Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Tool per la rendicontazione delle presenze dell'ICAR-CNR

Albino Altomare
Danilo Cistaro[1]
Antonio Scudiero[1]

RT-ICAR-CS-12-01 **Febbraio 2012**

1 Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR) - Sede di Cosenza, Via P. Bucci 41C, 87036 Rende, Italy, URL: www.icar.cnr.it

Tool per la rendicontazione delle presenze dell'ICAR-CNR

Albino Altomare* Danilo Cistaro† Antonio Scudiero‡

Sommario

In questo documento viene descritta l'architettura software dedicata alla gestione delle presenze dei dipendenti dell'ICAR-CNR. La rendicontazione delle ore che ogni singolo ricercatore ha dedicato ad uno specifico progetto rappresenta un delicato passaggio amministrativo e comporta spesso tempi relativamente lunghi per essere realizzata. A tal proposito, si è pensato di sviluppare un tool in grado di garantire strumenti adeguati per una rapida gestione dei rendiconti mensili ed un'immediata generazione dei corrispondenti timesheet direttamente nel formato desiderato.

1 Introduzione

La gestione del personale e la pianificazione delle attività rappresentano problematiche rilevanti per un'impresa, allo stesso tempo diventa fondamentale rappresentare l'organizzazione del lavoro riassumendola in documenti specifici e ben definiti. È estremamente importante, infatti, redigere documenti che delineino per ogni dipendente quelle che sono le attività lavorative svolte in un determinato arco temporale.

Questi documenti non sono solo basilari per determinare le ore lavorative e di conseguenza la retribuzione ma diventano fondamentali nel momento in cui ad esempio- si vuole effettuare un analisi degli obiettivi raggiunti dalla produzione con ima determinata configurazione del personale. Per rendere pi efficiente l'organizzazione del lavoro basta, per l'appunto, studiare lo storico dei dati di questa natura e prendere come modello l'organizzazione del personale che avrà dato risultati migliori. Alla luce di quanto appena riportato è possibile valutare l'effettivo peso che pu avere una dettagliata rendicontazione delle attività di un dipendente. È importante sottolineare come per avere una buona rendicontazione è necessario studiare a fondo la struttura aziendale per la quale si vogliono gestire tali informazioni, Quindi definire modelli di documenti ad hoc che permettano una rappresentazione della pianificazione del lavoro ottimale.

Tali principi generali valgono anche per pubblica amministrazione cos come per l'università e gli istituti di ricerca. Proprio per un istituto, l'ICAR-CNR, si è pensato di implementare un sistema in grado di preoccuparsi della rendicontazione delle presenze nello specifico le ore- dei ricercatori sui progetti.

Il sistema di rendicontazione in questione si presenta con un'interfaccia grafica intuitiva, dove all'utente vengono messe a disposizione le operazioni fondamentali di Inserimento e Modifica delle informazioni. Dalla stessa è possibile visualizzare informazioni riguardo l'allocazione delle ore per ogni ricercatore e gestire dati sensibili come l'elenco dei ricercatori e dei progetti con le relative attività. Nelle sezioni a seguire verranno presentate: il progetto POI Apache e la libreria che gestisce la rendicontazione.

2 Apache POI

Al fine di presentare in maniera esaustiva quanto realizzato si rende necessario descrivere brevemente la *libreria POI* sviluppata dal progetto Apache. Infatti uno dei package fondamentali del software si preoccupa di gestire documenti Microsoft Excel, a tale scopo è stata utilizzata la suddetta libreria.

L'*Apache Software Foundation* (ASF) fornisce un supporto organizzativo, legale e finanziario per una vasta gamma di progetti software open source. La fondazione definisce un quadro preciso sulla proprietà intellettuale e sui contributi finanziari in modo tale da garantire potenziali collaboratori da problemi di natura legale. Attraverso un proces-

*albino.altomare@gmail.com
†ICAR-CNR, cistaro@icar.cnr.it
‡ICAR-CNR, scudiero@icar.cnr.it

so di sviluppo collaborativo e meritocratico, i progetti Apache raggiungono un livello competitivo, prodotti software liberamente disponibili che attirano grandi comunità di utenti. Il tipo di licenza Apache garantisce la distribuzione dei prodotti Apache tra tutti gli utenti, siano essi partner commerciali o singoli individui.

