

Appendici

A. Prodotti e tecnologie utilizzate

BEA

BEA RFID Edge Server 2.1

Questo software è installato nei siti dove ci sono i lettori RFID, come magazzini, centri di distribuzione e magazzini di vendita al dettaglio, per registrare dati dei in arrivo e riportare quei dati all'Enterprise Server). L'Edge Server può essere eseguito in modalità stand-alone su un computer o essere integrato in una serie crescente di altri dispositivi, compresi i routers.

WebLogic RFID Edge Server fornisce numerose operazioni che aiutano ad adattare la soluzione RFID a particolari esigenze di business:

- Possibilità di connessione a diversi dispositivi RFID tra cui lettori e stampanti.
- Supporto per il filtraggio di tag e dati sfruttando le funzionalità degli Application Level Events.
- (ALE) API, comandi standard creati da BEA e approvati da EPCglobal.
- Supporto per monitorare e configurare dispositivi RFID attraverso la supply chain.
- Supporto per la configurazione di workflow che automatizzano processi di business relativi al magazzino, centro di distribuzione o livello di magazzino.
- Un'architettura leggera e distribuita con esigenze hardware e di rete minime.

WebLogic RFID Edge Server fornisce un'infrastruttura software completa per sviluppare e installare applicazioni RFID affidabili. La tecnologia dell'Edge Server permette agli sviluppatori di scrivere applicazioni ad un sufficientemente alto livello, mentre il RFID Edge Server gestisce tutte le interazioni di basso livello con molti tipi di lettori e assicura che la rete di readers e software funzioni alla massima efficienza.

Il RFID Edge Server fornisce sostegno per processi che sono cruciali a qualsiasi livello di utilizzo delle tecnologia RFID in un'architettura enterprise:

- Filtraggio e integrazione di dati;
- Gestione dell'infrastruttura RFID;
- Workflows locali.

Filtraggio e integrazione di dati

I lettori RFID generano un flusso di informazioni molto dettagliate. Ad esempio, un lettore RFID posto in un ambiente tipico di ripiano di magazzino riferisce la presenza di una particolare etichetta con una frequenza di cinque volte al secondo. La maggior parte delle applicazioni commerciali non possono occuparsi efficacemente di tali informazioni così dettagliate; essi sono impostati a ricevere rapporti di alto livello tipo "negli ultimi 60 secondi, i seguenti tags sono state aggiunte al ripiano".

WebLogic RFID Edge Server fornisce una semplice ed ancora più potente interfaccia di programmazione (API) attraverso cui le applicazioni possono definire eventi di così alto livello e riceverli in una varietà di formati progettati per una facile integrazione con il software aziendale. Le applicazioni definiscono a quali eventi di livello applicazione sono interessati e il RFID Edge Server soddisfa quelle richieste interagendo con i lettori RFID ed eseguendo filtri, contatori e raggruppamenti di dati.

Inoltre, WebLogic RFID Edge Server nasconde alle applicazioni i dettagli su quanti lettori sono utilizzati in una data posizione, la loro fabbricazione o il loro modello e come essi sono configurati.

Gestione dell'infrastruttura RFID

Di solito le applicazioni RFID in scala reale implicano centinaia a centinaia di migliaia di lettori RFID, schierati in dozzine di migliaia di posti remoti. La maggior parte dei lettori sono installati in posti che hanno un supporto di personale IT limitato o assente. Inoltre, i lettori RFID operano autonomamente, generando dati in processi di business aziendali senza intervento umano o controllo. Affinché i processi possano fare affidamento su questi dati, là devono avere la garanzia che l'infrastruttura è sana.

Ad esempio, fa differenza se nessuna etichetta RFID è letta alla postazione di carico #5 perché non è ricevuta alcuna spedizione o perché un forklift errante ha danneggiato l'antenna del lettore. WebLogic RFID Edge Server fornisce uno strumento completo di monitoraggio e gestione da remoto dell'infrastruttura RFID. Attraverso le strutture di monitoraggio e di gestione del server, il personale addetto può controllare centralmente lo stato di salute e il funzionamento di lettori remoti RFID e dell'hardware per computer associato. Il server può anche collegare queste informazioni sul monitoraggio col

sistema esistente di gestione dell'infrastruttura aziendale e apposite applicazioni di servizio. WebLogic RFID Edge Server fornisce anche un punto centrale per definire e amministrare e configurare i dispositivi RFID, compreso impostare parametri specifici del modello regolati solitamente per ottimizzare le operazioni.

Flussi di lavoro locali

Arricchire eventi tecnici con informazioni di business correlate è fatto al livello del modulo di workflow. Un flusso di lavoro è una serie di azioni innescate da un evento esterno, come l'osservazione di un'etichetta RFID. Ad esempio, l'osservazione di un'etichetta RFID su un bagaglio che viaggia su un nastro trasportatore può incrementare un contatore LED, accende una luce verde, inviare un evento EPC Information Service (EPCIS) o inviare una notifica a un altro sistema. Uno qualunque o tutti questi eventi collegati al flusso di lavoro possono essere innescati dall'osservazione dell'etichetta. In casi più complessi, possono essere innescate più azioni dello stesso tipo (ad esempio, gli eventi o le notifiche possono essere inviati a più destinazioni).

I flussi di lavoro sono costruiti da moduli, che prendono messaggi in input (report di cicli di eventi o output di altri moduli) ed emettono messaggi di workflow secondo la logica di business integrata. Ci sono due tipi di moduli workflow:

- moduli per la logica di business (Directional Portal o Observe Portal) che rappresentano i processi aziendali;
- moduli helper (Stack Light) che i moduli di business logic richiamano per controllare hardware o inviare messaggi ad altri sistemi.

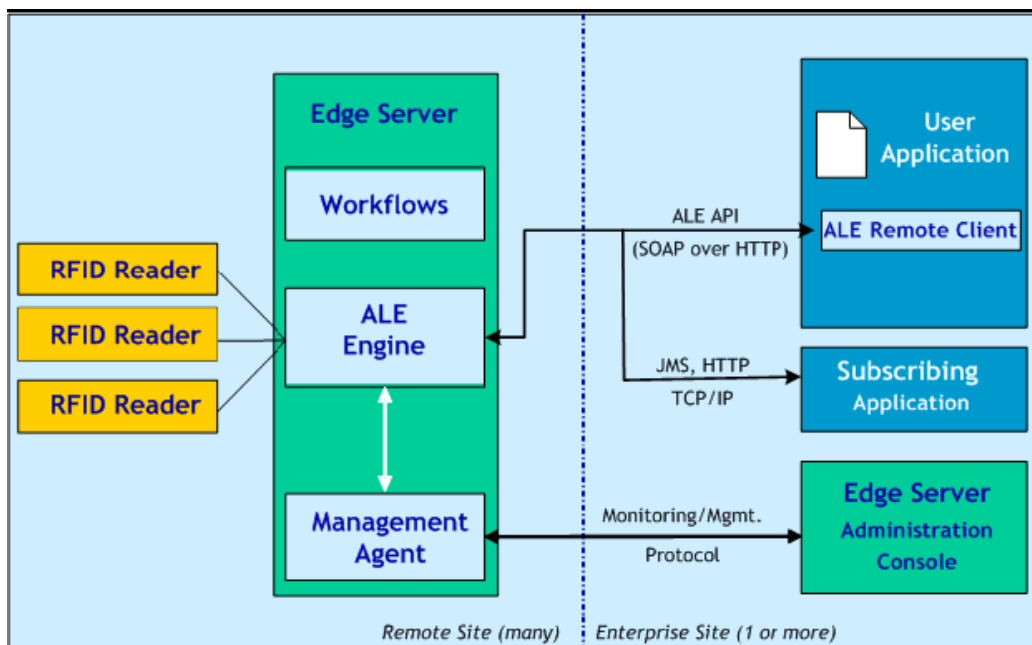


Figura A-1 Architettura di RFID Edge Server

BEA WebLogic 9.2

BEA WebLogic Server è un'application server Java 2 Enterprise Edition (J2EE) scalabile. L'infrastruttura di WebLogic supporta l'installazione di molti tipi di applicazioni distribuite ed è la base ideale per costruire applicazioni basate su architetture service-oriented. SOA è una metodologia di design orientata a massimizzare il riuso di servizi applicativi.

WebLogic implementa completamente le specifiche Sun Microsystems J2EE 1.4 che offrono un set standard di API per creare applicazioni Java distribuite che possono accedere ad una estesa varietà di servizi, come database, servizi di messaging e connessioni a sistemi enterprise esterni. Gli utenti finali accedono a queste applicazioni usando il web browser o client Java.

Oltre alla realizzazione di J2EE, WebLogic Server abilita le aziende ad installare applicazioni mission-critical in un ambiente robusto, sicuro, affidabile e scalabile. Queste caratteristiche permettono alle aziende di configurare clusters di istanze di server per distribuire il carico e fornire capacità aggiuntive in caso di guasti hardware o di altro genere. Nuovi strumenti diagnostici permettono agli amministratori di sistema di monitorare e mettere a punto le performance delle applicazioni installate e l'ambiente

del server stesso. È possibile configurare il server anche per monitorare e configurare la produttività delle applicazioni senza alcun intervento umano. Caratteristiche di sicurezza estese proteggono l'accesso ai servizi, mantengono i dati aziendali sicuri e prevengono attacchi informatici.

