Impianti Informatici

POLITECNICO DI MILANO





Web application - tecnologie

Web Application: tecnologie

Java-based (**J2EE**)

- Sviluppata inizialmente da Sun
- Cross-platform e open source
  - Gestire direttamente le funzionalità dell'applicazione
  - Benefici della comunità di utenti

Microsoft-based (**.Net**)

- Componenti proprietarie
- Tecnologia limitata alle piattaforme Microsoft
- Teoricamente accetta molteplici linguaggi di programmazione

Linguaggio: J2EE

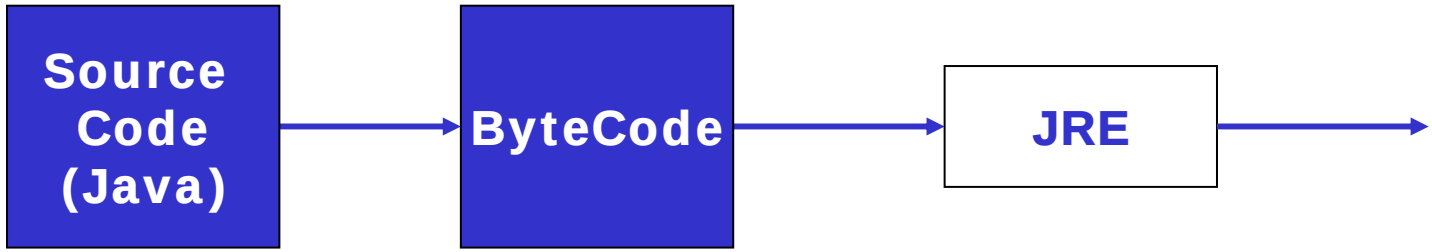
Codice scritto in Java

Codice Java compilato nel ***bytecode***

- Un intermediario cross-platform
- Mix tra codice sorgente e linguaggio macchina

A run-time, JRE (Java Runtime Environment) interpreta il bytecode

- Esegue l'applicazione



Linguaggio: .NET

Codice scritto in uno (o più) dei linguaggi supportati

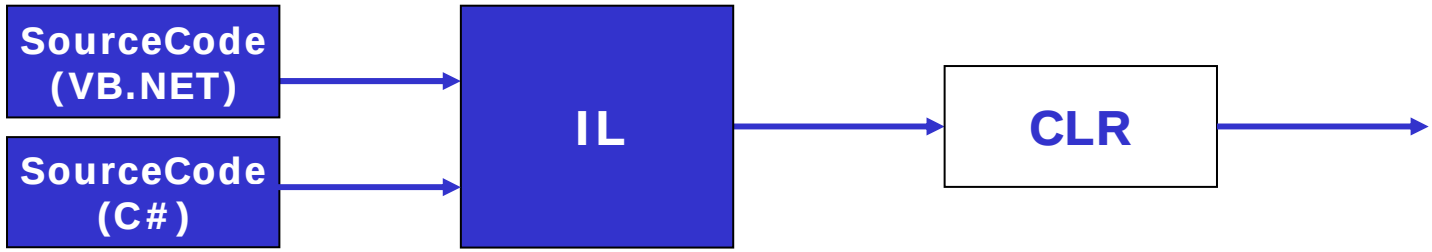
- VB.NET, C#,…

Il codice viene traslato nel (Microsoft) ***Intermediate Language*** (IL)

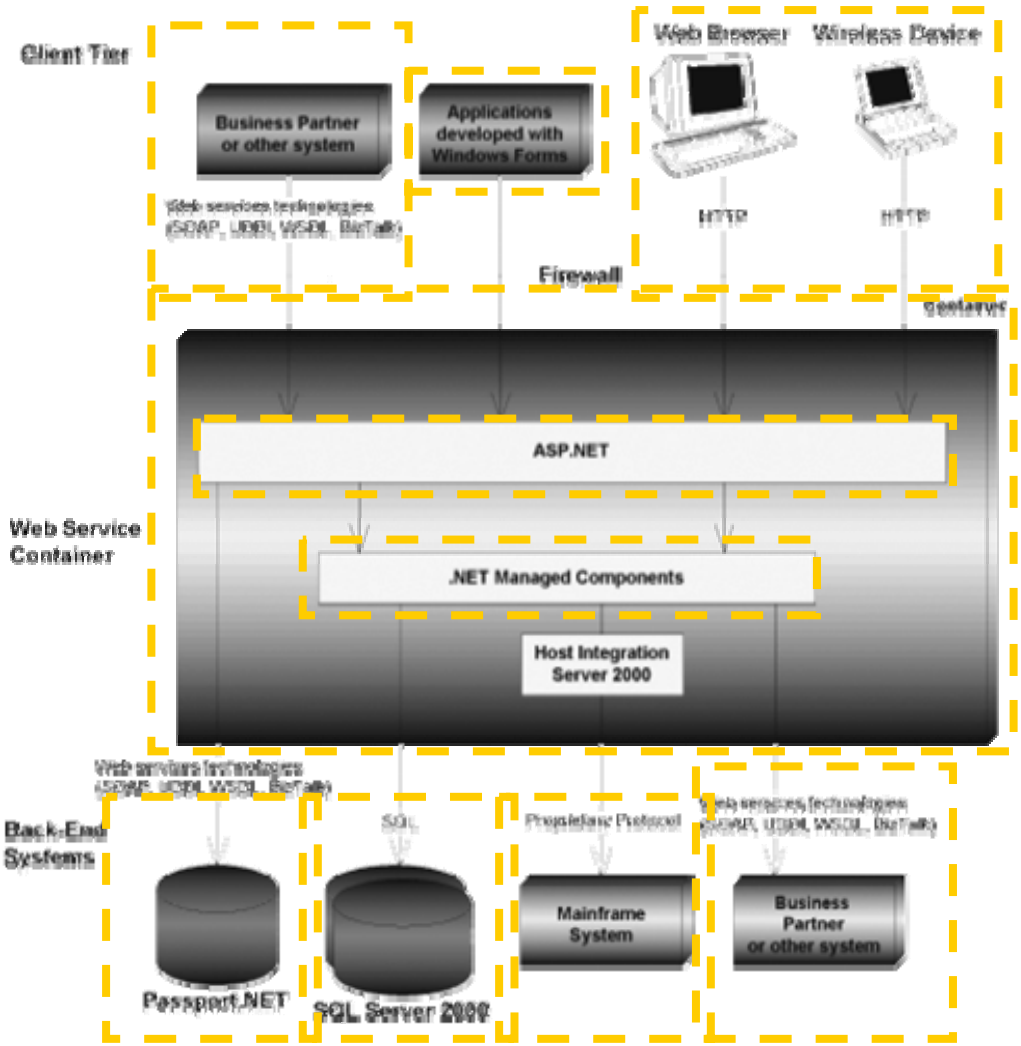
- Linguaggio cross-platform (simile al bytecode)

