TecWeb

Umberto Elias De Angelis

1 Introduzione

1.1 Browser

Un'applicazione browser è un client HTTP, che quindi prepara richieste HTTP da inoltrare al server web e interpreta le relative risposte. Tra le funzionalità essenziali di un browser sta la visualizzazione di oggetti web, come ad esempio documenti HTML.

I browser implementano il lato client dell'HTTP: per questo spesso vengono anche indicati semplicemente come client. I browser non sono l'unico client HTTP possibile. Alcuni programmi, come i bot o i crawler, anche possono agire come client. Pertanto, il termine più generale di "user agent" viene spesso utilizzato per riferirsi al client di un'applicazione web.
Un server web funge da deposito di oggetti web, ognuno dei quali è indirizzabile tramite un URI, e implementa il lato server di HTTP.

1.2 Ipertesto

Un ipertesto è un testo a cui sono stati aggiunti archi con direzione o link che rappresentano associazioni. E' quindi navigabile, consentendo di proseguire la lettura o seguire i link. I link possono essere interni (verso parti del documento di origine), o esterni (verso documenti diversi).

La struttura di un ipertesto è simile ad un grafo con direzione, con nodi che rappresentano oggetti web (immagini, suoni, filmati) e gli archi che rappresentano link. Tale struttura a grafo con direzione è ereditata dal web, che può essere considerato come un grande ipertesto in cui i link esterni vengono seguiti tramite richieste HTTP.

1.3 World Wide Web

Il World Wide Web è un'applicazione client/server che funziona su Internet e reti intranet utilizzando il protocollo TCP/IP.

Le applicazioni web vedono la rete come un fornitore di servizi.

1.4 Terminali, client e server

I computer collegati a Internet vengono chiamati host, terminali o end system in modo equivalente. Un'applicazione client è un'applicazione che utilizza un servizio da un programma server. Il client e il server possono essere eseguiti su macchine diverse o sulla stessa macchina. Questo modello client/server è la struttura predominante per le applicazioni Internet.

Tipicamente, il client e il server vengono eseguiti su macchine diverse. In altre parole, sono applicazioni distribuite che interagiscono tra loro scambiandosi messaggi attraverso Internet.

Non in tutte le applicazioni un programma esclusivamente client interagisce con un programma esclusivamente server. Nelle applicazioni peer-to-peer per la condivisione di file, ad esempio, l'applicazione può agire sia da client (quando richiede un file) che da server (quando fornisce file ad altri utenti).

1.5 Web

Il web non è altro che una delle molte applicazioni distribuite che utilizzano il servizio di comunicazione fornito da Internet.

1.6 Protocolli

I protocolli sono utilizzati per gestire tutte le attività in Internet che coinvolgono almeno due entità remote che comunicano tra loro. I protocolli definiscono il formato e l'ordine dei messaggi scambiati tra le entità, nonché le azioni che si verificano in seguito alla trasmissione o alla ricezione di un messaggio o di altri eventi.

Tra i più importanti protocolli di Internet ci sono:

- TCP (Transmission Control Protocol)
- IP (Internet Protocol): specifica il formato dei pacchetti scambiati tra router e terminali
- HTTP (HyperText Transmission Protocol)

1.7 Pagina Web

Una pagina web è un insieme di oggetti web correlati, come testo, immagini e video, che vengono visualizzati insieme. Un oggetto web è una risorsa, di solito un file, che ha un indirizzo specifico chiamato URL.
Esempi di oggetti web includono file HTML, immagini JPEG o GIF, applet Java, clip audio, e così via.

1.8 URI, URN, URL

Le pagine web spesso includono un documento HTML principale e diversi oggetti che sono indirizzati tramite corrispondenti Uniform Resource Identifier (URI).

Un URI ha una struttura composta da uno schema e dei dettagli specifici.

Il termine URI è più ampio del termine URL in quanto comprende anche gli Uniform Resource Names (URN). La differenza tra URN e URL può essere paragonata alla differenza tra il nome di una persona e il suo indirizzo: un URN serve solo a identificare un oggetto in modo univoco, senza fornire informazioni su come accedervi. Al contrario, un URL fornisce l'informazione su come raggiungere una copia locale di quella risorsa.
Quindi, URN e URL forniscono informazioni complementari.