Modelli di programmazione

WebLogic Server fornisce un supporto completo per le specifiche J2EE 1.4. Di seguito sono elencati i modelli di programmazione supportati:

- Le *Web Applications* forniscono il meccanismo base J2EE per installare un sito web con pagine dinamiche seguendo gli standard J2EE delle Servlets e Java Server Pages (JSP). Le applicazioni web sono usate anche per servire contenuti web statici come pagine HTML e immagini.
- I *Web Services* forniscono un insieme condiviso di funzioni disponibili ad altri sistemi in rete e possono essere usati come componenti di applicazioni web distribuite.
- Le potenzialità *XML* includono attività di scambio di dati, memorizzazione di contenuti indipendentemente dalla loro presentazione e altro ancora.
- *Java Messaging Service (JMS)* abilita le applicazioni a comunicare tra loro attraverso lo scambio di messaggi. Un messaggio è una richiesta, un report e/o eventi che contengono informazioni necessarie a coordinare la comunicazione tra applicazioni differenti.
- *Java Database Connectivity (JDBC)* fornisce un accesso alle risorse di un DBMS.
- *Resource Adapters* fornisce connettività a sistemi legacy e ad altri sistemi enterprise esterni.
- *Enterprise JavaBeans (EJB)* forniscono oggetti Java che incapsulano dati e logica business.
- *Remote Method Invocation (RMI)* è lo standard Java per la programmazione con oggetti distribuiti, permettendo alle applicazioni di invocare localmente metodi di un oggetto remoto.

- *Beehive* è un framework di programmazione J2EE open-source studiato per rendere più semplici difficili operazioni di programmazione J2EE usando annotazioni Java che contengono metadati.
- *Security APIs* permettono di integrare autenticazioni e autorizzazioni in applicazioni J2EE. È possibile inoltre usare le *Security Provider APIs* per creare un gestore della sicurezza personalizzato.
- *WebLogic Tuxedo Connectivity* (WTC) fornisce interoperabilità tra applicazioni su WebLogic Server e servizi su BEA Tuxedo. WTC permette a client sul server di invocare servizi e a client Tuxedo di invocare EJBin risposta a una richiesta di un servizio.

Amministrazione del sistema

L'ambiente di amministrazione di WebLogic Server include operazioni come la creazione di domini, l'installazione di applicazioni, la migrazione di domini da ambienti di sviluppo ad ambienti di produzione, monitoraggio e configurazione delle performance del dominio e diagnosi e risoluzione di problemi. WebLogic Server mette a disposizione molti strumenti per gli amministratori di sistema per aiutarli con queste operazioni:

- Console di amministrazione web-based;
- *WebLogic Scripting Tool* (WLST), un linguaggio di scripting per l'automatizzazione di operazioni di sistema basate su Jython;
- *SNMP*;
- Uno wizard per la configurazione
- Utilities a riga di comando.

Poichè la gestione di WebLogic Server è basata su standard J2EE e non solo, il sistema di amministrazione si integra bene con altri strumenti usati per gestire altri software enterprise e componenti hardware. Inoltre WebLogic Server implementa le specifiche J2EE *Java Management Extension* (JMX), che permettono un accesso per la programmazione al sistema di gestione.

Usando queste API si possono creare proprie utilities di amministrazione o automatizzare operazioni ricorrenti usando classi Java.

BEA AquaLogic Service Bus 2.5

Mentre le aziende si sforzano di essere snelle, dipende dalla capacità delle tecnologie informatiche di offrire nuovi servizi e riutilizzare quelli attuali alla velocità del business. Questo desiderio di essere guidati dai servizi sta generando una tendenza significativa affinché si adotti l'architettura orientata ai servizi (SOA). Per rendere SOA una realtà, l'IT richiede un'infrastruttura intelligente di servizio che guida e facilita il riuso dei servizi e fornisce un'integrazione affidabile attraverso uno scenario informatico eterogeneo e multi-vendor. BEA AquaLogic Service Bus unisce l'intermediazione intelligente di messaggi con il controllo e la gestione di servizio per fornire un prodotto software unico per implementare e installare un'architettura orientata ai servizi. Questo approccio aggiunge un livello di routing dinamico e trasformazione scalabile all'infrastruttura aziendale, in più offre la possibilità di gestione del ciclo di vita dei servizi, della loro registrazione, del loro uso e di Service Level Agreement (SLA). AquaLogic Service Bus è il cuore della soluzione completa di BEA per l'integrazione di business.

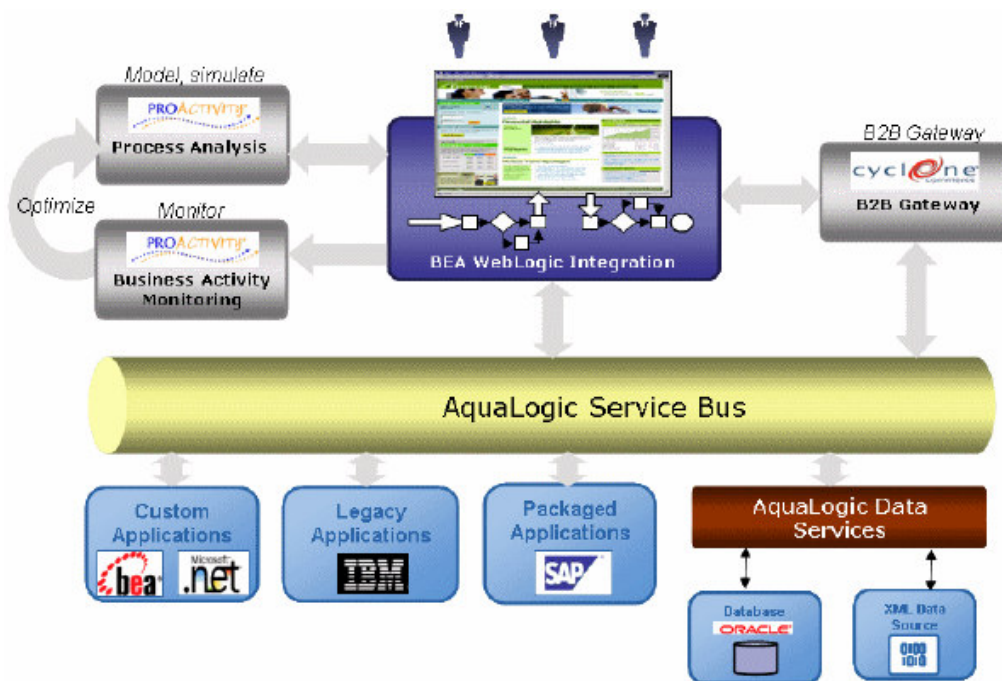


Figura A-2 BEA's Comprehensive Integration Solution

Il service bus elimina i problemi di questo movimento scomposto verso i servizi che sfida le imprese e i loro dipartimenti IT. Le sfide includono i collegamenti hard-wired

fra le applicazioni ed i servizi che danno vita ad una integrazione complessa, rigida e strettamente accoppiata. A sua volta, questo provoca una alterata capacità di riutilizzare i servizi e le sfide nel controllo di servizi installati. Tutto questo provoca un alto TCO (Total Cost of Ownership) per l'impresa.

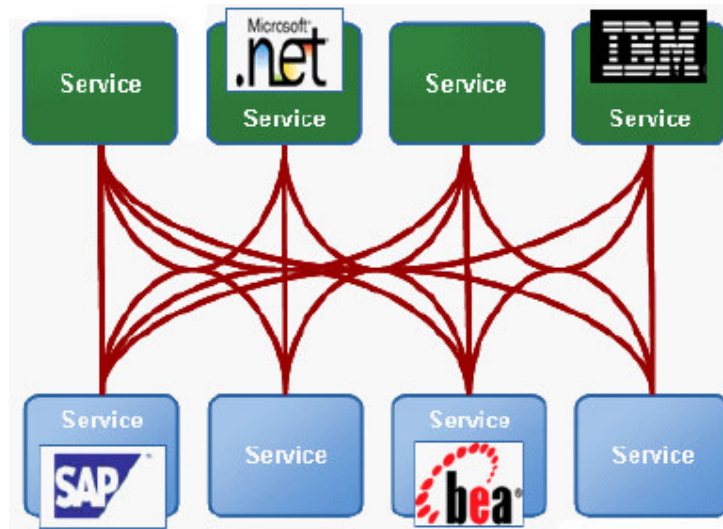


Figura A-3 The Service Sprawl Challenge

Le sfide specifiche affrontate dagli architetti informatici responsabili del messaging e delle iniziative SOA nell'attuale azienda includono:

- Introduzione di comportamenti dinamici con capacità di configurazione a runtime in un sistema.
- Riutilizzo di servizi che sono sviluppati in azienda e gestione del loro ciclo di vita
- Assicurazione sull'uso costante di servizi
- Assicurazione che i servizi siano sicuri
- Assicurazione che i servizi aderiscano alle politiche IT
- Verifica e controllo sull'uso dei servizi e gestione dei guasti del sistema

In breve, l'obiettivo dell'architetto applicativo e degli altri esperti è di riutilizzare, migliorare ed mantenere il controllo dell'infrastruttura IT. AquaLogic Service Bus è modellato per aiutare a raggiungere questi obiettivi.

Il service bus fornisce una ricca console di comandi per la configurazione dinamica delle politiche e dei servizi, così come per il controllo del sistema. Il bus consente di

avere un'architettura localmente ad accoppiamento lasco, facilita il riuso dei servizi e centralizza l'amministrazione. Tutto ciò permette di migliorare il TCO. La console permette di rispondere velocemente ed efficacemente ai cambiamenti dell'ambiente service-oriented.