A run-time, CLR (Common Language Runtime) interpreta il codice IL

- Esegue l'applicazione

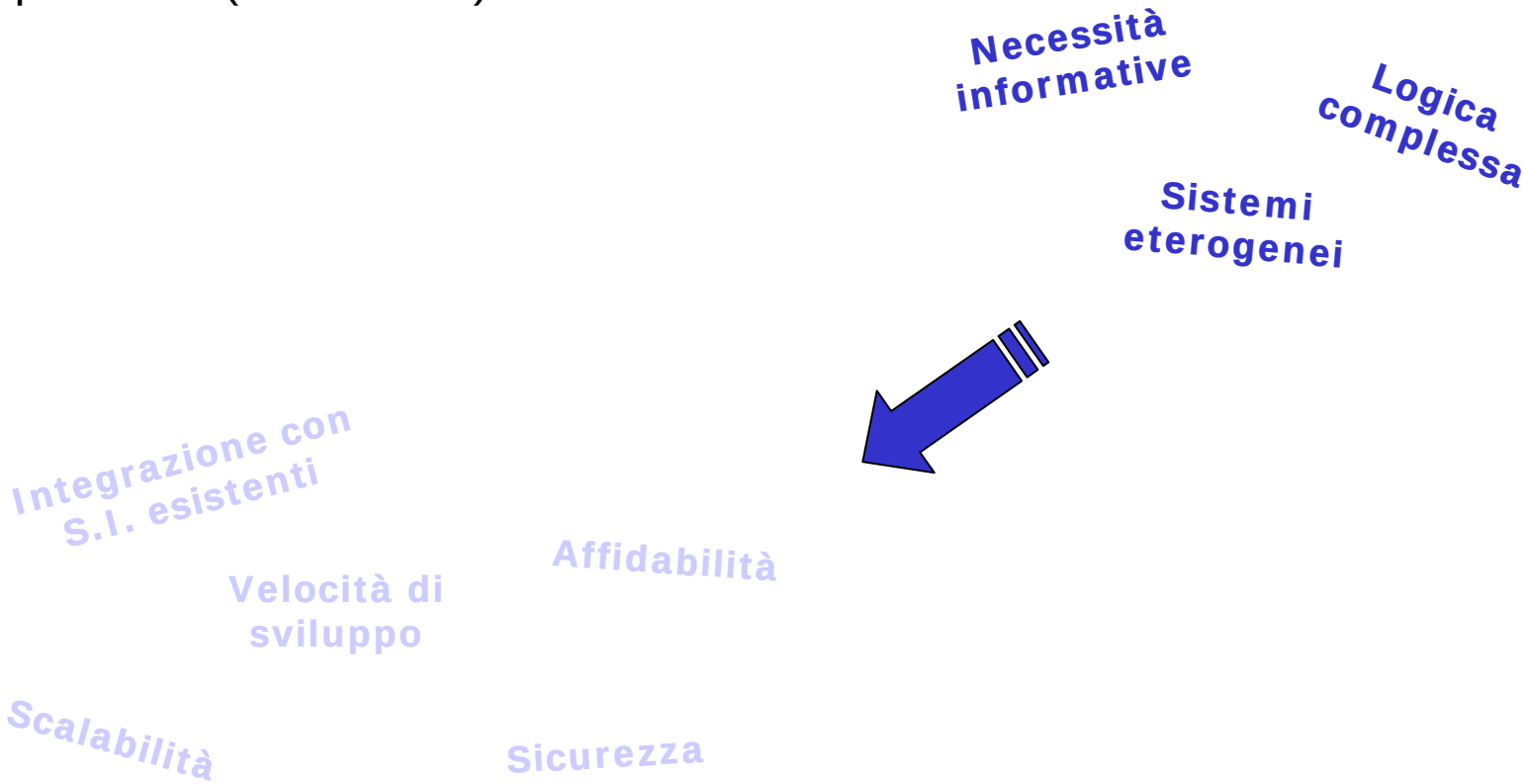


Tecnologia .Net



J2EE (Java 2 Platform Enterprise Edition)

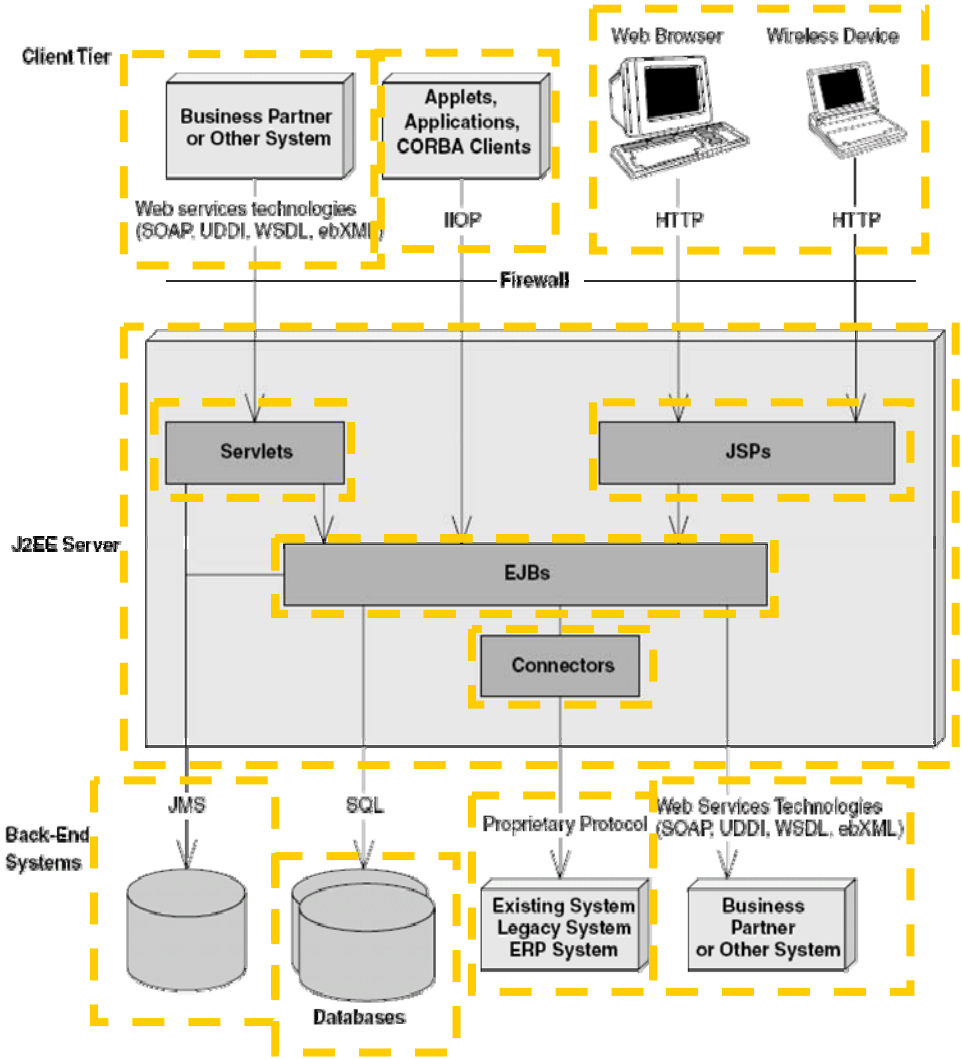
Suite di *servizi middleware* a disposizione degli sviluppatori di applicazioni (server-side)



J2EE

Servizi disponibili

- Servlet/JSP
- EJB
- JDBC
- JMS
- Web Services
- RMI, RMI-IIOP
- JNDI
- JTA
- JAXP



Middleware ad oggetti

Comunicazione tra oggetti distribuiti

- Remote Procedure Call (RPC)
- Oggetti remoti

Difficoltà nell'invocazione remota:

- Passaggio di parametri
  - Marshalling/Unmarshalling
  - Streaming
  - Modalità di passaggio
    - By value
    - By reference
- Instabilità del sistema
  - Macchine
  - Rete

RMI (RMI-IIOP) e CORBA

Remote Method Invocation

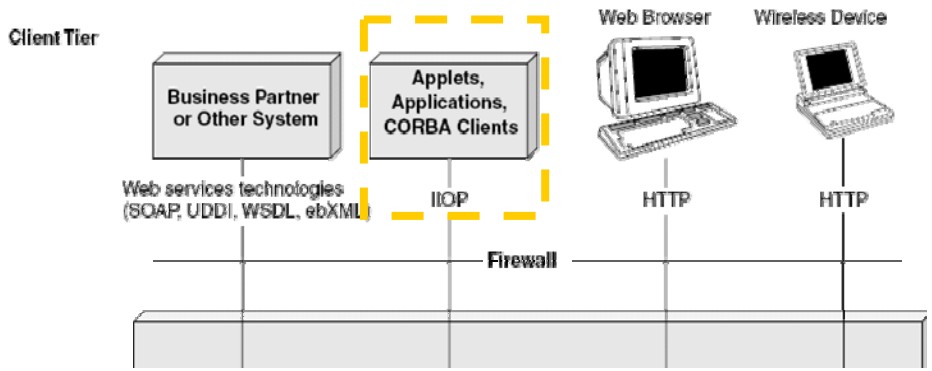
- Middleware nativo per il mondo JAVA

RMI-IIOP (Internet-Inter-ORB-Protocol)

- Compatibilità con CORBA

CORBA (Common Object Request Broker Architecture)

- IDL (Interface Definition Language)



Message Oriented Middleware (MOM)

Coordinano elementi della logica applicativa

Messaggio: richiesta asincrona generata da un applicativo

- Informazioni formattate
- Coordinano i sistemi

Messaging Agent

- Creazione, invio, ricezione messaggi

Indirizzamento messaggi

Servizi sui singoli messaggi

- QoS

MOM: Point-To-Point vs Publish-And-Subscribe

Point-To-Point (PTP)

- Funzionamento a code
  - Un client invia un messaggio verso una coda specifica
- Simile a mailbox
  - Unica coda per tutti i messaggi

Publish-And-Subscribe (Pub/Sub)

- *Topic*: è un message broker
  - I client pubblicano i messaggi
  - I client si sottoscrivono ai messaggi
- Simile a newsgroup

Java Messages Service (JMS)

Message Oriented Middleware
- Interfacce e semantica associata

Sia PTP che Pub/Sub

Applicazione JMS:
- JMS client
- Non-JMS client
- Messaggi
- JMS provider
- Oggetti amministrati

Limitazioni:
- Security
- Load-balancing/fault tolerance
- Triggering client
- Message type repository

Servlet

Applicazioni lato server basate su Java

Richiedono un particolare componente un servlet-container (o servlet-engine)

