valori *editable* di tipo `choose`)
  Una lista separata da *pipe* ("|") dei possibili valori del menu di scelta.
- **labels** (opzionale per i valori *editable* di tipo `choose`)
  Una lista separata da *pipe* ("|"), nomi alternativi a `values`.
- **tooltip**
  L'eventuale tooltip text (il callout al rollover) che si vuole associare al campo.

prot: /root/program/editor/@*

- **type** (Obbligatorio)
  I possibili valori sono quelli disponibili in `static/edit`, con bastet vengono forniti un editor di tipo "`compound`" ed un editor di tipo "`simple`".
- **program** (Obbligatorio)
  Il nome del program type come visualizzato nell'editor.

prot: /root/program/(block|chunk)/attribute/@*

- **name** (Obbligatorio)
  Stringa alfanumerica priva di spazi che inizia con una lettera, case sensitive.
- **type** (Obbligatorio)
  Tipo di interfaccia: "`choose`", "`rgbcolor`", "`text`", "`hidden`".
- **value** (solo per `type="hidden"`)
  Il valore dell'attributo, obbligatorio solo per attributi di tipo `hidden`.
- **label**
  Opzionale nome alternativo del parametro nella form di edit.
- **values** (obbligatorio, solo per `type="choose"`)
  Una lista separata da *pipe* ("|") dei possibili valori del menu di scelta.
- **labels** (opzionale, solo per `type="choose"`)
  Una lista separata da *pipe* ("|"), nomi alternativi a `values`.
- **tooltip**
  L'eventuale tooltip text (il callout al rollover) che si vuole associare al campo.

prot: /root/program/(block|chunk)/content/@*

- **type** (Obbligatorio)
  Tipo di interfaccia: "`choose`", "`rgbcolor`", "`text`", "`textarea`" o "`html`".
- **label**
  Opzionale nome alternativo del parametro nella form di edit.
- **values** (obbligatorio, solo per `type="choose"`)
  Una lista separata da *pipe* ("|") dei possibili valori del menu di scelta.
- **labels** (opzionale, solo per `type="choose"`)
  Una lista separata da *pipe* ("|"), nomi alternativi a `values`.

- **`tooltip`**
  L'eventuale tooltip text (il callout al rollover) che si vuole associare al campo.

prog/xslc: parametri XSL di default

- **`bastetServerURI`**
  Stringa. Viene valorizzato con l'URL «http://*<ip>*/*<bastet>*/»
- **`broadcastSystem`**
  Stringa. Il nome del sistema di brodcast di appartenenza.
- **`channelID`**
  Intero. L'ID numerico del canale in trasmissione
- **`channelName`**
  Stringa. Il nome del canale in trasmissione
- **`networkID`**
  Intero. L'ID numerico del network a cui appartiene il canale
- **`networkName`**
  Stringa. Il nome del network a cui appartiene il canale
- **`panelName`**
  Stringa. Il nome con cui è stato registrato il pannello.
- **`platform`**
  Stringa. Vale "L" per Linux, "M" per Mac OS X, "W" per Windows.
- **`platformLong`**
  Stringa. Nome completo della piattaforma su cui gira il client.
- **`platformName`**
  Stringa. Nome con cui il Sistema Operativo del client identifica sé stesso.
- **`platformPlugins`**
  Stringa. Riporta un elenco delimitato da "|" (*pipe*) dei i plug-in disponibili.
- **`screenDepth`**
  Intero. Il numero di bit di colore di profondità dello schermo.
- **`screenOrient`**
  Stringa. Vale "H" o "V", *orizzontale* o *verticale*.
- **`screenHeight`**
  Intero. Il numero di pixel di altezza disponibile dello schermo.
- **`screenHeightMax`**
  Intero. Il numero di pixel di altezza totale dello schermo.
- **`screenWidth`**
  Intero. Il numero di pixel di larghezza disponibile dello schermo.
- **`screenWidthMax`**
  Intero. Il numero di pixel di larghezza totale dello schermo.

Appendice B — Il software

Server: Moduli Perl “db”, “M”, “F” e “CMRA”

Questi moduli Perl sono basati su un core originario scritto da Marco Balestra dal 2000 al 2005, stati successivamente sviluppati e reingegnerizzati da Marco Balestra e Ferdinando Manzo per diversi progetti e prodotti tra il 2005 ed il 2009. Il modulo “CMRA” (Chimera) è lo sviluppo di un progetto del 2006 di Marco Balestra, realizzato nel 2008 e proposto come progetto pilota all’Università di Roma TRE nel corso dello stesso anno.

L’insieme di questi moduli costituisce il framework applicativo (con agganci anche nel client) su cui si basa lo sviluppo dei moduli Perl e delle CGI di Bastet.

Server: Moduli Perl “Bt”

Questi moduli sono stati sviluppati specificatamente per Bastet.

Client

Il client si basa anch’esso su un framework, costruito al di sopra del motore *XULRunner*.

Lo sviluppo è iniziato nel 2006 ad opera di M. Balestra per progetti freeware e prodotti conto terzi (come le edizioni del 2006 e del 2008 del “World Fisheries and Aquaculture ATLAS”, per conto della FAO of the UN – *Food and Agriculture Organization of the United Nations*).

Dal 2007 al 2009 M. Balestra e F. Manzo hanno riscritto ed organizzato un framework di librerie applicative nel client per l’interazione con il modulo “F” lato server, stabilendo un protocollo applicativo e verifica dati per web applications via AJAX, con ampio uso di codice XSLT ed oggetti x-javascript sviluppati in OOP.

Protocolli utilizzati

Le comunicazioni client/server avvengono tutte ed esclusivamente in `HTTP/1.1`

Su HTTP viene utilizzato un protocollo XML (AJAX) definito dai framework esposti.

Linguaggi utilizzati

Server: Perl, SQL, XML, XSLT, XHTML, CSS, javascript.

Client: XUL, CSS2, x-javascript, XML, XSLT, XHTML, javascript

Encoding utilizzato

Tutto il codice Perl, tutti i file XML ed i file XSLT e le transazioni AJAX sono in UTF-8.

Sviluppato interamente con computer Apple® Macintosh™

Ambiente di sviluppo: Apple Mac OS X, BareBones BBEdit.

Server Debian virtualizzati con "Vmware Fusion" su Mac OS X.

Marco Balestra e **Ferdinando Manzo**,

Roma, 23 Febbraio 2009.