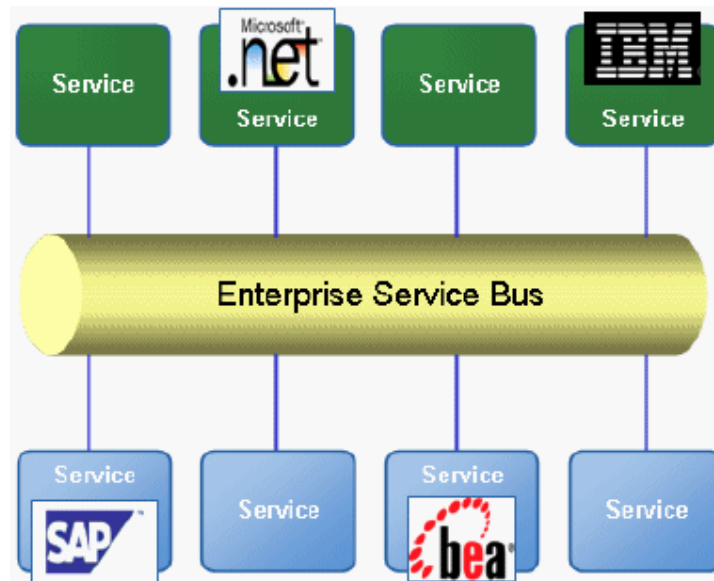


Figura A-4 Eliminating Service Sprawl

AquaLogic Service Bus si appoggia sull'affidabilità di WebLogic Server e fa leva sulle capacità del server di fornire funzionalità altamente disponibili, scalabili e affidabili.

Il bus di BEA è un Enterprise Service Bus (ESB) specificamente disegnato per l'integrazione di servizi, il controllo di Web Services e la consegna di messaggi tradizionali intermediando tra contesti IT eterogenei. L'architettura leggera, stateless e performante del service bus costituisce un elemento chiave per la rete di servizi distribuiti.

Con AquaLogic, si possono implementare dinamicamente integrazioni di servizi attraverso la configurazione di policy e proxy service. Questo approccio permette alle architetture di evolversi velocemente in concomitanza con le seguenti caratteristiche del sistema:

- Sicurezza
- Service location
- Disponibilità

- Formato dei dati
- Logging e monitoraggio
- Protocolli di trasporto e paradigmi di comunicazione

Grazie alla natura intermediaria dei proxy service, si può utilizzare il bus per risolvere le differenze fra il i servizi client ed i requisiti dei servizi di business nelle seguenti aree:

- Contenuto dei payload e schemi
- Protocolli di imbustamento
- Protocolli di trasporto
- Protocolli di pubblicazione e sottoscrizione
- Paradigmi one way e richiesta/risposta
- Comunicazione sincrona e asincrona
- Conformità della sicurezza

AquaLogic Service Bus mantiene le configurazioni delle risorse, dei policy e dei proxy service in meta dati che possono essere facilmente trasportati dall'ambiente di sviluppo a quello di produzione. Il motore del bus accede a queste informazioni di configurazione dal repository dei meta dati.

L'idea chiave del service bus di BEA è la separazione fisica delle funzioni di amministrazione dai servizi implementati. Questa separazione permette che le implementazioni evolvano indipendentemente e dinamicamente guidate dalle necessità di business senza richiedere sforzi costosi di sviluppo dell'infrastruttura. Il service bus può essere utilizzato orizzontalmente attraverso molti applicazioni e sistemi, abbracciando servizi implementati in differenti dipartimenti aziendali e sviluppati da squadre diverse.

AquaLogic Service Bus Architecture

AquaLogic Service Bus è un mediatore che prende i messaggi, li processa per determinare dove dirigerli e li trasforma come specificato. Riceve i messaggi su un protocollo di trasporto tra HTTP(S), JMS, File, FTP e così via, ed invia i messaggi sullo stesso o su un differente protocollo. La risposta del messaggio segue il percorso inverso.

L'elaborazione del messaggio è guidata dai meta dati contenenti il flusso del messaggio definito per un proxy service sulla console.



Figura A-5 I servizi consumatore e produttore interagiscono su AquaLogic

AquaLogic permette di stabilire un accoppiamento lasco fra i servizi client ed i servizi di business ma rimane un punto centralizzato per la gestione della sicurezza e il monitoraggio. La seguente figura mostra l'architettura ad alto livello, che fa vedere il bus composto dai seguenti sottosistemi:

- Service management
- Message brokering
- Configuration framework
- Security
- Protocolli di messaging e di trasporto.

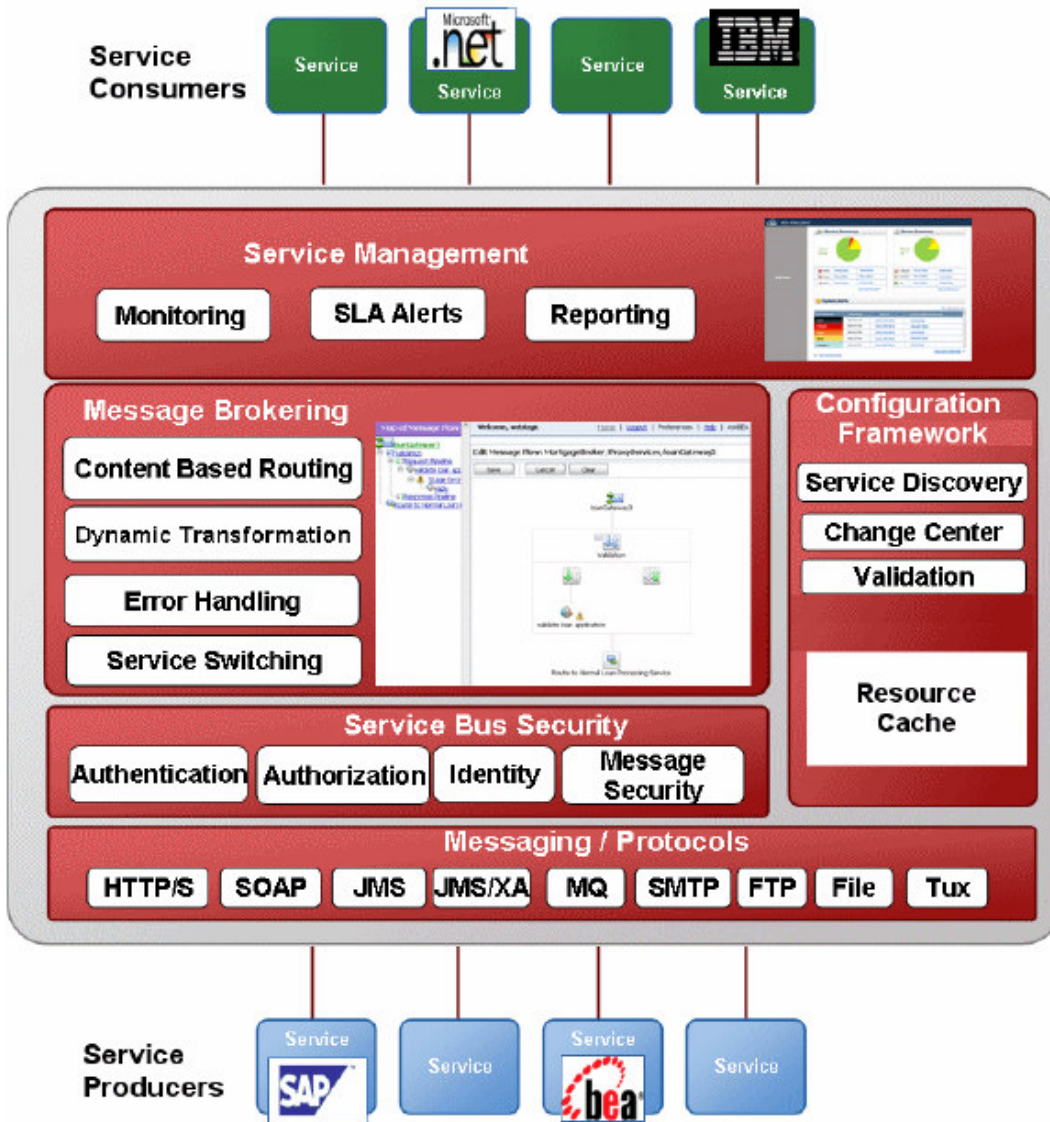


Figura A-6 Architettura di alto livello di AquaLogic Service Bus

ORACLE APPLICATION 11i

I SISTEMI DI TIPO “ENTERPRISE RESOURCE PLANNING” (ERP)

L'elemento determinante per realizzare con successo un progetto di innovazione deve essere la revisione dei processi interni ed esterni all'organizzazione attraverso la consulenza e il supporto di partner tecnologici affidabili e in grado di offrire soluzioni complete.

In particolare Borgonovi, direttore della SDA Bocconi di Milano, sostiene che “il controllo di gestione consente, attraverso una serie di dati, di valutare i risultati della gestione di un'Azienda in modo integrato. I sistemi ERP per il controllo di gestione permettono di collegare direttamente l'obiettivo di una determinata attività alle risorse impiegate, ma per farlo devono essere completi, integrati e soprattutto flessibili.

Il sistema ERP vincente è quello che riesce a mediare tra integrazione dell'informazione e flessibilità, mentre i settori dove risulta più necessario sono quelli che trattano un'elevata mole di informazioni, per i quali diventa importante poter inserire il dato una sola volta e utilizzarlo a cascata su diversi fronti”.