- Appartiene ad un *application server*
- Funzionalità a disposizione dell'applicazione

Ciclo di vita

- Temporanee: vengono instanziate nel momento della richiesta e distrutte al temine della richiesta
- Permanenti: instanziate all'avvio del server e distrutte solo quando viene spento

Servlet vs CGI

Servlet

- JVM sempre attiva
- Un thread per ogni richiesta
- Singola copia del codice
- Servlet in esecuzione

CGI

- Un processo per ogni richiesta
- Multiple istanze
- Difficile operazioni su persistenza dati

Migliori prestazioni

Platform-indipendent

Librerie Java a disposizione

- Applet
- Database
- Rmi
- …

Servlet: funzioni tipiche

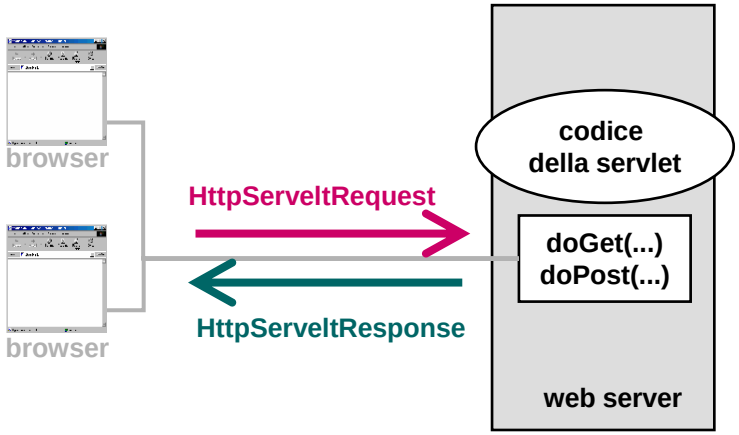
Gestire oggetto di sessione

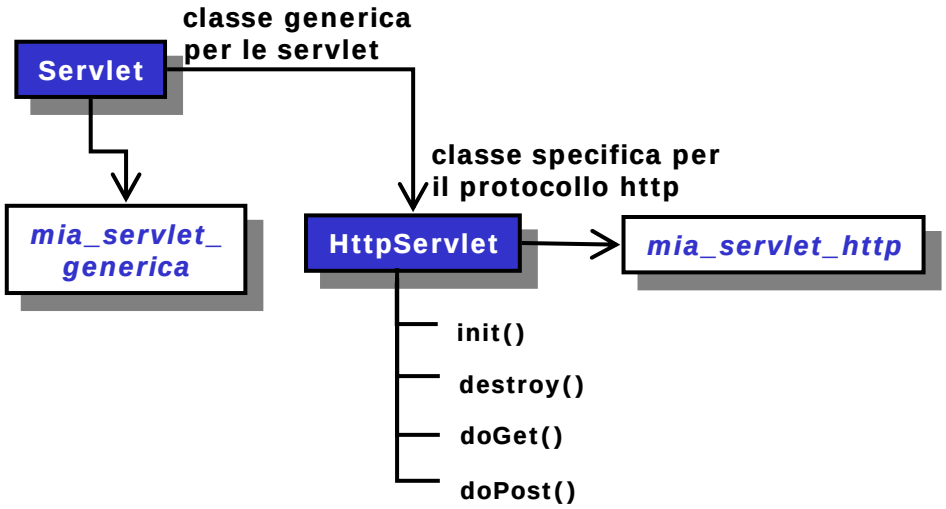
Verifica autorizzazioni

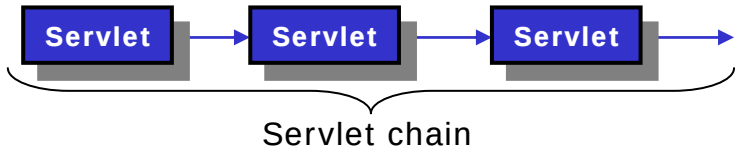
Prelevare i dati inviati dal browser

Interazione con Database

Inviare pagine al browser







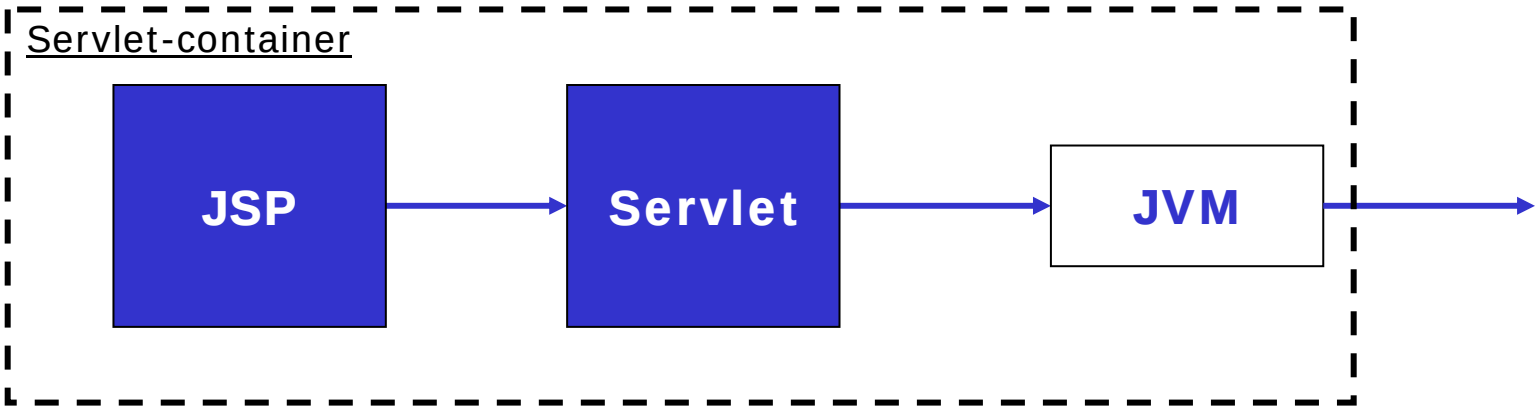
JSP (Java Server Pages)

Interfaccia utente

Necessitano di un servlet-container

- Compilate in servlet
- Eseguite come codice Java

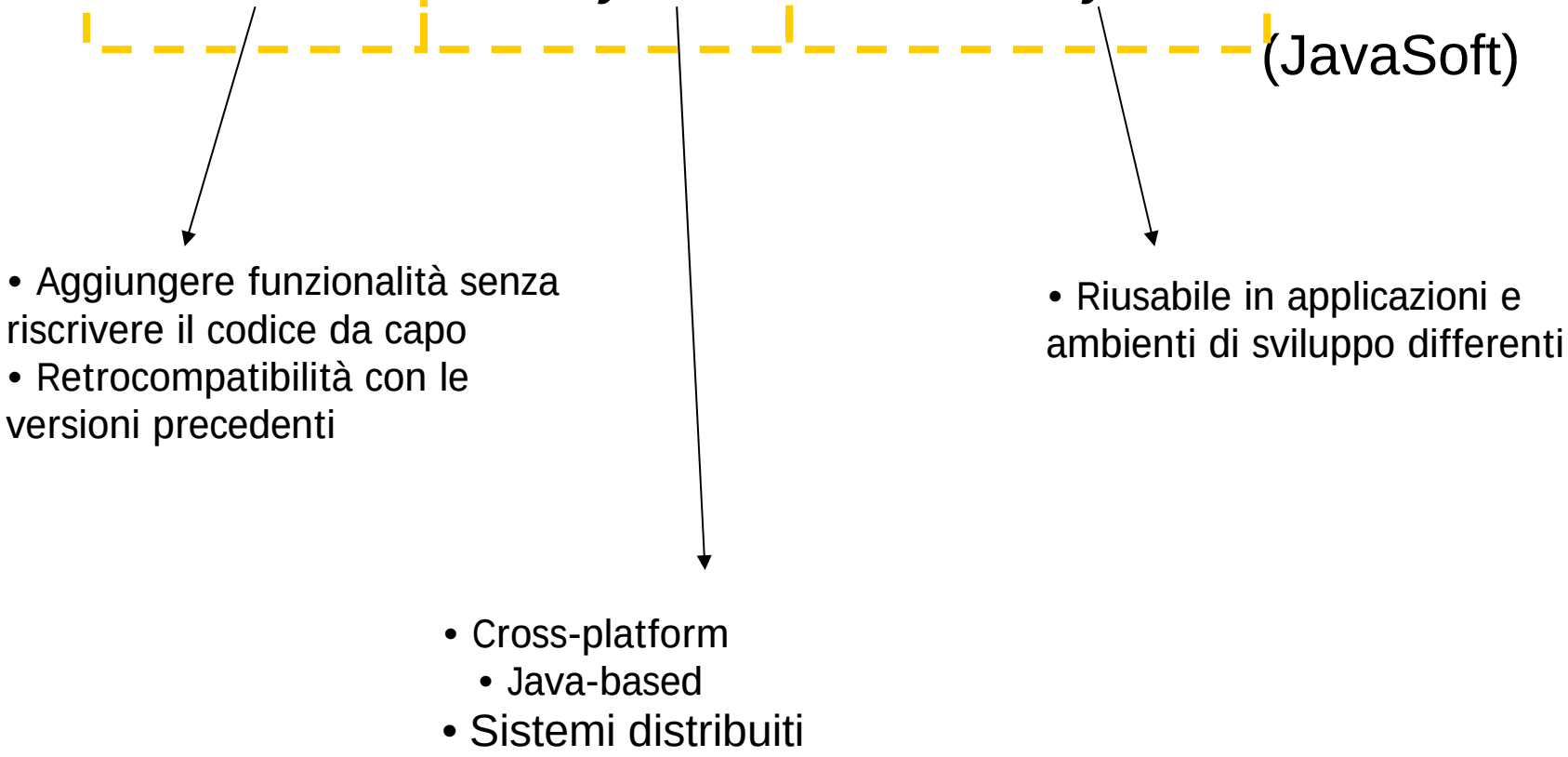
Stile di programmazione incentrato sul look-and-feel