1.9 User agent

Il concetto di user agent rappresenta l'interfaccia tra l'utente e l'applicazione, come ad esempio un browser, Outlook o Eudora.

Nel contesto del web, ci sono altri tipi di user agent come i sistemi di information retrieval, come i crawler, che cercano informazioni, e gli interpreti VoiceXML, che funzionano come i browser ma interagiscono esclusivamente attraverso la voce con l'utente.

È importante notare che in ogni istante la connessione è stabilita con un singolo server, anche se due interazioni successive possono coinvolgere due server distanti tra loro.

1.10 Programmazione lato server

Le richieste possono riguardare semplicemente oggetti web, ma spesso sono più complesse e coinvolgono l'accesso a un database.

È possibile che la richiesta sia già stata elaborata dal client, ma viene comunque inviata al server che la elabora e la inoltra al database.
Successivamente, il server prepara la risposta del database in una pagina web e la invia al client.

Approcci più recenti includono l'uso di servlet e portlet su server basati su Java. Un vantaggio di tali approcci è che non dipendono dalle funzionalità del browser.

1.11 Intranet

Lo sviluppo di applicazioni per una intranet offre un ambiente più controllato, in cui il tipo di piattaforme coinvolte è generalmente noto in anticipo.
Inoltre, nelle intranet è comune trovare del codice legacy sviluppato per applicazioni tradizionali da portare sul web.

1.12 La sicurezza sul WWW

Scaricare ed eseguire programmi automaticamente da Internet comporta il rischio di diffusione di virus.

Le firme digitali possono rappresentare una soluzione poiché certificano l'autenticità dell'autore.
Tuttavia, le firme digitali non sono una soluzione per i bug non intenzionali e, inoltre, il tempo trascorso tra il download del codice e la scoperta del danno può rendere impossibile identificare l'autore.

Gli utenti delle applicazioni web, spesso non esperti di informatica, potrebbero non essere in grado di adottare le contromisure necessarie. Gli attacchi possono essere di due tipi:

- Passivi: includono l'intercettazione del traffico di rete tra un browser e un server per accedere a informazioni riservate.
- Attivi: comprendono la simulazione di altri utenti, la modifica dei messaggi in transito tra client e server e l'alterazione delle informazioni contenute in un sito web.

2 Protocollo HTTP

Il protocollo HTTP è un'applicazione client-server.
Una caratteristica fondamentale di HTTP è la sua natura "senza stato". Ciò significa che né il server né il client mantengono informazioni sullo stato della comunicazione tra le richieste.
Se un'applicazione web ha bisogno di mantenere informazioni sullo stato della comunicazione, deve implementare meccanismi specifici per farlo.

HTTP, come protocollo di strato di applicazione, deve definire:

- tipo dei messaggi scambiati (richiesta e risposta)
- sintassi dei messaggi
- semantica dei diversi campi
- regole che determinano quando e come un processo invia messaggi o risponde a messaggi.

2.1 Tipi di messaggio

HTTP prevede due tipi di messaggio:

1. richiesta dal client al server
2. risposta dal server al client

Entrambi sono composti da:

- Una prima riga (request line nella richiesta e status line nella risposta)
- Righe di intestazione (anche nessuna)
- Una riga vuota
- Il body del messaggio

2.1.1 Messaggio di richiesta

La request line possiede tre campi:

1. metodo:
    - GET: richiede oggetto
    - POST: invia informazione
    - HEAD: come GET, ma restituisce solo la intestazione
    - PUT: carica oggetto
    - DELETE: cancella un oggetto
    - etc.
2. URL: oggetto richiesto
3. versione HTTP

Seguono le linee di intestazione come ad esempio:

- Host
- Connection: close significa che non usa connessione persistente
- User-agent

Dopo una riga vuota segue il body.

2.1.2 Messaggio di risposta

La prima riga viene detta status line e ha tre campi:

1. Versione del protocollo
2. Codice di stato
3. Corrispondente messaggio di stato

Per quanto riguarda i codici:

- 1xx. (Informativo): la richiesta è stata ricevuta, l'elaborazione continua;
- 2xx. (Successo): la richiesta è stata non solo ricevuta, ma anche compresa e accettata;
- 3xx. (Redirezione): sono necessarie ulteriori operazioni per portare a buon fine la richiesta;
- 4xx. (Errore lato client): la richiesta contiene errori sintattici e non può quindi venire accolta;
- 5xx. (Errore lato server): il server non è in grado di soddisfare una richiesta che appare corretta.