Definizione di sistema gestionale ERP

L'acronimo ERP, che sta per Enterprise Resource Planning, è stato introdotto per la prima volta negli anni '80 per identificare i primi applicativi gestionali aziendali che avevano la caratteristica, fondamentale ed innovativa per quel periodo, di integrazione tra le applicazioni di Contabilità e quelle di Gestione della Logistica e/o della produzione Manifatturiera.

Dati gli evidenti benefici apportati dai primi sistemi ERP, con il diffondersi di questa filosofia, le maggiori aziende produttrici di software applicativo a livello mondiale si sono dedicate sin dai primi anni '90 alla produzione, commercializzazione ed implementazione di sistemi sempre più integrati, completi, e configurabili in modo da potersi adattare non solo ai requisiti delle aziende Manifatturiere, ma anche a quelle della Distribuzione, dei Servizi, alle Banche ed Assicurazioni, e specialmente nell'ultimo quinquennio anche alle Aziende e/o Enti della Pubblica Amministrazione.

Oggi l'acronimo ERP viene usato anche da produttori di software più o meno locali per identificare qualsiasi soluzione gestionale che abbia una o più aree applicative integrate, ma in realtà sono i fatti (l'orientamento delle aziende acquirenti), il notevole grado di diffusione, gli aspetti tecnologici e le cosiddette "Quote di Mercato" a dettare l'attuale definizione di sistema ERP (o extended ERP, per identificare che l'integrazione applicativa viene estesa lungo la catena del valore, anche a Fornitori, Clienti/Utenti, Partner Commerciali, ecc.).

L'ERP è un sistema applicativo gestionale completo, integrato, orientato ai processi e non alle singole funzioni, basato sulle "Best Practices", precostituito ma configurabile, concepito in modo da poter essere implementato in tutti i settori aziendali, sviluppato con tecnologia software allo stato dell'arte dai principali produttori di software a livello mondiale.

Infatti solo un numero limitato di aziende a livello mondiale possono oggi realmente permettersi una capacità finanziaria, un volume di investimenti ed un bacino di clientela atti a garantire la continua evoluzione tecnologica, l'arricchimento funzionale ed una longevità dei prodotti, caratteristiche che attualmente il mercato delle soluzioni applicative richiede per essere riconosciute come "standard de facto".

Il modello tecnologico

Il modello tecnologico su cui è basato il Sistema Informativo per le Aziende o Enti Pubblici che Oracle propone è noto come "three-tier architecture" (architettura a tre livelli), il quale esprime un'architettura "Internet based" omogenea, basata su componenti tecnologiche distribuite e scalabili su tre livelli elaborativi distinti:

- il livello "Presentazione" con interfaccia utente grafica evoluta GUI operante su piattaforma client che per l'ambiente Internet richiede la presenza del solo componente *web-browser*
- il livello "Applicazione" dove è operativa la logica applicativa, posizionato su una macchina server denominata appunto Application Server

- il livello “Dati” dove opera il gestore di database posizionato su un’altra macchina server denominato in questo caso Database Server.

Tale modello prevede che il *client* (la stazione di lavoro) si faccia carico della presentazione dell’interfaccia utente, che l’*application server* si faccia carico della logica elaborativa dell’applicazione e che il *data base server* provveda a fornire i servizi di accesso ai dati, cioè in altri termini a sostenere il *relational data base management system*.

Il modello architetturale a 3-livelli si è affermato negli ambienti applicativi integrati e distribuiti, come evoluzione e superamento dei limiti del modello client/server a due livelli e del semplice browser/web statico. Distribuendo i compiti applicativi secondo il modello a 3-livelli, si ottengono diversi vantaggi:

- disponibilità generalizzata dei dati alle diverse istanze applicative attuali e future
- scalabilità e controllo prestazionale e funzionale ottimale delle applicazioni
- possibilità intrinseca di definire il livello più appropriato di disponibilità, affidabilità e sicurezza di ogni singolo modulo
- controllo generalizzato in rete dei servizi da una singola postazione (parametri HW e SW dei sistemi, controllo del livello di affidabilità, automazione dei backup, ecc...)

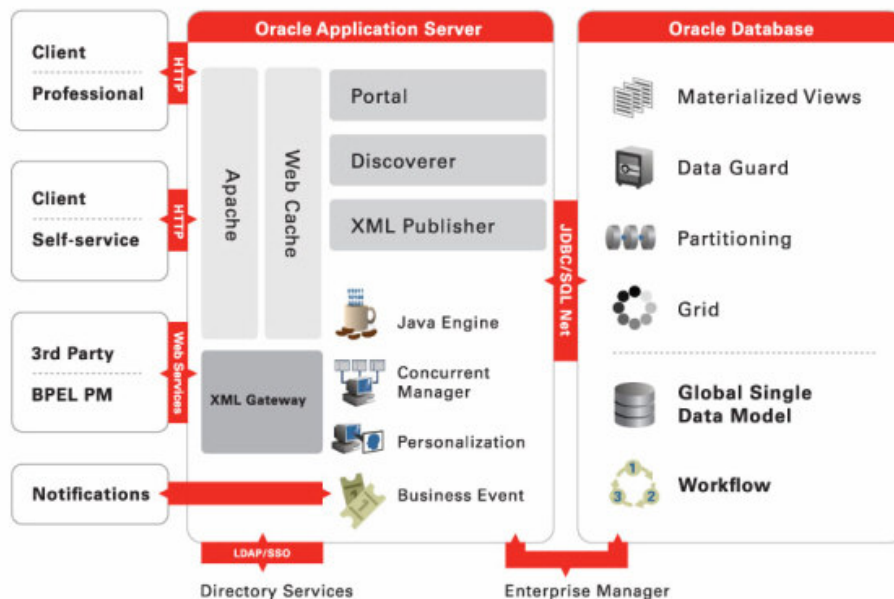


Figura A-7 Architettura di alto livello di Oracle Application

Caratteristiche generali e peculiari di Oracle eBusiness Suite

Oracle e-Business Suite è una tra le soluzioni software gestionali integrate (chiamate anche comunemente E.R.P.) più avanzate attualmente presenti sul mercato nazionale (circa 200 installazioni) ed internazionale (più di 13.000 installazioni). Oracle e-Business Suite è certificata su sistema operativo Unix, Linux, Windows NT, Windows 2000 e Windows 2003 e la totalità delle funzioni è fruibile utilizzando un “Browser Internet/Intranet” standard.

Quest’ultima rilevante caratteristica permette di abbattere a zero i costi interni di distribuzione ed allineamento del software sui Personal Computer che fungono da terminale per l’utente finale, eliminando uno dei principali problemi delle soluzioni in architettura “client-server” tradizionale.

In estrema sintesi, le caratteristiche peculiari di Oracle e-Business Suite sono le seguenti:

- È un sistema gestionale *completo, modulare, integrato*
- È una soluzione di facile utilizzo per gli utenti, dotato di aiuto in linea e funzionalità “Self Service” auto-guidate
- È una soluzione di tipo *orizzontale*, utilizzabile su tipologie di aziende diverse, data l’elevata flessibilità che lo contraddistingue
- È sviluppato con *tecnologie molto avanzate, mantenuto costantemente allo stato dell’arte*
- È un sistema orientato ai *processi aziendali* (non solo un pacchetto gestionale costituito da un insieme di funzionalità, ma un’architettura per la gestione dei processi di business).

Oracle e-Business Suite è una soluzione completa, interamente costruita sullo “stack tecnologico” di un unico fornitore che è Oracle, a copertura delle esigenze di tutte le aree funzionali di una azienda (Gestione Risorse Umane, Amministrazione, Finanza e Controllo, Business Intelligence, Gestione Progetti e Commesse, Gestione delle Relazioni con i Clienti e Marketing, Catena Logistica e Manutenzioni, Produzione,...), ed è corredata da una serie di tools e strumenti per l’amministrazione di sistema e la produzione parametrica di reportistica, che rendono flessibile l’uso del sistema.



Figura A-8 Moduli Oracle e-Business Suite

La soluzione e-Business suite dal punto di vista dell'Architettura Applicativa comporta notevoli vantaggi.

Prima fra tutti la **scalabilità**, grazie alla quale si hanno:

- la disponibilità in tempi ridotti di una soluzione che copra le esigenze di start-up, grazie alla presenza di un modello implementativo per le diverse Aziende
- un approccio dai costi contenuti
- la flessibilità nell'uso del sistema e quindi, partendo da una modalità analoga all'attuale organizzazione, la possibilità di adeguare le procedure secondo l'evolversi dei modelli organizzativi
- la gradualità in quanto tutte le funzionalità disponibili nel prodotto potranno essere messe a frutto nel tempo per aumentare l'efficienza del sistema
- un sensibile aumento dell'efficienza operativa dell'Azienda (workflow, integrazione con Office, multimedialità, internet, ecc.) utilizzando gli strumenti, integrati nel prodotto
- la possibilità di utilizzare strumenti, soluzioni e modelli già sperimentati con successo in altre Aziende simili pubbliche oltre che in quelle private
- la salvaguardia dell'investimento in quanto la soluzione segue costantemente e tempestivamente lo sviluppo tecnologico.

Oltre ai vantaggi dell'architettura la soluzione *internet based* di Oracle e-Business Suite offre un impatto estremamente minimizzato nei costi di gestione del parco installato. Basterà infatti agire nei programmi residenti nell'Oracle Application Server per avere immediatamente tutte le postazioni di lavoro aggiornate all'ultima release.