JavaBean

“Write once, run anywhere, reuse everywhere“

(JavaSoft)



- Aggiungere funzionalità senza riscrivere il codice da capo
- Retrocompatibilità con le versioni precedenti

- Cross-platform
  - Java-based
- Sistemi distribuiti

- Riusabile in applicazioni e ambienti di sviluppo differenti

JavaBean

Property

- Comportamento, caratteristiche bean
- Customization

Event

- Comunicazione con altri bean

Persistence

- Serializable (*java.io.serializable*)

Method

- Tutti i metodi pubblici sono esportati

Introspezione

- Design pattern (reflection)
- BeanInfo interface

EJB (Enterprise Java Bean)

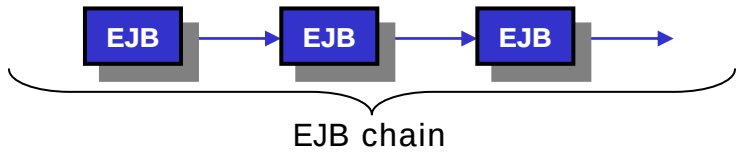
Un *Enterprise Bean* è un componente software server-side.

Può essere composto di uno o più oggetti, a fronte di un'unica interfaccia con cui il *client* può interagire.

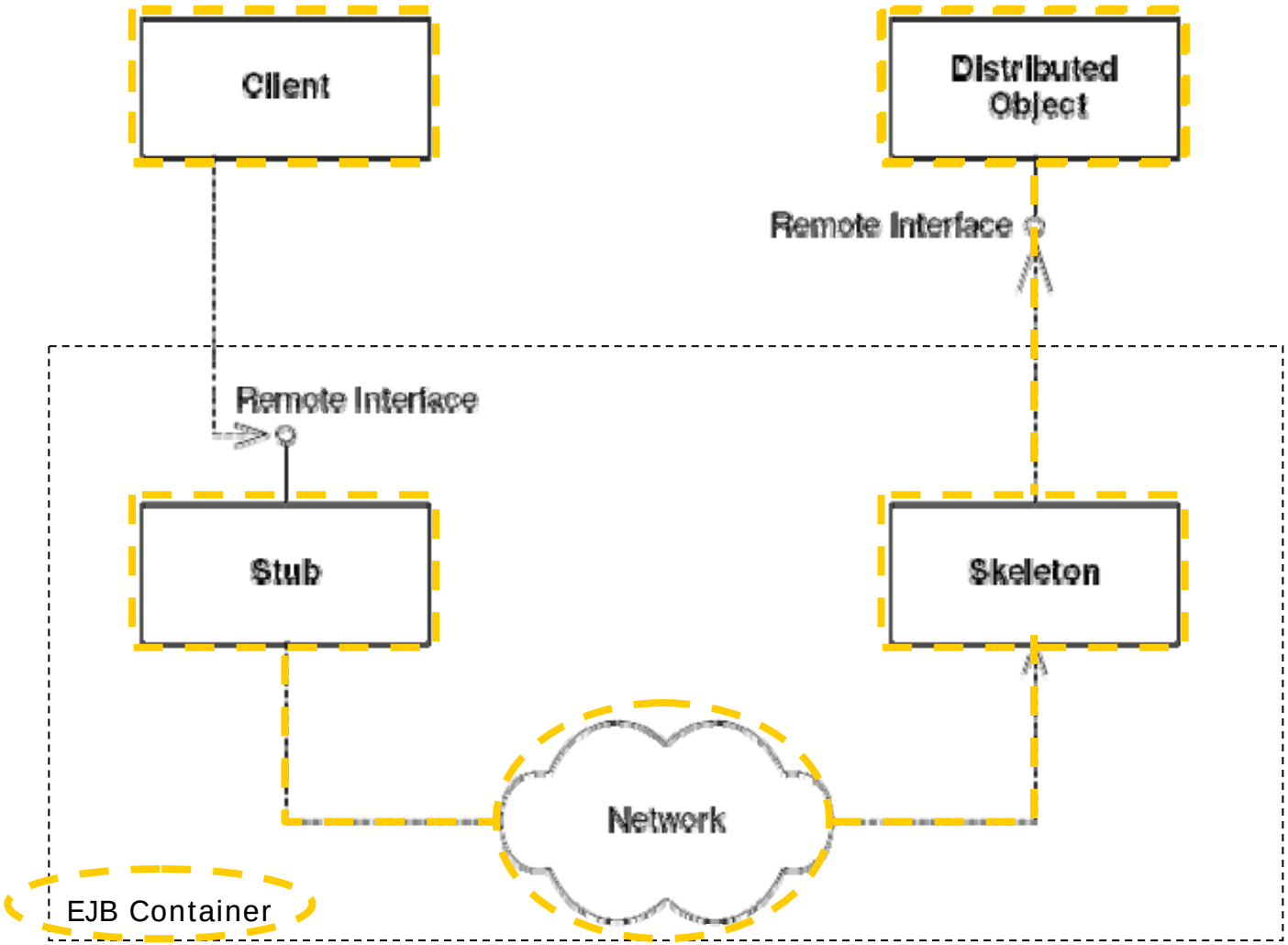
- Catena di EJB

Necessitano di un apposito *container,* e quindi di un *application server EJB-compliant*

- Ambiente per EJB
- Persistenza, transazioni, sicurezza, connessioni
- Accesso alle risorse esterne (DB,..)



EJB: middleware implicito



EJB: Session Bean

Modellano un processo, un'azione

- Accedere ad un DB
- Collegarsi ad un legacy system
- Chiamare altri EJB.

Solitamente non persistente

EJB: Entity Bean

Modellano i dati

Fungono da contenitore per le informazioni di un database

- Un prodotto
- Un ordine
- Un operaio
- Una carta di credito

Sono usati dai *session bean*

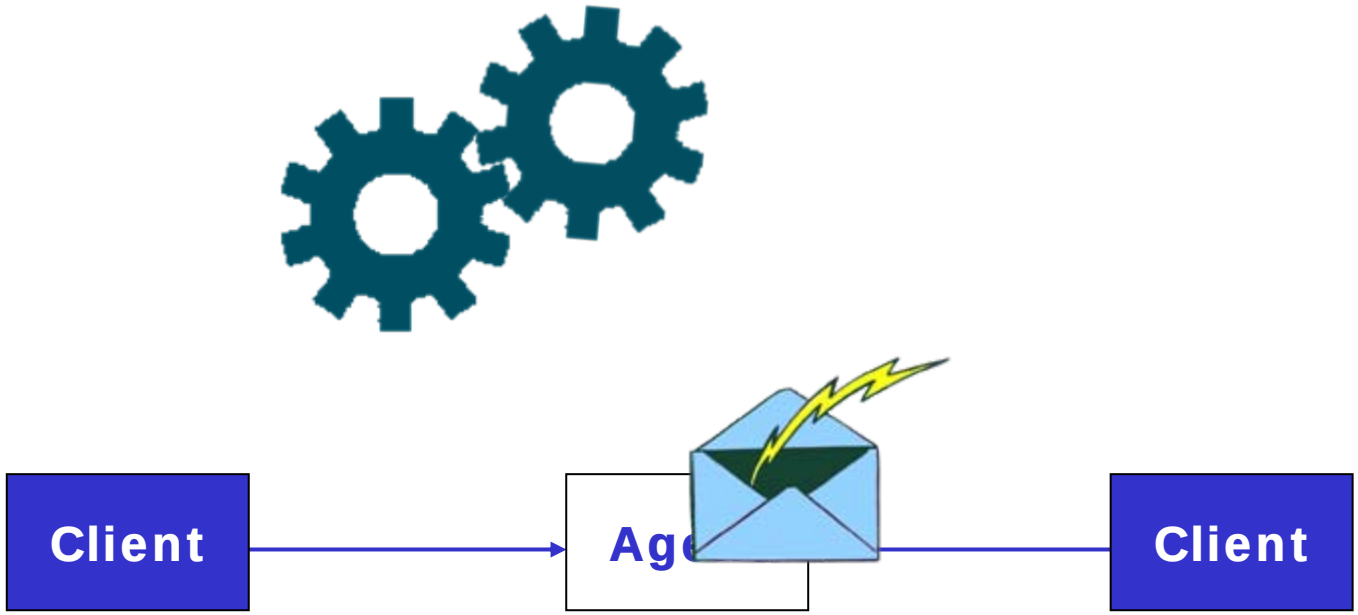
Gestiscono la persistenza

EJB: Message-Driven Bean

Modellano le azioni

Comunicano tramite messaggi JMS

Esecuzione del processo asincrona e separata dalla chiamata.