2.2 Meccanismo di cache

Un meccanismo di cache consente di conservare l'ultima copia di una pagina web e scaricarla solo quando necessario, anziché scaricare l'intera pagina ogni volta che si accede al sito.

Per utilizzare efficacemente il meccanismo di cache, il client deve essere in grado di verificare se la pagina richiesta è stata modificata o se la copia memorizzata nella cache è ancora valida.
Per fare ciò, HTTP utilizza il campo "If-Modified-Since" nell'intestazione della richiesta e il codice di stato "304 Not Modified" nella status line della risposta. In questo modo, è possibile richiedere il download della pagina solo se è stata effettivamente modificata.

2.3 Session tracking

L'adozione di strategie di session tracking è resa necessaria dal fatto che il protocollo HTTP è senza stato. L'HTTP fornisce due meccanismi a sostegno del session tracking:

- autenticazione: il codice 401 della status line e i campi per la gestione dell'autorizzazione
- cookies: i campi Set-Cookie e Cookie per lo scambio dei cookies

2.3.1 Session tracking via cookies

Un cookie è un pezzo di testo scambiato tra client e server che contiene informazioni essenziali:

- host e percorso dell'applicazione
- data di scadenza
- nome e valore del cookie
- etc.

Il server crea sempre il cookie e lo apssa al client tramite il campo "Set-cookie" nell'intestazione della risposta HTTP. Il client può decidere di salvarlo.

Ogni volta che lo user-agent (ad esempio, un browser) prepara una richiesta per un'applicazione specifica identificata dall'host e dal percorso, inserisce nell'intestazione della richiesta tutti i cookie che ha memorizzato per quella combinazione di host e path.
Quindi, se viene ripetuta la stessa richiesta, il browser aggiungerà una riga nell'header contenente i cookie corrispondenti.

Una volta che il server riceve il cookie dal client, evita di inserire il campo "Set-Cookie" nella risposta. È importante notare che la decisione di includere o meno questo campo nell'intestazione della risposta non può essere eseguita automaticamente dal server, ma richiede una programmazione specifica sul lato del server.

L'expiration date consiglia allo user agent quando cancellare il cookie. Se negativo, va cancellato subito. Se il server manda un cookie con le stesse informazioni, il cookie viene sovrascritto. Per cancellarlo, il server invia un nuovo cookie con lo stesso nome di quello da cancellare, in modo che questo venga prima sovrascritto e poi cancellato.

2.3.2 Session tracking via autenticazione

E' un metodo utilizzato per tenere traccia delle sessioni degli utenti. Si basa sull'autenticazione dell'utente attraverso un token di sessione unico, che viene associato all'utente e inviato al server.
Ciò permette al server di mantenere lo stato dell'utente e limitare l'accesso alle aree protette.

E' importante garantire la sicurezza del token di sessione per prevenire attacchi.

3 HTML

HTML è un linguaggio di markup per la creazione di pagine web.
Le etichette in HTML sono utilizzate per dare indicazioni sulla formattazione del testo.

Un documento HTML è composto da elementi, che sono parti del documento racchiuse tra le etichette di apertura e chiusura.
Gli elementi differiscono dalle etichette, poiché rappresentano una parte effettiva del documento, mentre le etichette sono semplicemente nomi che caratterizzano il tipo di elemento.

HTML non fornisce indicazioni precise sull'aspetto grafico, ma si concentra sulla funzione delle diverse parti di testo in un documento.
Per definire dettagliatamente l'aspetto grafico, è necessario utilizzare i fogli di stile, in particolare il Cascading Style Sheet (CSS).

In altre parole, HTML descrive la struttura del documento, mentre CSS descrive il layout grafico o sonoro del documento stesso.

Oltre alle indicazioni sulla struttura del documento, l'HTML fornisce anche delle etichette per interagire con l'utente.
L'utente può generare eventi a cui vengono associate diverse azioni. Le form sono uno dei modi per inviare parametri al server, ma non l'unico.

Di solito, la presenza di una form in un documento HTML richiede la programmazione lato server per implementare un programma che riceva i parametri raccolti dalla form e li elabori in modo appropriato.