Precedentemente noto come Apache Group, la Foundation è stata costituita nel 1999 a partire da un insieme di membri base, organizzazione senza fini di lucro al fine di garantire che i progetti Apache continuino ad esistere al di là della partecipazione dei singoli volontari. Gli individui che hanno dimostrato un impegno tangibile nello sviluppo di software open source, attraverso la partecipazione costante contribuendo ai progetti della Fondazione, sono ammissibili all'ASF.

L'obiettivo del progetto Apache POI [1] è quello di creare e mantenere API Java per manipolare vari formati di file basati sullo standard Office Open XML (OOXML) e Microsoft OLE 2 Compound Document Format (OLE2). In breve, queste API ti danno la possibilità di leggere e scrivere file MS Excel, MS Word e MS PowerPoint utilizzando Java. Infatti il formato OLE2 include la maggior parte dei file di Microsoft Office.

Il progetto POI fornisce le API sia per il file system OLE2 (POIFS) che per OLE2 Document Properties (HPFS). Per ogni applicazione MS Office esiste un modulo che tenta di fornire un livello di astrazione comune alle API Java sia ai formati OLE2 che OOXML. Questo approccio è stato pi sviluppato, al momento, per le cartelle di lavoro Excel (SS = HSSF + XSSF), sono in corso i lavori in tale direzione per i documenti Word (HWPF + XWPF) e per le presentazioni PowerPoint (HSLF + XSLF). Il progetto ha recentemente aggiunto il supporto per Outlook (HSMF).

2.1 API Java per gestire documenti Microsoft Excel

Il progetto POI Apache è composto da diversi moduli, in questa particolare sezione viene presentata la libreria che permette di gestire i documenti MS Excel. In particolare si individuano due API:

- HSSF è l'implementazione del progetto POI in Java della gestione dei file Excel versione '97(-2007);
- XSSF è l'implementazione del progetto POI in Java del formato Excel 2007 OOXML (.Xlsx).

HSSF e XSSF forniscono gli strumenti per creare, modificare, leggere e scrivere fogli di calcolo XLS. Essi forniscono:

- strutture di basso livello per chi ha particolari necessità;
- un'API eventmodel per un efficiente accesso in sola lettura;
- un'API completa usermodel per la creazione, la lettura e la modifica di file XLS.

Quando è necessario leggere i dati di un foglio di calcolo è sufficiente utilizzare l'API eventmodel sia del package *org.apache.poi.hssf.eventmodel* o del package *org.apache.poi.hssf.eventusermodel* a seconda del formato del file. Mentre quando si vogliono utilizzare i dati di un foglio di calcolo è necessario utilizare l'API usermodel. È inoltre possibile generare fogli di calcolo con questa libreria.

Si noti che il sistema usermodel comporta un ingombro di memoria superiore al eventusermodel di basso livello, ma il primo fa il grande vantaggio di essere uno strumento molto pi semplice con il quale lavorare.

Per iniziare ad usare in maniera rapida HSSF e XSSF per leggere e scrivere fogli di calcolo si rimanda alla guida [2]. Al contrario nel momento in cui si vuole una descrizione sull'utilizzo delle API HSSF e XSSF si consiglia di consultare la guida [3]. Per quel che riguarda una descrizione accurata di come tali librerie sono state utilizzate si rimanda alle sezioni successive.