Ulteriori caratteristiche peculiari sono:

Facilità d'uso

La facilità d'uso è la caratteristica che permea tutta la suite ed è realizzata mediante una serie di componenti tra loro complementari. Va innanzitutto precisato che la versione fornita è interamente in lingua italiana, ivi compresi l'aiuto contestuale in linea, che può essere integrato con il manuale specifico dell'Azienda.

Interfaccia Utente Grafica

L'interfaccia Utente utilizzata è totalmente grafica, con possibilità di finestre multiple, e visualizzata come pagina web. Sono utilizzate tutte le componenti tipiche dell'ambiente grafico (widgets: menu pull-down, stacked regions, scroll, pop list, radio buttons, bottoni, ecc.; azioni: puntamento e click, singolo carattere, validazione campi, query by example, ecc.).

Possibilità di inserire nuovi dati negli archivi e nelle maschere

Mediante i *flexfield* descrittivi, campi che possono essere aggiunti in tutte le maschere di gestione, è possibile inserire nuovi dati nella propria applicazione, non previsti nello standard, per consentire l'acquisizione e il trattamento di informazioni importanti ed univoche per l'Utente.

Agente sensibile agli eventi

In tutti i moduli è attivo l'*alert*, un meccanismo sensibile agli eventi, che consente di far operare il sistema secondo le proprie necessità:

- inviare notifiche automatiche via e-mail gestire le eccezioni
- effettuare controlli di azioni periodiche guidati dagli eventi
- eseguire script SQL o programmi esterni
- integrare applicazioni esterne con la Oracle e-Business Suite

É facilmente personalizzabile da parte dell'Utente senza necessità di programmazione grazie alle seguenti caratteristiche:

Strumenti di workflow

In Oracle e-Business Suite è integrato il modulo di *workflow* che permette di modellare i processi aziendali:

- automatizza le regole aziendali
- instrada qualsiasi tipologia di documento (spreadsheet, immagini, ecc.)
- consente la notifica degli eventi
- può interfacciare qualsiasi applicativo che possa comunicare con Oracle RDBMS

Gestione della sicurezza

Sono previsti sofisticati meccanismi di gestione degli accessi attraverso la definizione di responsabilità cui sono associate delle regole di sicurezza che controllano a quali funzioni e valori l'Utente ha accesso.

Profili Utente

Un profilo Utente è un insieme di opzioni modificabili che consentono di personalizzare la modalità di esecuzione delle applicazioni. Oracle e-Business Suite determina un valore per ciascuna opzione del profilo Utente al momento del collegamento o della modifica della responsabilità. È possibile cambiare il valore di un'opzione del profilo Utente in qualsiasi momento.

Apertura e interoperabilità con software esterni

In tutti i moduli di Oracle e-Business Suite sono previste le *open interface* che consentono l'integrazione del modulo con altri sistemi, definendo le conversioni di formato delle informazioni, solitamente senza programmare.

È possibile:

- integrare applicazioni custom o proprietarie
- importare dati storici provenienti da sistemi informativi aziendali utilizzati in precedenza
- importare informazioni da spreadsheet.

I Benefici Attesi

Sempre basandosi sulle esperienze maturate da Oracle in ambito Aziendale e di Enti pubblici, i benefici che Oracle e-Business Suite può portare, sono in sintesi i seguenti:

Per le funzioni operative

- Razionalizzazione e snellimento dei flussi di lavoro
- Abbattimento dell'utilizzo e della circolazione di documenti cartacei, grazie al governo degli interi processi in modalità ON-LINE (esempio le approvazioni) ed alla possibilità di allegare alle transazioni documenti multimediali
- Alleggerimento del lavoro di inserimento dati, grazie alla modalità Self Service per alcuni documenti come richieste di approvvigionamento da Reparto, Note spese, Fogli ore, Approvazione di Documenti, ecc.
- Tempestività nelle fasi di chiusura contabile e di magazzino, grazie ad un'ampia gamma di automatismi ed a processi governati da un motore di Workflow
- Tempestività e facilità nella ricerca di informazioni, grazie al collegamento nativo tra tutti i documenti e le transazioni che compongono un flusso di lavoro
- Maggiore consapevolezza e controllo da parte degli utenti sulla fase di processo in corso di svolgimento, nonché sulle fasi precedenti e successive, grazie alla presenza del motore di Workflow che ne scandisce, anche in modalità grafica, la sequenza.

Per le funzioni dei Sistemi Informativi

- Garanzia di gestione di una soluzione applicativa mantenuta costantemente allo stato dell'arte della tecnologia
- Minimizzazione degli interventi di sviluppo software, grazie all'utilizzo di strumenti di configurazione delle logiche di business flessibili e parametrici (come Oracle Workflow) e a strumenti di sviluppo "standard de facto" a livello mondiale (come Oracle Developer)
- Contenimento delle spese di rete e di connessione di nuove utenze, ovunque sparse nel territorio, grazie alla fruibilità del 100% delle funzioni via Browser Intranet / Internet
- Abbattimento a zero dei costi di distribuzione/allineamento del software sui Client, sempre grazie all'architettura a tre livelli
- Apertura ed interoperabilità nativa (integrazione con strumenti Office, Interfacce Aperte)
- Ottimizzazione di costi di gestione del sistema che avviene, per concezione architeturale, in modo centralizzato.

Per le funzioni Direttive

- Migliore possibilità di controllo dei costi, dell'efficienza e dell'efficacia della propria struttura, grazie alla presenza di strumenti di business intelligence che puntano ad una base dati integrata, di qualità, univoca e sempre congruente
- Abbattimento dei costi amministrativi grazie alla razionalizzazione dei processi aziendali (es. funzione acquisti)
- Garanzia di massimo controllo dell'ambito operativo di tutti gli utenti
- Possibilità di monitoraggio e conseguenziale miglioramento e ottimizzazione continua dei processi aziendali, grazie alla tracciabilità dei fenomeni governati dal motore di Workflow
- Migliore utilizzo delle risorse su attività a maggior valore aggiunto, grazie al risparmio di tempo su aspetti di scarso valore come la gestione delle transazioni
- Ottimizzazione dell'intero processo degli acquisti, con benefici quali la più facile individuazione di un numero limitato di Fornitori Strategici e la razionalizzazione del "catalogo aziendale" che contenga esclusivamente gli articoli di cui l'azienda necessita e che sono già stati oggetto di negoziazione con i Fornitori.

I Moduli utilizzati

Il modulo che abbiamo utilizzato maggiormente è il "Procurement", che solitamente viene associato al modulo "Order Management" ma che al nostro caso di studi non è servito trattandosi di una cineteca. Infatti le movimentazioni di ingresso/uscita riguardano prestiti/restituzioni verso/da redazioni Rai o inserimenti di nuovi filmati girati su diversi supporti.

Gestione della Logistica

Oracle Logistics permette di gestire l'intero processo logistico: dalla gestione dei magazzini al trasporto dei beni, ai resi, eccetera. Ottimizza il flusso di materiali attraverso il ciclo di fornitura, gestendo e misurando continuamente il trade-off tra costo e servizio.

La soluzione Oracle per il magazzino agevola un efficiente movimento dei materiali in un ambiente controllato che permette di monitorare le giacenze per tutte le esigenze della domanda.

Poiché le scorte sono integrate con acquisti e ordini, la gestione delle transazioni e i processi decisionali non avvengono più sulla base di proiezioni probabilistiche, ma dei fatti. Questa integrazione riduce il capitale di esercizio, aiutando l'azienda ad accelerare la rotazione delle giacenze e a ridurre il livello delle scorte, senza pregiudicare il servizio alla clientela. Con aggiornamenti in tempo reale, avrete una visione accurata delle scorte e la capacità di gestire attivamente e rapidamente le aspettative dei clienti.

Warehouse Management

All'interno del modulo Inventory troviamo il WMS (*Warehouse Management*) che opera direttamente sugli stessi dati del resto del suite di E-Business suite di Oracle. Questo approccio "incorporato" elimina la duplicazione e lo stato latente di dati, così come le integrazioni complesse e costose connesse con un sistema tradizionale di WMS esterno "non integrato".

Questo applicativo ha, nel proprio nucleo, un motore flessibile e potente di regole di affari che consente con facilità l'adattamento dei processi chiave di WMS, come la raccolta diretta ed il salvataggio dei dati senza la necessità di adattare il codice per il cliente. Questa architettura è basata su regole estendibili e permette ai processi chiave di evolversi dinamicamente ed adattarsi. Per mezzo del "Rules Workbench" le regole specifiche possono essere selezionate in base al contenuto di ogni oggetto di business (per esempio ordini, clienti, articoli, fornitori, progetti ecc.). In più queste regole possono variare a seconda del magazzino e dalla data di censimento dei prodotti, consentendo al sistema di adattarsi dinamicamente ai bisogni e ai cambiamenti.

Le tecnologie RFID sono abilitate all'interno dell'applicazione e permettono l'identificazione automatica delle entità ogni volta che una modifica elettronica effettuata nel magazzino è portata all'interno del raggio di azione di un lettore. Poiché le etichette RFID possono essere lette senza aver bisogno di una linea visiva e poiché possono essere letti più transponder simultaneamente, questa tecnologia può automatizzare efficientemente le operazioni in base all'evento ed il flusso di dati nel magazzino. Tutto ciò riduce i tempi, i costi operativi e facilita l'interazione con il

sistema; le transazioni sono innescate senza l'intervento dell'operatore. Oracle WMS viene sviluppato con la possibilità di utilizzare l'RFID sui prodotti e le funzionalità associate a tale tecnologia, compreso il supporto dei lettori, i filtri sui dati, l'identificazione delle modifiche, i meccanismi di risposte con suoni e luci, ecc. Questo metodo incorporato riduce significativamente la complessità ed il costo degli investimenti che è molto importante data la natura dinamica e d'evoluzione della tecnologia RFID.