3.1 Attributi "name" e "id"

L'attributo "name" determina il nome del parametro che viene passato al server. È possibile avere più parametri con lo stesso "name". D'altra parte, l'attributo "id" deve essere univoco, quindi al di fuori delle form è consigliabile utilizzare sempre "id" anziché "name". Infatti, l'uso di "name" è considerato deprecato.

3.2 XML e XHTML

XML è un linguaggio di markup estendibile utilizzato per descrivere dati.
A differenza di HTML, le etichette in XML non sono predefinite, ma possono essere scelte dall'utente per strutturare i dati in modo personalizzato. Questo perchè è stato progettato per descrivere dati.

XHTML è una variante dell'HTML che segue la sintassi dell'eXtensible Markup Language o XML

3.3 HTML5

HTML5 introduce modifiche significative che possono causare problemi di compatibilità con le versioni precedenti di HTML. Gli user agent devono essere in grado di gestire sia HTML5 che le versioni precedenti.

In HTML5, alcuni elementi sono stati esclusi, il che significa che gli sviluppatori web devono evitarli, gli user agent, invece, devono ancora essere in grado di trattarli correttamente, mantenendo il loro significato nelle versioni precedenti.
Questo divide i requisiti in due parti:

- requisiti per gli autori
- requisiti per gli user agent

HTML5 è stato introdotto principalmente per:

- ridurre la necessità di plugins esterni come flash
- migliorare la gesitone degli errori
- indipendenza dal device
- etc.

Si noti che nei documenti di specifiche dell'HTML5, col termine "semantico" si riferisce alla descrizione della struttura del documento anziché ai contenuti.
Si contrappone alla presentazione del documento. Di conseguenza, molte etichette che descrivono la formattazione del documento sono state completamente rimosse.

Un'importante caratteristica di HTML5 è la capacità di memorizzare dati localmente all'interno del browser, come alternativa ai cookie. Questi dati sono salvati come coppie di stringhe nome=valore. A differenza dei cookie, questi dati non vengono inviati in ogni richiesta HTTP al server, ma solo quando è necessario.

3.4 Sintassi

A partire dalla versione 5, un documento HTML può seguire una tra due possibili sintassi:

- text/html
- application/xhtml+xml o application/xml

3.5 Same-origin policy

Gli user agent, eseguono azioni basate sui contenuti scaricati dai siti web.
Tuttavia, poiché alcuni siti possono fornire contenuti malevoli o dannosi, gli user agent devono applicare politiche di sicurezza per proteggere le informazioni di cui sono responsabili. Occorre quindi cautelarsi, visto che in particolare si potrebbe correre il rischio di fornire informazioni riservate.

Gli user agent, di solito, affrontano questo problema utilizzando il concetto di origine per implementare la politica di same-origin (stessa origine). Secondo questo principio, i contenuti possono interagire liberamente tra loro solo se provengono dalla stessa origine, mentre l'interazione tra contenuti provenienti da origini diverse è limitata o vietata. La same-origin policy si basa sulla fiducia accordata a un sito in base al suo URI.

Due URI sono considerati appartenere alla stessa origini se hanno uguali

- schema
- host
- porta

3.6 SQL Injection

L'SQL Injection è una vulnerabilità delle applicazioni web in cui un attaccante sfrutta input non validato per inserire codice SQL malevolo all'interno delle query eseguite dal database.
Ciò consente all'attaccante di manipolare le query, ottenere dati sensibili, modificare o cancellare dati nel database, o eseguire altre azioni dannose.

Per prevenire l'SQL Injection, è fondamentale utilizzare parametri preparati o metodi di interrogazione del database che evitano l'inserimento diretto di input non verificato nelle query SQL.

E' importante usare un approccio whitelists-based per prevenire l'sql injection bloccando tutto ciò che non sia esplicitamente previsto. Se al contrario ci si limita a bloccare solo ingressi rinetuni pericolosi (approccio blacklist-based) si rischia di dimenticarsi di casi rari.