Message-Driven Bean: caratteristiche

Non hanno interfaccia locale o remota

- Comunicano solo con messaggi
- Qualsiasi clienti che implementi JMS può usarli

OnMessage()

- Analisi a run-time del messaggio

Nessun valore di ritorno

Nessuna *eccezione* verso il client

- Il container cattura *eccezioni* di sistema

Stateless

Conservazione dei messaggi per destinatari non pronti

Impianti Informatici

POLITECNICO DI MILANO

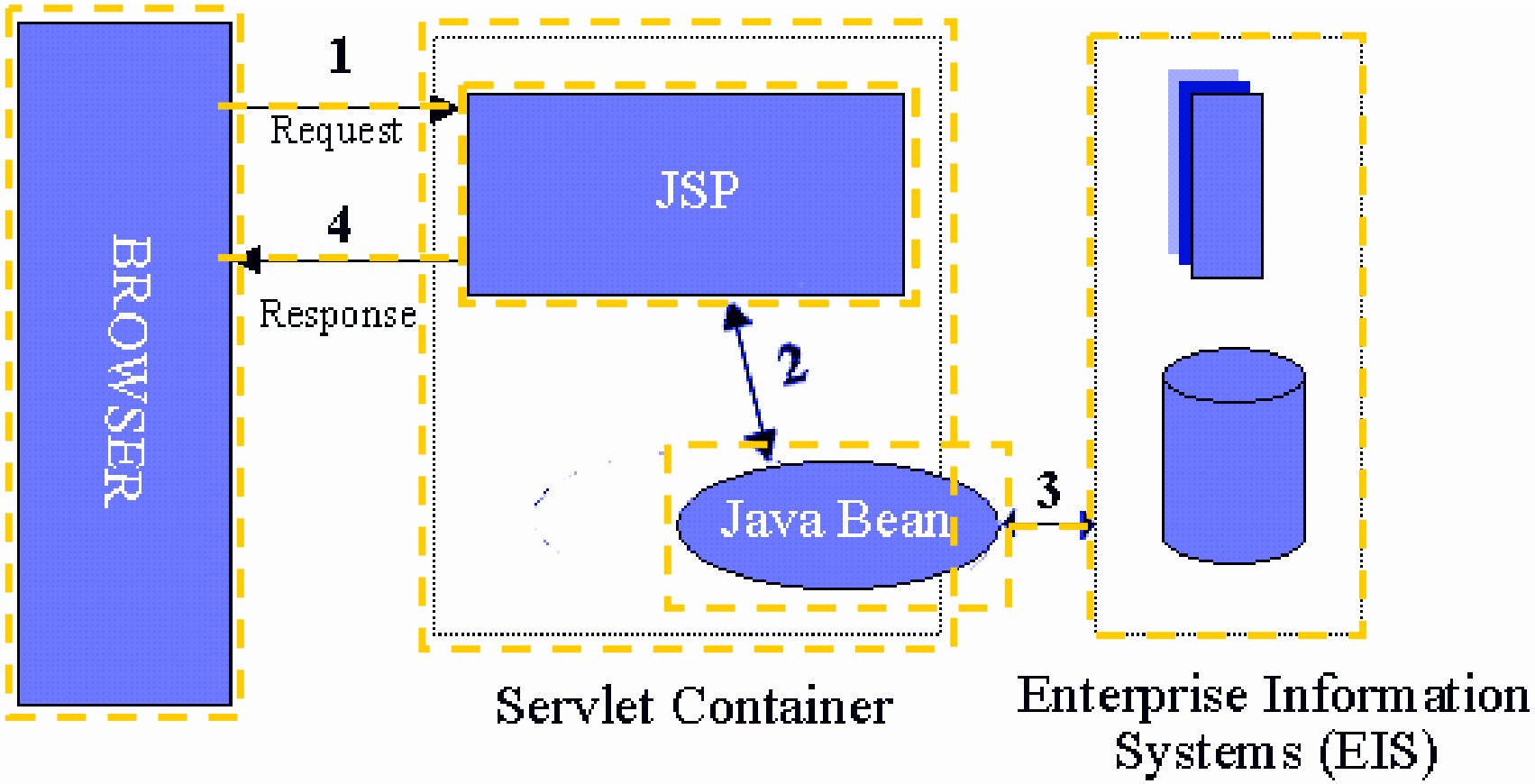




Web application - tecnologie

Model 1

Compilate Java-enabled
Write-once
Run anywhere

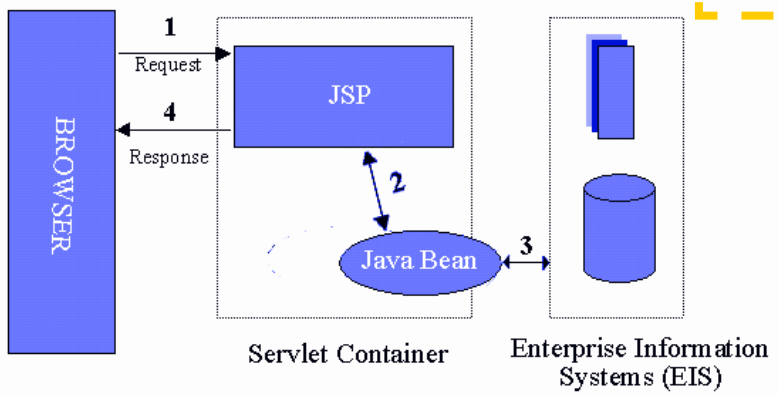


Model 1: problematiche

Accoppiamento dati-pagine

Accoppiamento JAVA-HTML

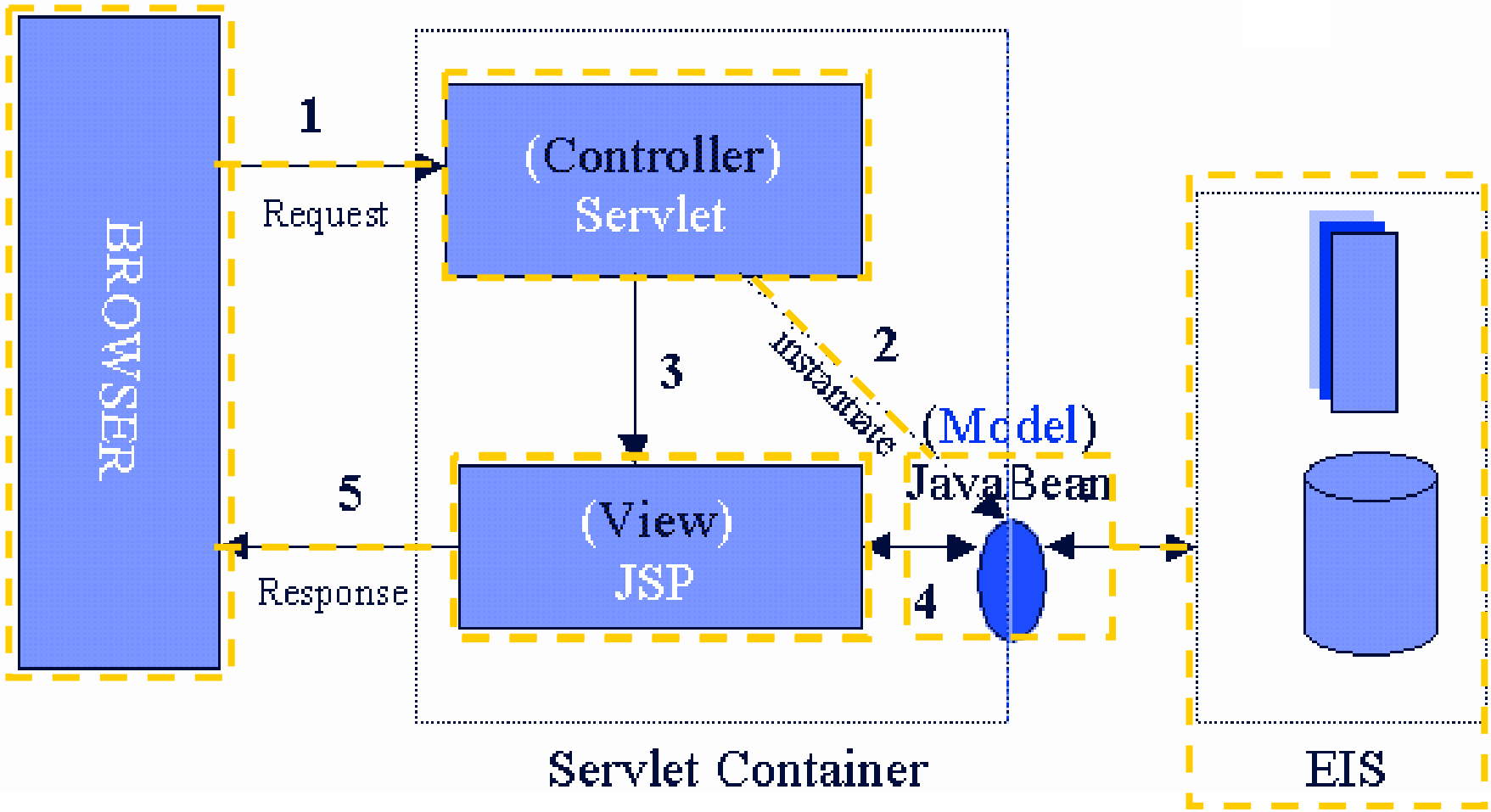
Difficoltà di debug



Flusso logico integrato

JavaScript vs Java

Model-View-Controller (Model 2)