3 Libreria rendicontazione presenze personale

L'architettura del sistema è delineata dall'implementazione di quattro meccanismi fondamentali:

- *parsing*;
- *scheduling*;
- *gestione documenti Microsoft Excel* con le API del progetto POI Apache mentre per quel che riguarda la gestione dei dati si è costruita una Struttura Dati ad hoc con il linguaggio Java.

Per iniziare a lavorare con il software dall'interfaccia utente sia che si voglia effettuare un inserimento di nuove ore, sia che si vogliano modificare quelle esistenti o pi semplicemente visualizzare le informazioni si devono specificare tre parametri fondamentali: *ricercatore*, *mese* e *anno*.

3.1 Parsing

La prima volta che una configurazione ricercatore/mese/anno- viene scelta entra in gioco il Parser che si preoccupa di andare a creare la struttura dati base necessaria per le operazioni di rendicontazione (Figura 1).

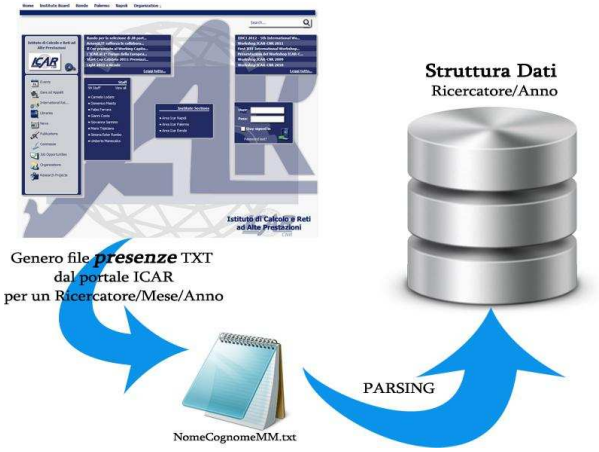

Figura 1: Schema della fase di Parsing.

Il *Parser* per operare necessità di un file testuale in input. Questo documento deve essere generato direttamente dal portale ICAR da utenti autorizzati, in particolare si deve selezionare il mese e l'anno del ricercatore del quale si vogliono rendicontare le ore di lavoro. Il documento pu essere salvato direttamente nella directory delle Presenze oppure in qualsiasi altro punto del file system, quest'ultima scelta comporta l'ulteriore passaggio di caricare il file da interfaccia utente. A questo punto *ParserText* legge il documento testuale, estrapola le informazioni per ogni giorno di quel mese e le converte in un formato ben definito: oggetti *DataDipendente*, ne esiste uno per ogni giorno di quel mese. La collezione di DataDipendente cosi creata sarà utilizzata dal metodo *inserisciRecordParse* per costruire la struttura dati che da qui a breve sarà definita. Alla fine di queste operazioni sono garantite le condizioni necessarie per procedere con le diverse fasi della rendicontazione.

3.2 Struttura dati

Durante la fase dell'analisi dei requisiti, tenuto conto delle esigenze dell'utente finale del programma, si è pensato di evitare l'implementazione di una base di dati classica. Si è optato per una gestione tramite file dati. Il software non fa altro che salvare la struttura dati Java in un file .dat nella directory StrutturaDati, in questo modo condividendo semplicemente la cartella di lavoro è stato possibile lavorare alla rendicontazione.

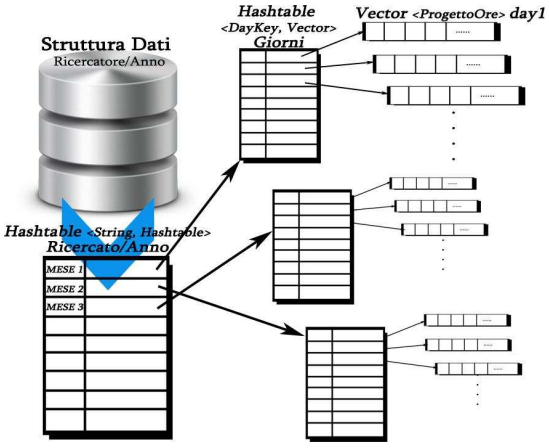

Figura 2: Struttura dati in Java.