3.7 XSS

L'attacco di cross-site scripting (XSS) è una vulnerabilità delle applicazioni web in cui un attaccante riesce a inserire script malevoli in pagine web visualizzate da altri utenti.
Ciò avviene solitamente attraverso campi di input non validati o non correttamente sanitizzati all'interno dell'applicazione. Quando un utente visualizza la pagina compromessa, gli script dannosi vengono eseguiti nel suo browser, consentendo all'attaccante di rubare informazioni sensibili, modificare il contenuto della pagina o eseguire altre azioni dannose a nome dell'utente.

Per prevenire gli attacchi XSS, è importante validare e sanitizzare adeguatamente tutti gli input ricevuti dall'utente e utilizzare le funzioni di escape adeguate per evitare l'inserimento di codice HTML o script non autorizzati nelle pagine web.

3.8 Come costruire filtri di validazione whitelist-based

Per evitare i due tipi di attacco citati precedentemente:

- bisogna validare anche elementi che possono sembrare innocui come quelli di tipo img, validando anche gli attributi.
- se si desidera permettere la costruzione di URL, anche lo schema di quest'ultimo deve essere basato su whitelist
- se si permette l'inserimento di un elemento di tipo base, gli script presenti nella pagina possono essere compromessi, incluso l'URL per l'invio dei dati dei form che potrebbero essere rediretti a siti dannosi.

3.9 Cross-site request forgery (CSRF)

Il Cross-Site Request Forgery (CSRF) è una vulnerabilità delle applicazioni web in cui un attaccante sfrutta la fiducia di un utente autenticato per eseguire azioni non autorizzate a suo nome.

L'attacco CSRF si verifica quando un sito web malevolo o compromesso invia una richiesta falsificata a un sito legittimo, sfruttando le credenziali e le sessioni valide dell'utente per eseguire azioni indesiderate.

Per prevenire il CSRF, le applicazioni web devono implementare controlli di sicurezza come l'utilizzo di token CSRF, che garantiscono che le richieste provengano da fonti legittime e non da siti esterni o compromessi.

3.10 Clickjacking

Il clickjacking è un tipo di attacco in cui un sito web dannoso sovrappone o nasconde elementi interattivi su una pagina web legittima, ingannando gli utenti affinché eseguano azioni indesiderate.

Per prevenire il clickjacking, si potrebbe pensare ad un controllo basato sul confronto tra l'oggetto "window" e il valore dell'attributo "top".

4 Java

Java è un linguaggio orientato agli oggetti, questo lo rende semplice e adatto alle applicazioni web, anche se non è un linguaggio di scripting.

Esso possiede una gestione automatica della memoria (garbage collector) che semplifica le operazioni di memoria, evitando errori, in particolare quelli legati alla deallocazione di memoria.

Java è indipendente dalla piattaforma, ovvero viene compilato in un bytecode che viene eseguito dalla Java Virtual Machine. In questo modo:

- è più portabile di un linguaggio compilato;
- è più efficiente di un linguaggio interpretato;
- possiede caratteristiche che un linguaggio interpretato non ha (ex. è fortemente tipato)

La Java Virtual Machine è una macchina astratta e non vengono fatte ipotesi sulla tecnologia in cui viene implementata né sulla piattaforma ospite. Non sa nulla di Java, ma solo del formato class. Un file class (bytecode) contiene istruzioni per la JVM, oltre che interpretati dalla JVM, i file class possono anche essere compilati in file eseguibili da diverse macchine.

Inoltre, Java fornisce un miglior controllo degli errori, in particolare a errori legati a due aspetti:

- Gestione della memoria -¿ ad esempio memoria allocata da liberare.
- Condizioni eccezionali -¿ ad esempio divisioni per 0 o errori di lettura dovuti a mancanza di file di ingresso (che appesantiscono il codice).

E possiede un buon supporto al multi-thread, che può essere utile per l'impementazione di interfacce sofisticate e nella gestione di richeiste concorrenti.

Una piattaforma Java è composta da:

- Linguaggio Java;
- Java Virtual Machine;
- Application Programming Interfaces (librerie API).

4.1 La sicurezza Java

L'aspetto più importante di Java per quel che riguarda il suo uso in applicazioni Web è la sicurezza che dipende da:

- il progetto del linguaggio come sicuro e di facile uso
- verifica del bycode prima dell'esecuzione
- durante l'esecuzione:
    1. le classi vengono caricate e linkate
    2. possibile generazione del codice macchina e ottimizzazione dinamica

3. esecuzione del programma

durante questo processo, il class loader definisce un namespace locale, in modo che il codice non sicuro non possa interferire.