MVC: Model

Business Logic

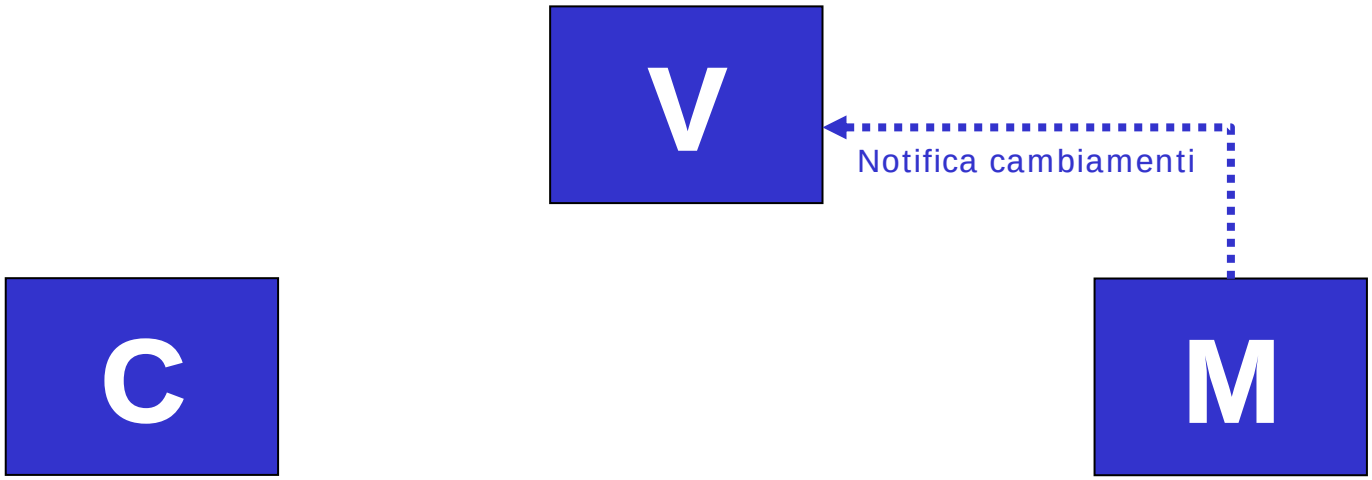
Indipendente da *view* e *controller*

Viene realizzato mediante i *JavaBean*

- contengono la maggior parte della logica dell'applicazione

Incapsula lo stato dell'applicazione

Notifica i cambiamenti alle *view*



MVC: View

Visualizza le informazioni all'utente

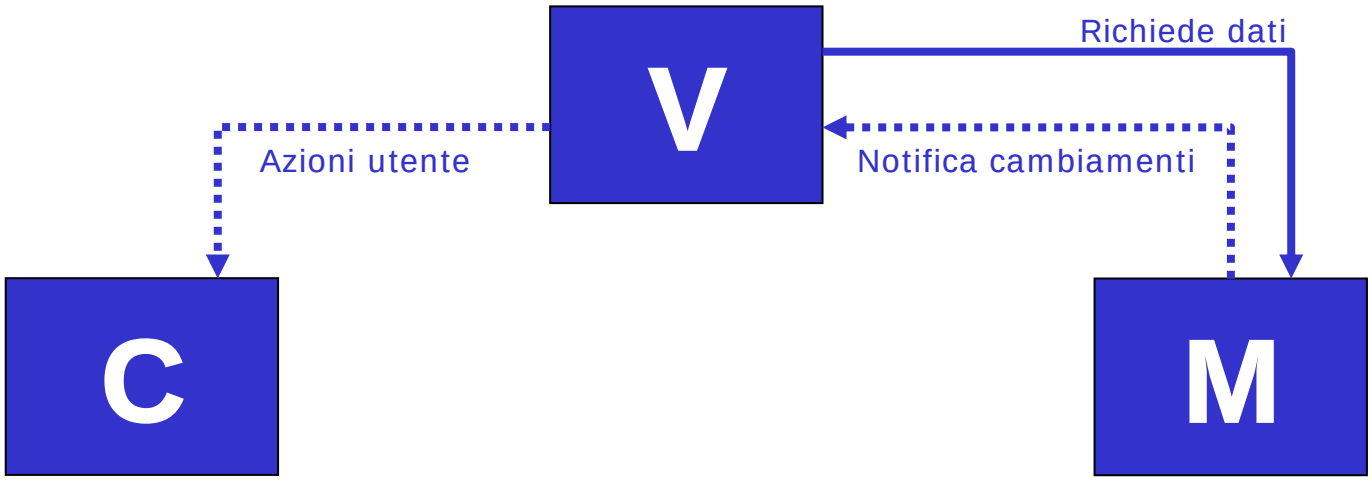
- Pagina di risposta alla richiesta dell'utente

Corrisponde alle *JSP*

- Rendering del modello

Richiede i dati aggiornati al *model*

Invia le azioni dell'utente al *controller*



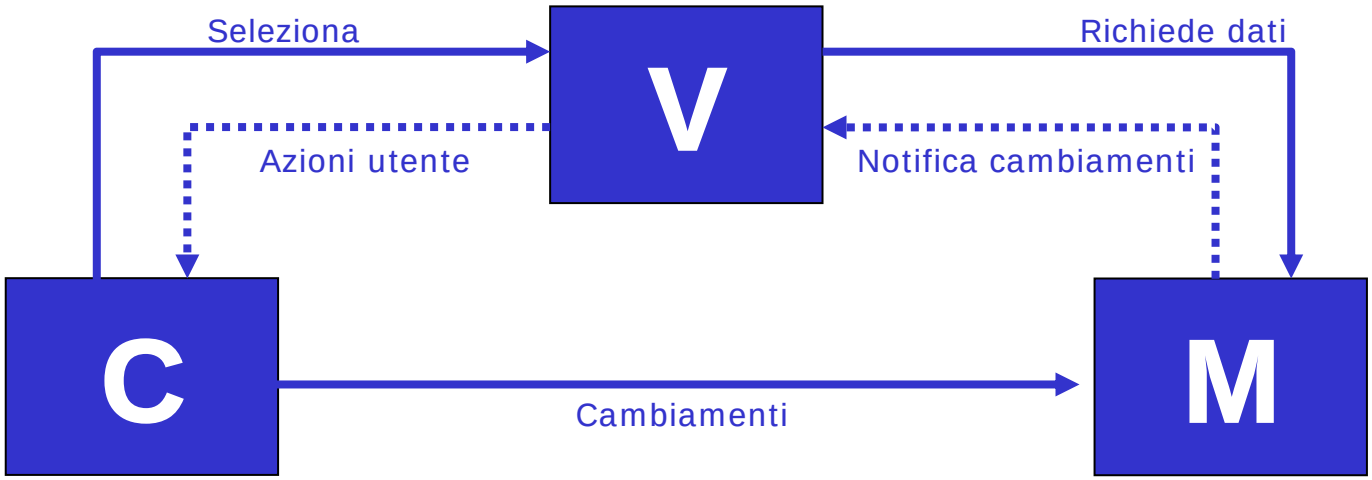
MVC: Controller

Gestisce l'input dell'utente

- Intercetta richieste HTTP

Implementato con *Servlet*

- Comportamento dell'applicazione
- Modifica il *model*
- Seleziona la *view* successiva