Oracle WMS integra completamente l'uso dei License Plate Numbers (LPN) nei processi del magazzino. Un LPN è un numero unico (in genere bar-coded) che può essere assegnato a qualsiasi gruppo arbitrario dell'inventario. LPN permette agli utenti di realizzare le transazioni complesse di inventario con una singola esplorazione attraverso il codice a barre del LPN "che gestisce l'unità", migliorando significativamente l'efficienza e l'esattezza.

Oracle WMS fornisce l'integrazione ai sistemi di ricerca dell'informazione e di memorizzazione, ai carrelli, ai trasportatori ed ai veicoli guidati automatizzati usando i corredi pre-costruiti di integrazione così come le API open-source.

Le API permettono ad un sistema "esterno" di controllo di magazzino o ad un'altra interfaccia il passaggio delle informazioni di rintracciamento al WMS per poi essere processate come transazione in Oracle.

Il menù di controllo del WMS permette al responsabile del magazzino di controllare ed infine sintonizzare le attività. Può essere usato per l'amministrazione di tutte le operazioni più caratteristiche di un magazzino, come il controllo delle uscite/entrate, l'aggiornamento di una richiesta, il cambiamento delle priorità, l'assegnazione manuale di operazione ed al rilascio di operazione.

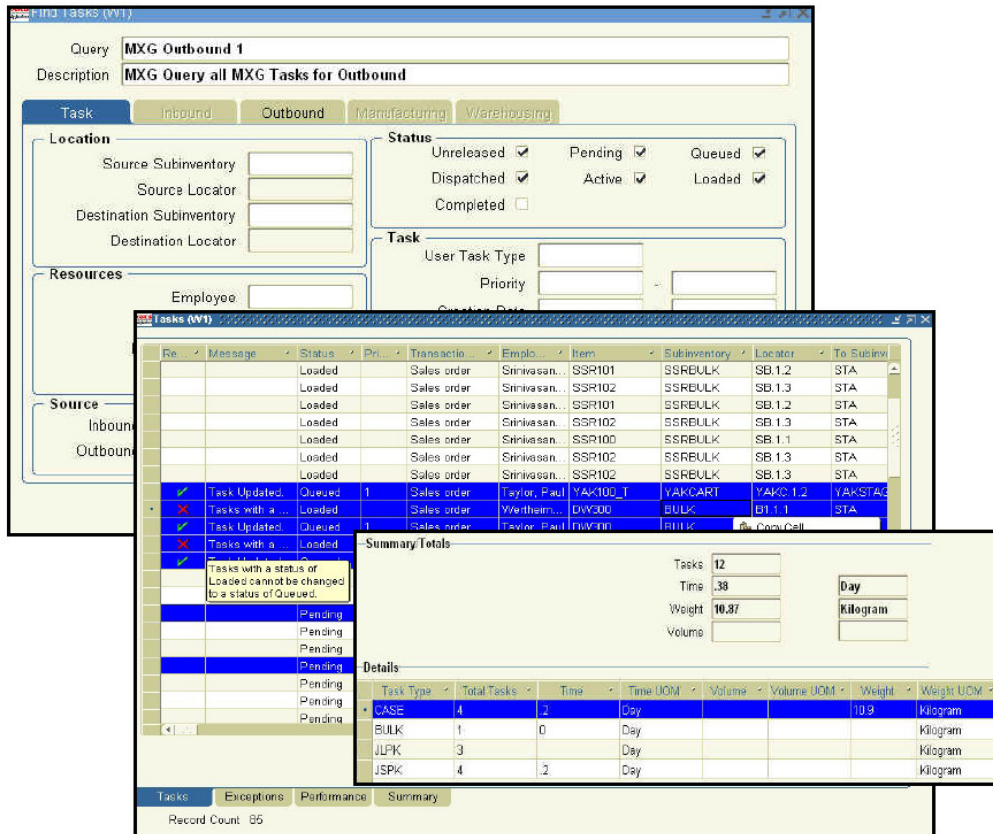


Figura A-9 Interfaccia Oracle modulo Inventory

I vantaggi introdotti dall'Oracle WMS migliorano l'efficienza dei funzionamenti di business configurando gli allarmi in tempo reale e i workflow della catena di rifornimento su cui sono state inserite notifiche di eventi ed eccezioni. Per esempio l'utente può configurare un allarme che informa il personale della scarsità di un determinato materiale. I workflow inseriti dall'utente definiscono azioni correttive che possono essere innescate da tutte le operazioni che non sono state completate come richiesto.

Gestione degli Ordini

Oracle Order Management offre la flessibilità necessaria a gestire tutti i processi di acquisizione e evasione ordini, prevede un sistema di controllo per garantire che le consegne rispondano esattamente alle richieste dei clienti e una piattaforma collaborativi per tutti i partner commerciali. Oracle Order Management gestisce l'intero ciclo di vita dell'ordine dall'inserimento al pagamento, rendendo il processo più efficiente e riducendo significativamente i costi di esecuzione.

Framework Web Application

Ibatis

Il framework Ibatis Data Mapper aiuta a ridurre significativamente la quantità di codice Java di cui si ha bisogno normalmente per accedere ad una base di dati relazionale. Ibatis mappa semplicemente i JavaBeans alle dichiarazioni SQL usando un descrittore XML molto semplice. La semplicità è il vantaggio chiave di Ibatis rispetto ad altri framework ed tool di associazione di oggetti a strutture dati relazionali. Per usare questo strumento basta solamente avere familiarità con i JavaBeans, XML e SQL. C'è veramente poco altro da imparare.

Non è richiesto definire uno schema complesso per i join tra le tabelle o eseguire query complesse. Usando il Data Mapper si ha il pieno potere di SQL a portata di mano.

Le API di Ibatis Data Mapper permette ai programmatori di mappare facilmente gli oggetti JavaBeans ai parametri di PreparedStatement e ai ResultSets. La filosofia che c'è dietro è semplice: fornire un framework semplice per realizzare l'80% delle funzionalità JDBC usando soltanto 20% del codice.

Il Data Mapper fornisce un framework molto semplice per usare i descrittori XML per tracciare JavaBeans, implementazioni di oggetti Map, tipi wrapper primitivi (es. String, Integer...) e perfino documenti di XML ad una dichiarazione SQL. Quello che segue è una descrizione ad alto livello del ciclo di vita:

1. Fornire un oggetto come un parametro (come JavaBean, Map o wrapper primitivi). Il parametro oggetto sarà usato per impostare i valori di input di uno statement di tipo update, insert o delete, o valori delle clausole where in una query,...
2. Eseguire lo statement mappato. Questo step è dove accade la magia. Il framework Data Mapper genera una istanza di PreparedStatement, impostando tutti i parametri usando l'oggetto fornito in input, esegue la dichiarazione e costruisce un oggetto risultante a partire dal ResultSet.
3. Nel caso di un update, viene restituito il numero di righe modificate. In caso di query, viene restituito un singolo oggetto, o una collezione di oggetti. Come i parametri, gli oggetti restituiti possono essere JavaBean, Map, tipo wrapper primitivi o istanze XML.

Lo schema seguente illustra il flusso appena descritto.

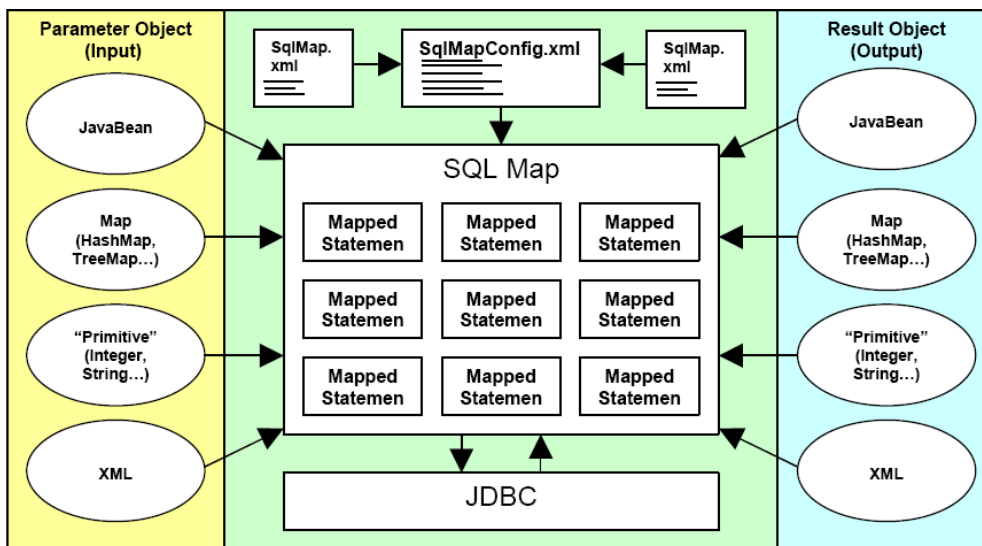


Figura A-10 Architettura del framework Ibatis

Il framework SQL Maps è molto tollerante rispetto a cattivi modelli di database e addirittura a cattivi modelli ad oggetti. Ciò nonostante, si raccomanda di progettare il proprio database (attraverso una corretta normalizzazione) ed il proprio modello secondo lo stato dell'arte. Facendo ciò, si avranno buone performance ed un progetto migliore.