- l'utilizzo di una classe security manager per l'accesso alle risorse del sistema

4.2 Sicurezza del linguaggio

Nel linguaggio Java sono state eliminate caratteristiche che potevano essere pericolose.

Una delle principali è l'impossibilità di accedere alla memoria in modo non controllato, questo grazie all'assenza di aritmetica dei puntatori (non equivale a non supportare una qualche forma di puntatori ovvero le references) Questo permette la riduzione o addirittura l' eliminazione di attacchi tramite accessi non protetti alla memoria per ottenimento di dati privati.

Le specifiche del linguaggio Java definiscono che le variabili:

- sullo heap vengano automaticamente inizializzate
- sullo stack non vengano inizializzate e quindi tale operazione deve avvenire in modo esplicito prima dell'uso

Un altro aspetto di Java che contribuisce alla robustezza è la garbage collection automatica (riguarda solo la heap), ovverò la capacità della macchina virtuale di liberare automaticamente memoria non più utilizzata.

4.3 Sicurezza di macchina virtuale

La macchina virtuale Java usa una sandbox per mandare in esecuzione programmi Java, che:

- verifica classi che entrano;
- opera numerosi controlli a runtime;
- per alzare barriere attorno all'ambiente di runtime e controllare gli accessi all'esterno della sandbox.

La macchina virtuale offre quindi un luogo sicuro per l'esecuzione di programmi. Il peggio che potrà accadere sarà una condizion di DoS, in cui il programma consuma tutte le risorse della CPU.

Il bytecode è il linguaggio macchina della macchina virtuale Java.
E' importante sapere che non è sufficiente implementare costrutti sicuri nel linguaggio, perché il sistema potrebbe comunque essere attaccato da bytecode ostile generato da compilatori java modificati.
E' quindi importante implementare controlli sull'ambiente di esecuzione.

La verifica del bytecode caricato è compito del ClassLoader.
Il ClassLoader non cerca mai di caricare le classi di java da remoto: in questo modo si evita che esse possano venir sostituite con classi modificate.
Quindi, non è possibile la comunicazione all'interno della macchina virtuale da classi provenienti da host diversi.

Ogni macchina virtuale Java in esecuzione ha al più un Security Manager installato, responsabile per la gestione delle politiche di sicurezza (es. restringere o meno vincoli).
Una volta che un manager è installato, non è possibile sostituirlo, visto che questo genererà una eccezione, quindi nessuno può variare questa funzione.

4.4 Applets

Le applet sono applicazioni sviluppate con Java che vengono eseguite all'interno di una pagina web, rappresentando un esempio di programmazione lato client.

Le loro caratteristiche principali sono:

- la portabilità, in quanto possono essere eseguite su diverse piattaforme

- la sicurezza, poiché l'host che esegue l'applet deve avere fiducia nell'autore e accettarne l'esecuzione.

Tuttavia, non possiamo assumere che tutti gli user agent accettino le applet, specialmente se queste rappresentano un potenziale rischio per la sicurezza del sistema.

Le azioni delle applet sono ristrette all'interno di una sandbox, ovvero un'area del browser allocata per l'applet.
Le politiche di sicurezza previste nei browser prevedono che applet scaricate dalla rete:

- non possono leggere o scrivere file sulla macchina ospite
- non possono creare connessioni di rete con host diversi dal server originario
- non possono caricare librerie
- non possono avviare nuovi processi sulla macchina ospite
- non possono avere accesso a informazioni sulla macchina locale
- tutte le finestre aperte da un applet visualizzano un avvertimento

Esistono comunque situazioni in cui queste politiche di sicurezza sono troppo rigide, ad esempio in una intranet aziendale (nelle quali le applet Java sono molto usate per facilitare la gestione). Sono quindi stati introdotti strumenti per gestire le restrizioni.

Le applet vengono richiamate nei file HTML attraverso le etichette <APPLET> e <OBJECT> (che non è supportata su browser più vecchi)

Un altro metodo per distribuire applet attraverso la rete è Java Web Start. Tuttavia, scaricare applet può essere costoso poiché richiede una richiesta separata per ogni classe. Pertanto, è vantaggioso impacchettare tutti i file .class e altri file correlati in un unico file JAR (Java Archive), che può essere scaricato con una singola richiesta.
Inoltre, può essere applicata la firma digitale ai file di un archivio JAR, così da rendere l'applet firmata.