Fin dalla fase di Parsing, come accennato precedentemente, viene creata la struttura dati Java nella quale vengono salvate le informazioni. Esiste una struttura dati per ogni Ricercatore/Anno, il file dati che verrà salvato si riferirà al lavoro svolto da un ricercatore per l'intero anno. Nei dettagli, come in Figura 2, viene creato un Hashtable Java che ha come chiave la tripla: ricercatore-mese-anno, mentre come valore ha un altro Hashtable. È evidente come nell'Hashtable Ricercatore/Anno si avranno 12 record, corrispondenti ai mesi presenti in un anno, quindi il mese rappresenta la chiave di lettura per questo particolare Hashtable, Ogni mese, come detto, individua un ulteriore Hashtable: Giorni.

Come è facilmente intuibile l'Hashtable Giorni avrà tanti record quanti giorni ha il mese a cui si riferisce, in particolare avremo come chiave un oggetto *DayKey* che oltre il giorno -per l'appunto-

mantiene altre informazioni importanti come il tipo di giornata (lavorativa, missione, malattia, ecc.). Il valore corrispondente ad ogni chiave DayKey - quindi per ogni giorno del mese- sarà una collezione di oggetti *ProgettoOre*, in cui si tiene traccia delle ore allocate per ogni singolo progetto.

3.3 Scheduling

La fase di *Scheduling* viene avviata ogni qual volta l'utente da interfaccia cerca di allocare le ore su un progetto/attività per una determinata configurazione di Ricercatore/Mese/Anno. L'operazione che viene effettuata quando entra in gioco lo *Scheduler* è quella dell'Inserimento.

La classe Scheduler in particolare attua una politica Round Robin, dove l'unità di tempo è stata fissata in 2 ore, in base alle ore di input specificate dall'utente si allocano le ore secondo questa unità assegnandole alle giornate lavorative pi libere. Per questo si intende che verranno allocate le ore in quei giorni che hanno al momento meno ore lavorative già assegnate. Nel momento in cui si cerca di inserire meno di 16 ore, lo Scheduler si preoccuperà di spalmare le ore in maniera bilanciata nell'arco di tutte le giornate lavorative del mese. Tutto ci è stato pensato e implementato sulla base di precise richieste manifestate in fase di progettazione preliminare e non dall'utente finale.

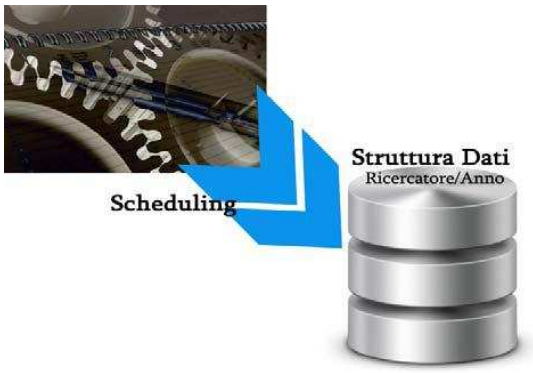

Figura 3: Scheduling.

In particolare lo Scheduler appena descritto viene invocato all'interno del metodo *aggiornaDati*, infatti la nuova allocazione delle ore una volta che da interfaccia viene confermata dall'utente sarà salvata nella struttura dati Java precedentemente descritta. Il corrispondente file dati di conseguenza viene modificato in maniera da garantire la consistenza delle informazioni.

3.4 Package xlsCore

Sicuramente il package *xlsCore* dove viene implementata la gestione dei documenti MS Excel attraverso le librerie del progetto POI Apache è da considerarsi la funzionalità pi importante in questo progetto per la rendicontazione.