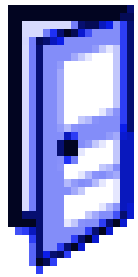
MVC: vantaggi

Facilità sviluppo e gestione applicazioni

- Netta separazione tra *business logic*, *presentation* e *request processing*

Unico punto di ingresso: il *controller*

- Le applicazioni sono generalmente più semplici da mantenere e più facilmente estendibili
- Sicurezza, validazione dell'input, internazionalizzazione



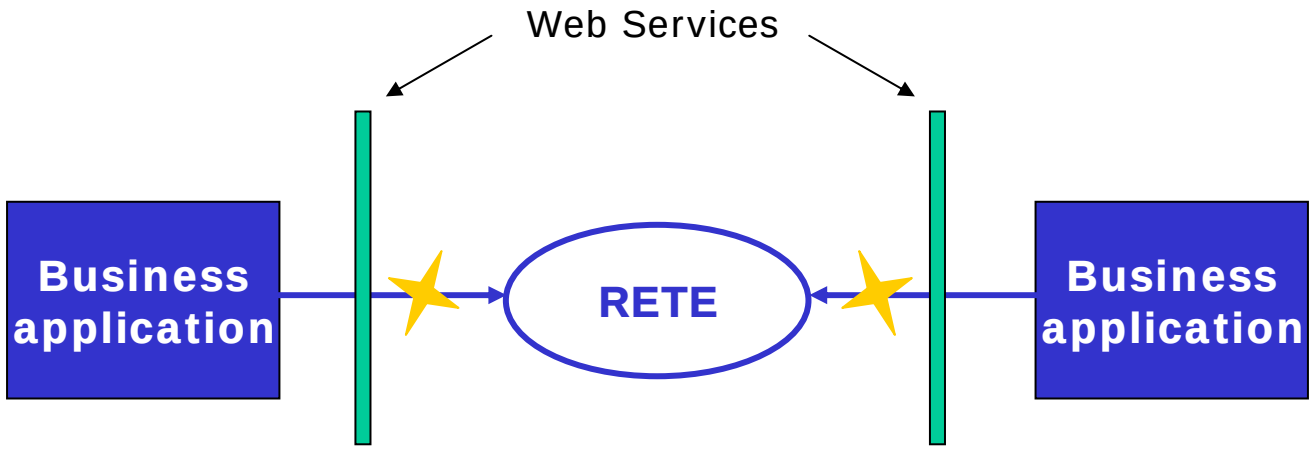
Web Services

Servizi offerti via Web

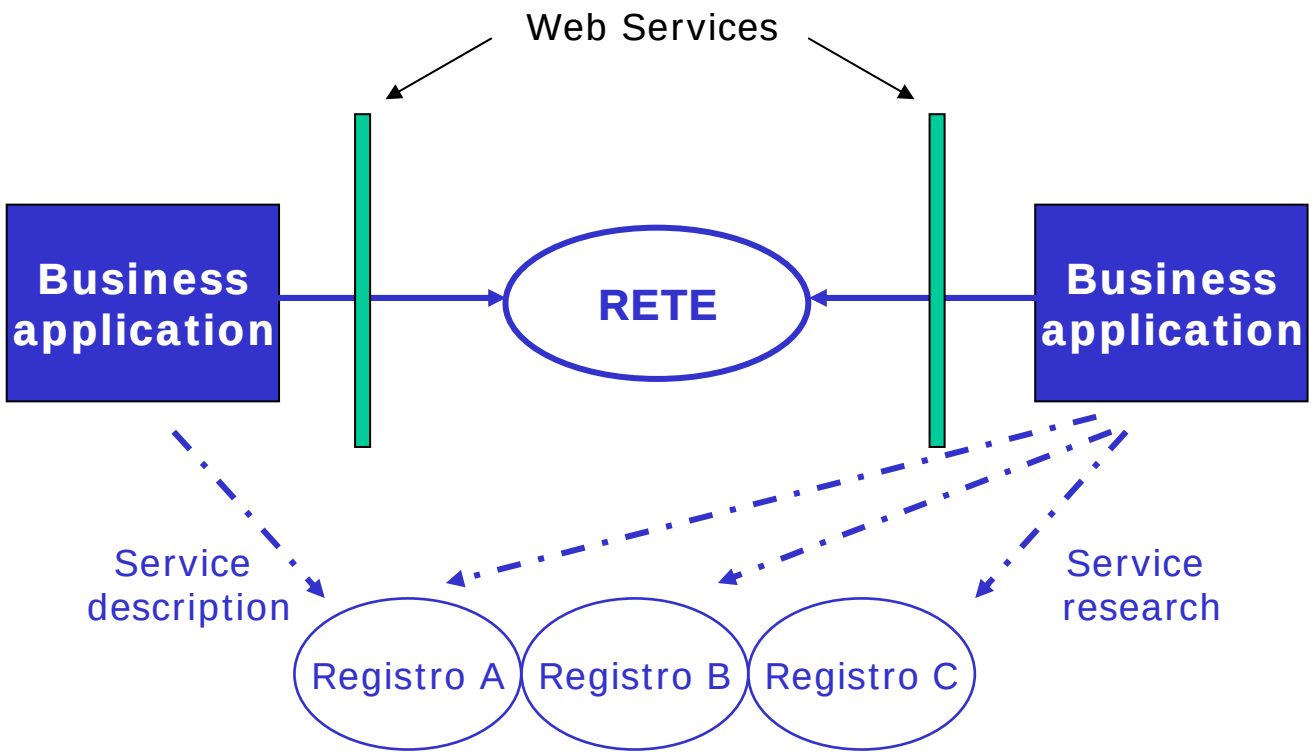
- Applicazioni B2B

Sono una nuova tecnologia di *middleware distribuito* basata su XML (E**x**tensible **M**arkup **L**anguage)

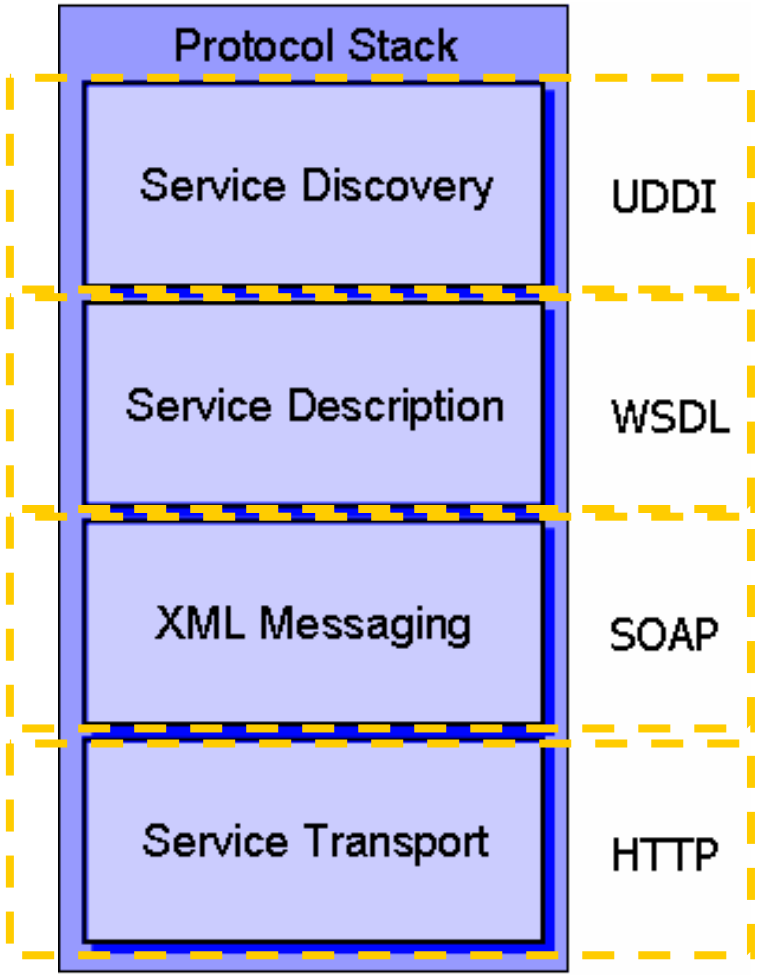
- Opportunità di integrazione tra le applicazioni business



Web Services: opportunità



Web Services: stack protocollare



SOAP (Simple Object Access Protocol)