Il modo più semplice per iniziare è analizzare su cosa si sta lavorando. Quali sono gli oggetti di business? Quali sono le tabelle del database? Come si relazionano tra loro? Come primo esempio, si consideri la seguente semplice classe Person, conforme alla tipica convenzione sui JavaBean.

SQL Maps non vi costringe ad avere relazioni del tipo *una-tabella-per-classe* oppure *più-tabelle-per-classe* oppure *più-classi-pertabella*. Dato che avete disponibile tutta la potenza dell'SQL, esistono ben poche restrizioni.

Dopo aver preso confidenza con le classi e le tabelle da utilizzare, il luogo migliore per iniziare è il file di configurazione di SQL Maps. Tale file agisce da *radice* per quanto riguarda la configurazione della nostra particolare implementazione di SQL Maps. Il file di configurazione è un file XML. Al suo interno configureremo proprietà, fonti dati JDBC e *mappature* SQL. Il file rappresenta un punto unico di configurazione per i vostri DataSource, che possono essere diversi e differenti. Il framework può gestire

varie implementazioni dell'interfaccia DataSource, tra cui l'iBATIC SimpleDataSource, Jakarta DBCP (Commons) e qualunque altro DataSource che possa essere reperito tramite un contesto JNDI (ad esempio dall'interno di un application server).

Ora che abbiamo un DataSource configurato ed il nostro file di configurazione è pronto, dobbiamo produrre un file SQL Map che contenga il nostro codice SQL e le mappature per gli oggetti che faranno da parametri di input e di output.

Struts

Come già detto Struts è un MVC web application framework, ovvero è un insieme di classi ed interfacce che costituiscono l'infrastruttura per costruire web application J2EE conformi al design pattern MVC.

I componenti fondamentali di struts sono:

- *ActionServlet*: e' la servlet di controllo centralizzata che gestisce tutte le richieste dell'applicazione.
- *struts-config.xml* E' il file XML di configurazione di tutta l'applicazione. In questo file vengono definiti gli elementi dell'applicazione e le loro associazioni.
- *Action*: le Action sono le classi alle quali la ActionServlet delega l'elaborazione della richiesta.
- *ActionMapping*: contiene gli oggetti associati ad una Action nello struts-config.xml come ad esempio gli ActionForward.
- *ActionForm*: gli ActionForm sono classi contenitori di dati. Vengono popolati automaticamente dal framework con i dati contenuti nelle request http.
- *ActionForward*: contengono i path ai quali la servlet di Struts inoltra il flusso elaborativo in base alla logica dell'applicazione.
- *Custom-tags*: Struts fornisce una serie di librerie di tag per assolvere a molti dei più comuni compiti delle pagine JSP.

La ActionServlet è la servlet di controllo di Struts. Gestisce tutte le richieste client e smista il flusso applicativo in base alla logica configurata. Si potrebbe definire come la 'spina dorsale' di una applicazione costruita su Struts. Tutta la configurazione

dell'applicazione è contenuta nello struts-config.xml. Questo file XML viene letto in fase di start-up dell'applicazione dalla ActionServlet e definisce le associazioni tra i vari elementi di Struts. Nello struts-config.xml sono ad esempio definite le associazioni tra i path delle richieste http e le classi Action associate alle richieste stesse.

Le associazioni tra le Action e gli ActionForm, che vengono automaticamente popolati dal framework con i dati della richiesta ad essi associata e passati in input alla Action.

Contiene inoltre l'associazione tra la Action e le ActionForward , ovvero i path configurati nello struts-config.xml ai quali la ActionServlet redirigerà il flusso applicativo al termine della elaborazione della Action.

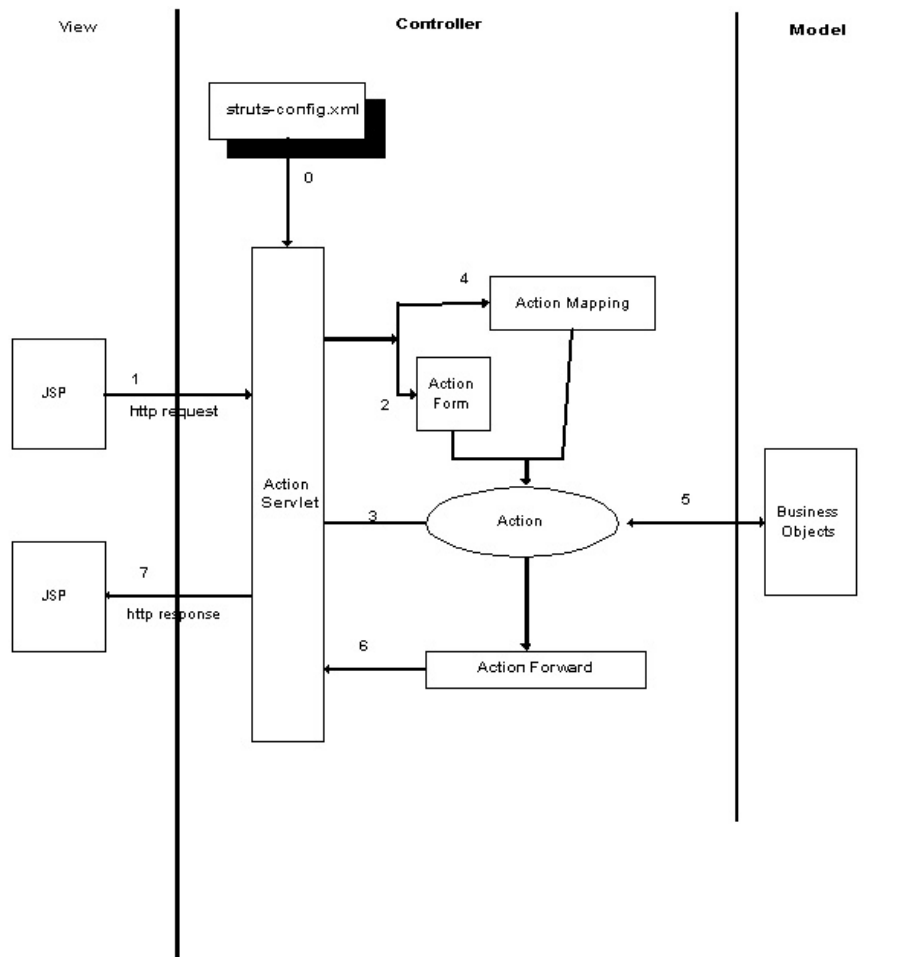


Figura A-11 Implementazione di Struts del pattern MVC

Nella figura precedente è rappresentato schematicamente il flusso elaborativo nella logica di Struts:

1. Il client invia una richiesta http (1)
2. La richiesta viene ricevuta dalla servlet di Struts che provvede a popolare l'ActionForm associato alla richiesta con i dati della request (2) e l'ActionMapping con gli oggetti relativi alla Action associata alla richiesta (4) . Tutti i dati di configurazione sono stati letti in fase di start-up dell'applicazione (0) dal file XML struts-config.xml.
3. La ActionServlet delega l'elaborazione della richiesta alla Action associata al path della richiesta (3) passandole in input request e response http e l'ActionForm e l'ActionMapping precedentemente valorizzati.
4. La Action si interfaccia con lo strato di business che implementa la logica applicativa. Al termine dell'elaborazione restituisce alla ActionServlet un ActionForward (6) contenente l'informazione del path della vista da fornire all'utente.
5. La ActionServlet esegue il forward alla vista specificata nell'ActionForward (7).

Caratteristiche di base di Jakarta Struts

Dalla discussione dei precedenti paragrafi possiamo quindi individuare alcune caratteristiche peculiari di Struts, che sono poi comuni anche ad altri MVC framework.

- Esiste una sola servlet di controllo centralizzata. Tutte le richieste sono mappate sulla ActionServlet nel web.xml dell'applicazione. Ciò consente di avere un unico punto di gestione del flusso applicativo e quindi permette di implementare in modo univoco e centralizzato funzioni quali sicurezza, logging, filtri etc.
- Le viste dell'applicazione non contengono al loro interno il riferimento al flusso dell'applicazione e non contengono logica applicativa. I livelli logici dell'applicazione sono disaccoppiati.
- Le viste sono identificate con nomi logici definiti nel file di configurazione struts-config.xml. Nel codice Java non è presente alcun riferimento a nomi di pagine JSP il che rende molto più semplice variare il flusso applicativo.

- Tutta la configurazione dell'applicazione è scritta esternamente in un file XML il che consente di modificare le associazioni tra le richieste http e le classi ad essa associate in modo molto semplice.

C'è da osservare che tutti i componenti di Struts descritti nel paragrafo precedente fanno parte del livello di controllo tranne i custom-tags che fanno parte della view. Le stesse Action che sono le classi alle quali la ActionServlet delega l'elaborazione delle richieste sono componenti del controller e non del model.

Ciò per sottolineare che Struts è un framework model-neutral, ovvero che implementa esclusivamente i livelli di controller e view.

Utilizzando Struts è possibile realizzare il livello di business logic in base alle proprie scelte; con semplici classi Java quindi implementando la logica applicativa nel web-container, o ricorrendo agli EJB quindi sfruttando i servizi del EJB-container.