Le classi che sono contenute in tale package entrano in gioco ogni qual volta l'utente conferma un'operazione di inserimento o modifica da interfaccia. Questo per garantire che il documento Excel corrispondente per quel Ricercatore/Mese/Anno sia creato/modificato in maniera consistente a quelle che sono le informazioni custodite nella struttura dati Java. Tutto ci garantisce un enorme risparmio di tempo, evitando che gli stessi vengano creati manualmente, come già ampiamente dimostrato dall'utilizzo di questo strumento durante le recenti operazioni di rendicontazione dell'istituto.

Il format dei time sheet è quello versione 2012 pervenuto direttamente dal MIUR, proprio per questo ogni volta che creo un nuovo documento Excel o vado ad aggiungere un nuovo foglio il programma fa riferimento costantemente a tale struttura standard. Nei dettagli quando si crea un nuovo documento il programma carica il modello direttamente dal file *formatTimeSheet.xls* presente nella directory principale del programma, prima di modificare l'unico foglio esistente in questo file vado a duplicare il foglio in modo da preservare lo schema di time sheet per le modifiche successive. Infatti nel momento in cui dovr inserire un nuovo foglio, quindi il documento esiste già, trover come ultima scheda sempre il format MIUR che prima di modificare, anche in questo caso, andr a duplicare.

I file Excel creati sono salvati nella directory Timesheet, per la precisione nella sottodirectory corrispondente all'anno di riferimento. Infatti i time sheet per ogni ricercatore sono annuali, al loro interno si individua una scheda per ogni mese/progetto che si è inserito.

Nel momento in cui l'utente conferma un'operazione tra le altre importanti operazioni come rendere effettive le modifiche confermate sul file dati viene creato un oggetto *TimesheetMIUR* in seguito meglio specificato- sarà questo oggetto ad

avviare le operazioni di gestione del documento Excel corrispondete. In particolare se il file Excel di riferimento non è ancora stato creato viene richiamato il metodo *inserisci Mese(...)* della classe *TimesheetMIUR*. Mentre se questo già esiste si lancia il metodo *aggiorna Mese(...)*. Specificato il momento esatto in cui vengono chiamate in causa le classi del package xlsCore, prima di procedere con una descrizione dettagliata delle stesse è bene far presente che è stata utilizzata la libreria *org.apache.poi.ss.usermodel*:

TimesheetMIUR questa classe si preoccupa di fornire le metodologie per interfacciare il software con i documenti Excel dando la possibilità di creare nuovi file, modificare e gestire quelli esistenti. I principali metodi che permettono questo sono:

- `inserisciMese(String ricercatore, String mese, String anno, Vector<ProgettoOre> attività, Hashtable<DayKey, Vector<ProgettoOre> > htOreMese)`: questo metodo inizializza il time sheet del mese anno specificato, in particolare andrà a creare un nuovo documento;
- `aggiornaMese(String ricercatore, String mese, String anno, Vector<ProgettoOre> attività, Hashtable<DayKey, Vector<ProgettoOre> > htOreMese)`: questo metodo aggiorna il documento Excel esistente, si preoccupa di aggiornare il mese specificato che è già stato creato precedentemente all'interno del file Excel di riferimento;
- `inserisciMeseNP(String ricercatore, String mese, String anno, int indTSmese[], Vector<ProgettoOre> attività, Hashtable<DayKey, Vector<ProgettoOre> > htOreMese)`: questo metodo crea una nuova scheda all'interno del documento, in questa scheda sarà inserito il time sheet di un mese/progetto perch per il mese in esame al ricercatore viene assegnato un progetto diverso da quelli già presenti.
- `verificaTimesheet(String ricercatore, String mese, String anno, Vetor<ProgettoOre> attività)`: questo metodo viene usato per fare un controllo della consistenza dei vari time sheet presenti nel file Excel corrispondente ai parametri specificati (Ricercatore/Anno).