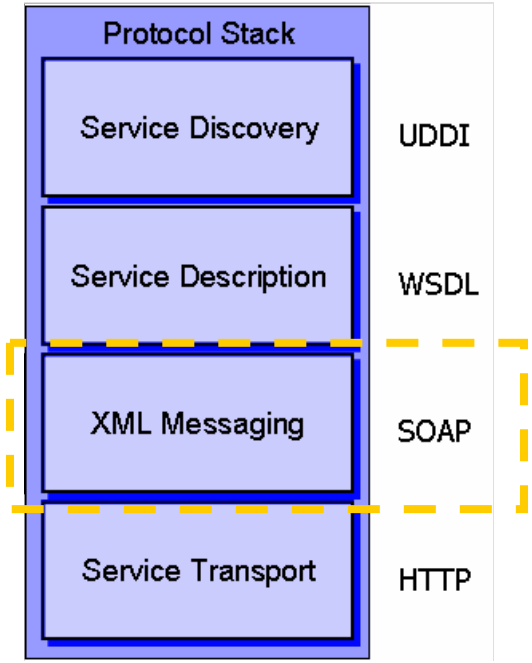
Lightweight protocol

- Modalità di passaggio di dati in XML

Interfaccia con HTTP

Indipendenti da

- sistema operativo
- protocollo di comunicazione
- linguaggio di programmazione

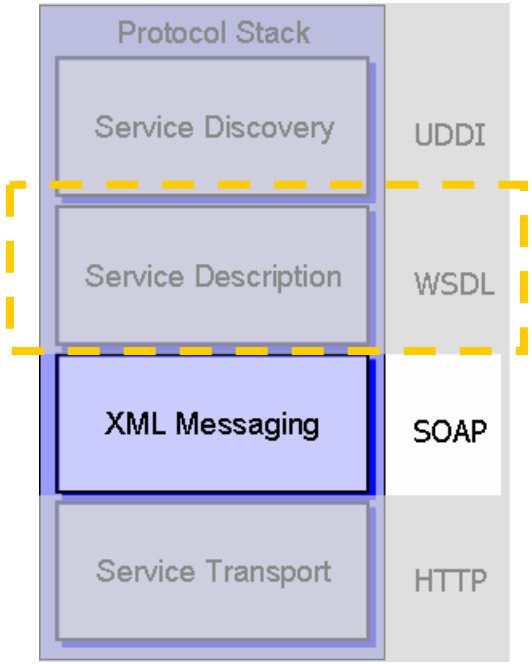


WSDL (Web Services Definition Language)

Interfaccia servizi

Modalità e protocolli di accesso

- Parametri in ingresso



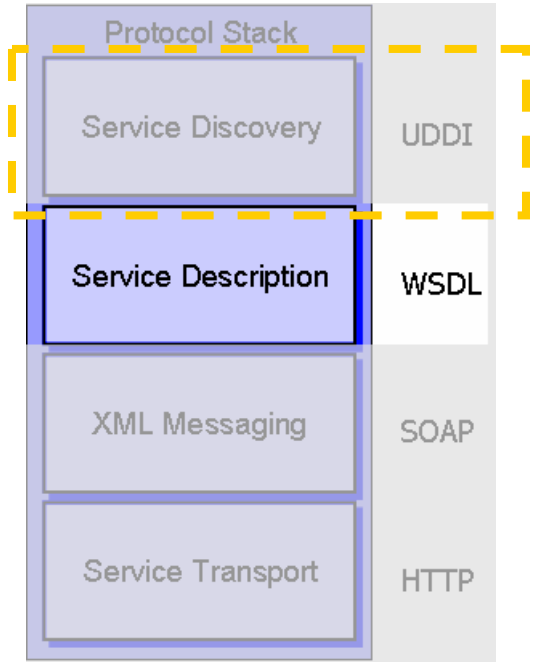
UDDI (Universal Description Discovery and Integration)

Sistema di directory distribuito

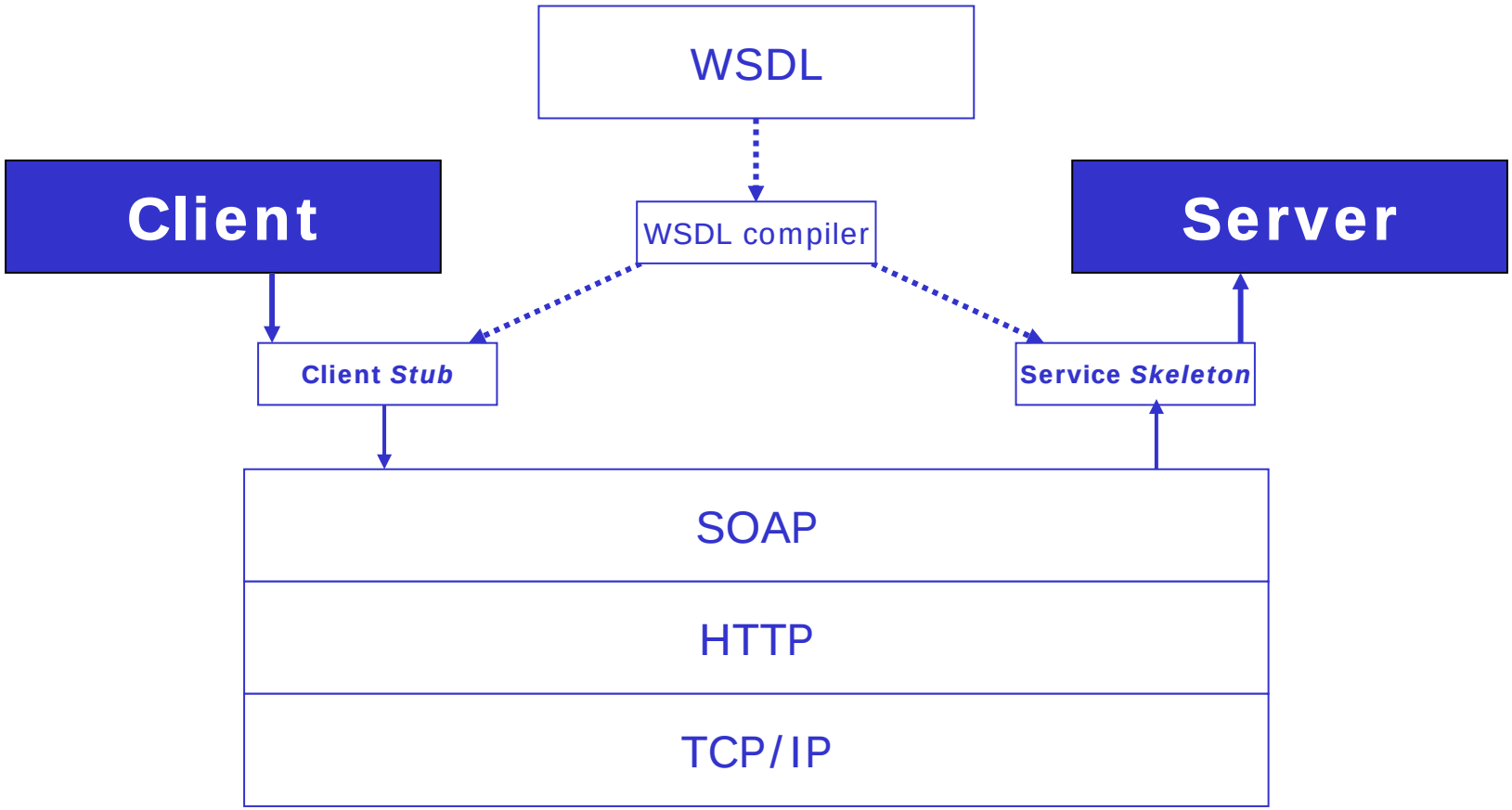
- Registri pubblici
- Registri privati

Tipi di registri

- White pages
  - Informazioni sul provider
- Yellow pages
  - Tassonomia standard dei servizi e delle organizzazioni registrate
- Green pages
  - Informazioni tecniche sui servizi



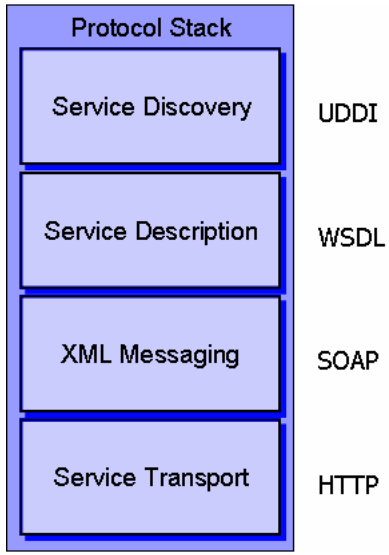
Web Services: data flow



Web Services: pro

Operazioni asincrone

Protocolli standard



Interfaccia con Business logic

Applicazioni disaccoppiate

Modularità

Supporto al codice

Web Services come middleware