Principali vantaggi nell'uso di Jakarta Struts

Questa discussione introduttiva su Struts non può ovviamente evidenziarne tutti gli aspetti né essere considerata come una trattazione esaustiva del framework, per la quale sarebbe necessario molto più spazio.

Da quanto esposto però si possono già evidenziare alcune delle caratteristiche di una applicazione sviluppata con Jakarta Struts e alcuni vantaggi conseguenti al suo utilizzo:

- Modularità e Ricusabilità
- I diversi ruoli dell'applicazione sono affidati a diversi componenti. Ciò consente di sviluppare codice modulare e più facilmente riutilizzabile.
- Manutenibilità

L'applicazione è costituita da livelli logici ben distinti. Una modifica in uno dei livelli non comporta modifiche negli altri. Ad esempio una modifica ad una pagina JSP non ha impatto sulla logica di controllo o sulla logica di business, cosa che avveniva nel JSP Model 1.

- Rapidità di sviluppo
- A differenza di quanto avveniva utilizzando il JSP Model 1, è possibile sviluppare in parallelo le varie parti dell'applicazione, view (JSP/HTML) e logica di business (Java) sfruttando al meglio le conoscenze dei componenti del team di sviluppo. Si possono utilizzare sviluppatori meno esperti e anche con

poche conoscenze di Java per la realizzazione delle view, permettendo agli sviluppatori Java più esperti di concentrarsi sulla realizzazione della business logic.

Axis

Axis non è un semplice engine SOAP ma un framework per realizzare sistemi di integrazione basati su SOAP. Il progetto non è una revisione della precedente versione del toolkit, Apache SOAP 2.x, ma una completa riscrittura che nasce dalla necessità di disporre di una migliore architettura basata su criteri di:

- flessibilità: è possibile estendere l'engine per eseguire elaborazioni custom degli header dei messaggi;
- componibilità: i gestori di richieste, chiamati Handler in terminologia Axis, possono essere riutilizzati;
- indipendenza dal meccanismo di trasporto dei messaggi: in questo modo è possibile utilizzare vari protocolli oltre ad HTTP come SMTP, FTP e messaging.

Rispetto alla versione precedente di SOAP in Axis si è cercato inoltre di migliorare le performance utilizzando SAX per l'elaborazione dei messaggi.

Nel seguito si farà riferimento alla beta 2 di Axis che implementa le specifiche 1.1 di SOAP e offre un supporto parziale a SOAP 1.2.

Installazione

L'installazione di Axis è abbastanza semplice: il prerequisito fondamentale è un servlet container e negli esempi presentati si farà riferimento a Tomcat.

Innanzitutto si deve copiare il folder webapps/axis dalla distribuzione nel folder webapps di Tomcat. I web service saranno quindi accessibili all'URL

```
http://hostname:port/axis/...
```

E' possibile cambiare il nome di questo folder ma cambierà corrispondentemente anche l'URL.

All'interno di webapps/axis si trova il folder WEB-INF in cui copiare le classi che si vogliono esporre come web service: i .class verranno copiati nel folder classes mentre i .jar nel folder lib. Prima di usare Axis è comunque necessario copiare un parser XML, Xerces o Crimson, in WEB-INF/lib.

Ora Tomcat è configurato per esporre web service con Axis

Un semplice web service

In questa sezione si affronterà il problema del deploy di un web service con Axis mostrando tre casi di difficoltà crescente:

1. deploy di una classe Java con copia di file;
2. deploy della stessa classe mediante deployment descriptor;
3. deploy di un EJB

Per i primi due esempi si userà la seguente classe Java che implementa un semplice sommatore:

```
public class Calculator {
    public int add(int x, int y) {
        return x + y;
    }
}
```

Per eseguire un deploy istantaneo della classe è sufficiente copiare il sorgente nel folder della web application di Axis con estensione jws, Java Web Service. A questo punto il servizio è immediatamente disponibile all'URL `http://localhost:8080/axis/Calculator.jws` E' il runtime di Axis che compila il file JWS in classe Java e inoltra quindi ad essa le chiamate SOAP.

Questa modalità è semplice e immediata ma non sempre utilizzabile; spesso il servizio di cui fare deploy non sarà implementato con una singola classe Java ed è inoltre necessario un modo per configurare i parametri di deploy come ad esempio il mapping dei tipi. Per questa ragione in Axis sono previsti i deployment descriptor, documenti XML che contengono le informazioni di deploy; segue come esempio il deployment descriptor per il servizio calculator:

```

<deployment xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

<service name="calculator" provider="java:RPC">
<parameter name="className"
value="com.mokabyte.javaxml.calculator.Calculator"/>
<parameter name="allowedMethods" value="add"/>
</service>

</deployment>

```

L'elemento service definisce appunto il web service a cui è stato dato il nome calculator. L'attributo provider permette di specificare il modo in cui il servizio viene esposto; in questo caso si usa il provider per una classe Java quindi si usa java:RPC come valore dell'attributo.

L'elemento parameter permette di specificare i parametri di configurazione del servizio, in questo caso il nome della classe e i metodi esposti. Il deployment descriptor è contenuto in un file che ha in genere estensione wsdd.

EJB e web service

Un web service espone in genere logiche applicative implementate come EJB; in questa sezione si vedrà appunto come accedere via SOAP ad un EJB. Si supponga di voler realizzare un servizio che esponga i dati di una lista di attività. Il servizio verrà implementato come un session EJB stateless che espone un singolo metodo getList(): questo ritorna la lista delle attività come documento XML. Ecco la remote interface:

```

package com.mokabyte.javaxml.todolist;

import javax.ejb.*;
import java.rmi.RemoteException;

public interface TodoList extends EJBObject {
    public String getList() throws RemoteException;
}

```

Questa è invece l'implementazione del metodo `getList()` contenuta nella classe `ToDoListEJB.java`

```
...
public String getList() throws RemoteException {
    return "<ToDoList>" +
        "<Task>Leggere la posta</Task>" +
        "<Task>Riunione</Task>" +
        "<Task>Corsa nel parco</Task></ToDoList>";
}
...
```

Una implementazione reale avrebbe considerato identificativo e ruolo dell'utente per ricercare le attività in un database oppure interrogando un engine di workflow ma ai fini dell'esempio non sarebbe cambiato nulla. Ecco il deployment descriptor per l'ejb

```
<deployment xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">

<service name="todolist" provider="java:EJB">
<parameter name="className"
value="com.mokabyte.javaxml.todolist.ToDoList"/>
<parameter name="beanJndiName" value="moka/ToDoListHome"/>
<parameter name="homeInterfaceName"
value="com.mokabyte.javaxml.todolist.ToDoListHome"/>
<parameter name="remoteInterfaceName"
value="com.mokabyte.javaxml.todolist.ToDoList"/>
<parameter name="jndiContextClass"
value="org.jnp.interfaces.NamingContextFactory"/>
<parameter name="jndiURL" value="localhost:1099"/>
<parameter name="allowedMethods" value="getList"/>
</service>

</deployment>
```

Come si può vedere è stato indicato un provider di tipo `java:EJB` e vi sono più parametri per specificare il nome JNDI, le interfacce home e remote e infine le property JNDI per connettersi al server J2EE, in questo caso JBoss. A questo punto è possibile fare deploy del servizio e accedervi da un qualsiasi client SOAP.

In precedenza è stato mostrato come accedere a un servizio usando JAX-RPC; nella sezione seguente si mostrerà come realizzare un client più evoluto basato su WSDL.

Axis e WSDL

WSDL (Web Services Description Language) è un linguaggio XML che descrive un web service, ovvero dove si trova, quali metodi espone e quali tipi di dato utilizza. Axis offre un supporto completo a WSDL: innanzitutto è possibile ottenere il WSDL di un servizio in modo immediato con un browser e appendendo un "?WSDL" all'URL del servizio.

Ad esempio accedendo all'URL

```
http://localhost:8080/axis/services/todolist?WSDL
```

si ottiene il seguente documento

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
targetNamespace="http://localhost:8080/axis/services/todolist"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:impl="http://localhost:8080/axis/services/todolist-impl"
xmlns:intf="http://localhost:8080/axis/services/todolist"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:message name="getListResponse">
<wsdl:part name="return" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="getListRequest">
</wsdl:message>
<wsdl:portType name="ToDoList">
<wsdl:operation name="getList">
<wsdl:input message="intf:getListRequest"/>
<wsdl:output message="intf:getListResponse"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="todolistSoapBinding" type="intf:ToDoList">
```

```

<wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="getList">
<wsdlsoap:operation soapAction=""/>
<wsdl:input>
<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="getList" use="encoded"/>
</wsdl:input>
<wsdl:output>
<wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/axis/services/todolist"
use="encoded"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ToDoListService">
<wsdl:port binding="intf:todolistSoapBinding" name="todolist">
<wsdlsoap:address
location="http://localhost:8080/axis/services/todolist"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

Questa funzionalità è molto utile non solo perchè è semplice e immediata ma soprattutto perchè è indipendente da Axis e da Java: un qualsiasi client SOAP in grado di utilizzare WSDL si può collegare all'URL e usare il documento così generato per interrogare il servizio.

Conclusioni

Axis è un toolkit con molte caratteristiche interessanti, in particolare il supporto a WSDL che semplifica notevolmente l'integrazione con altri sistemi.

Axis risulta quindi essere un importante strumento per lo sviluppatore Java.

B. Contenuto del CD