ModelloXLSmiur questa classe definisce metodologie fondamentali che sono richiamate nella classe *TimesheetMIUR*, si avvale dei metodi di supporto della classe *OperazioniSheet*. Ad ogni buon fine si puntualizza che quando si parla di informazioni dinamiche si intendono quei dati del file .dat (struttura dati Java) che devono essere inseriti nel documento Excel, mentre le poche informazioni statiche sono quelle che vengono inserite solo la prima volta quando si crea la scheda e sono riconducibili ai parametri selezionati dall'utente. I principali metodi di questa classe sono:

- `creaTimeSheet(Workbook wb, int indSheet, Sheet sheet, String ricercatore, String mese, String anno, Vector<ProgettoOre> attività, Hashtable<DayKey, Vector<ProgettoOre> > htOreMese )`: questo metodo, richiamato dal metodo inserisciMese di TimesheetMIUR, si preoccupa di andare a inserire le informazioni della struttura dati e le formule. In particolare gestisce direttamente le modifiche delle informazioni pi semplici, dette fisse, e per quelle denominate dinamiche utilizza il metodo timesheetDinamico() sempre in questa classe.
- `aggiornaTimeSheet(Sheet sheet, String mese, String anno, Sheet[] mese Progetti, Vector<ProgettoOre> attività, Hashtable<DayKey, Vector<ProgettoOre> > htOreMese)`: questo metodo, richiamato dal metodo aggiornaMese di TimesheetMIUR quando la scheda del mese/progetto esiste già. In questo caso non modifico le informazioni fisse ma richiamo timesheet-

DinamicoNP(...) per aggiornare quelle dinamiche.

- `timesheetDinamico(OperazioniSheet opS, String[] dOW, Vector<ProgettoOre> attività, Hashtable<DayKey, Vector<ProgettoOre> > htOreMese)`: questo metodo completa il time sheet del mese con le informazioni dinamiche.
- `creaTimeSheetNP(Workbook wb, int indSheet, Sheet sheet, Sheet[] mese Progetti, String ricercatore, String mese, String anno, Vector<ProgettoOre> attività, Hashtable<DayKey, Vector<ProgettoOre> > htOreMese)`: questo metodo, richiamato dal metodo inserisciMeseNP di TimesheetMIUR, si preoccupa di andare a inserire le informazioni della struttura dati e le formule. In particolare gestisce direttamente le modifiche delle informazioni pi semplici, dette fisse, e per quelle denominate dinamiche utilizza il metodo timesheetDinamicoNP(...) sempre in questa classe. La particolarità in questo caso è che vanno gestiti i riferimenti tra i diversi fogli di calcolo per tener presente le ore assegnate agli altri progetti nello stesso mese. La lista di Sheet mese Progetti contiene appunto i fogli di altri progetti che si riferiscono al mese in oggetto dell'aggiornamento.
- `timesheetDinamico(OperazioniSheet opS, String[] dOW, Vector<ProgettoOre> attività, Hashtable<DayKey, Vector<ProgettoOre> > htOreMese)`: questo metodo completa il time sheet del mese con le informazioni dinamiche.
- `timesheetDinamicoNP(OperazioniSheet opS, String[] dOW, Vector<ProgettoOre> attività, Hashtable<DayKey, Vector<ProgettoOre> > htOreMese)`: questo metodo completa il time sheet del mese con le informazioni dinamiche, rispetto al metodo timesheetDinamico() questa versione si preoccupa della gestione dei riferimenti tra i diversi time sheet di uno stesso mese ma di progetti diversi.

OperazioniSheet: questa classe definisce tutti i metodi di utilità utilizzati dalla classe ModelloXLSmiur. I metodi di questa classe si preoccupa di dare la possibilità di accedere direttamente ad una singola Cella del foglio Excel per modificarla, il funzionamento di questi metodi sono molto intuitivi e si evita di riportarne una spiegazione dettagliata.

Riferimenti bibliografici

[1] *http://poi.apache.org*

[2] *http://poi.apache.org/spreadsheet/quick-guide.html*

[3] *http://poi.apache.org/spreadsheet/how-to.html*