Le applet firmate possono essere abilitate a essere eseguite con molti meno vincolo rispetto ad applet non firmate, e possono prendere il posto delle normali applicazioni.
Tuttavia esse hanno sempre il pesante vincolo di essere eseguite in un browser, il quale causa:

- maggior carico computazionale per il client che deve eseguire anche il browser
- impatto visivo non trascurabile, visto che vengono visti anche i pannelli relativi al browser

4.5 Ciclo di vita di una applet

La classe Applet fornisce quattro metodi principali, che consentono di gestire il ciclo di vita dell'applet;

1. init(): Questo metodo viene chiamato automaticamente dal sistema quando l'applet inizia l'esecuzione. È qui che vengono effettuate tutte le inizializzazioni necessarie.
2. start(): Questo metodo viene chiamato dopo init() e viene eseguito ogni volta che l'utente torna alla pagina contenente l'applet.
3. stop(): Non è necessario chiamare esplicitamente questo metodo. Viene richiamato ogni volta che l'utente esce dalla pagina in cui si trova l'applet. Serve a sospendere le attività che consumerebbero risorse inutilmente.
4. destroy(): Questo metodo viene chiamato quando il browser viene chiuso in modo normale, dopo aver chiamato stop().

4.6 JNLP e Java Web Start

La JNLP (Java Network Launch Protocol), permette di scaricare ed eseguire una applet anche fuori dal browser, permettendo quindi di scaricare dinamicamente risorse da Internet durante l'esecuzione.

Ovviamente, occorre comunque cautela da parte del client nel trattare queste applicazioni, a cui vengono generalmente applicate le stesse restrizioni della sandbox. Come le applet possono essere scaricate

in file JAR firmati, cosìcchè l'utente possa fidarsi di chi firma.

JNLP è comunque un protocollo, a cui serve un'implementazione come ad esempio Java Web Start (JAWS).
Un'applicazione JNLP prevede un file JAR contenente l'applicazione standard e uno XML per indicare al client come scaricare e installare l'applicazione.

5 Servlet

Le servlet sono un esempio di utilizzo di Java per la programmazione web lato server. Offrono diversi vantaggi, tra cui maggiore efficienza, facilità di sviluppo, sicurezza e maggiore condivisione delle informazioni all'interno dell'applicazione e con il server.

Per utilizzare le servlet, è necessario che il server sia predisposto per l'esecuzione delle servlet e disponga di un "servlet engine" o "servlet container" come Tomcat, ad esempio.

Le servlet sono classi Java che possono essere caricate dinamicamente per estendere le funzionalità del server. Vengono eseguite all'interno della Java Virtual Machine sul server, rendendole sicure e portabili. Operano solo sul server e non richiedono il supporto del browser.

Le servlet vengono gestite all'interno del processo del server da thread separati, offrendo vantaggi come l'efficienza, la scalabilità e un'ottima interazione con il server.

Oltre alle servlet, le Java Server Pages (JSP) vengono tradotte in servlet e offrono un sistema per implementare il meccanismo di richiesta e risposta del server senza doversi preoccupare di tutti i dettagli implementativi delle servlet.

- si utilizzano le JSP quando la maggior parte del contenuto inviato al client è testo statico e markup.
- si utilizzano le servlet quando la porzione di testo da generare staticamente è minore. Infatti, la maggior parte delle servlet non genera direttamente testo, ma svolge compiti richiesti dal client e alla fine richiama altre servlet o JSP per produrre una risposta.

Le servlet inviano i risultati al client come file HTML, XHTML, XML e altri formati come immagini e dati binari. Non si limitano solo ai server web, ma possono estendere anche altri tipi di server come quelli di posta e FTP. Tutte le servlet devono implementare l'interfaccia Servlet, che rappresenta l'API standard fornita dai provider di servizi.

I pacchetti delle servlet definiscono due classi astratte che implementano l'interfaccia Servlet:

1. GenericServlet (in javax.servlet);
2. HttpServlet (in javax.servlet.http).

Queste classi forniscono un'implementazione di default per tutti i metodi dell'interfaccia.

5.1 L'interfaccia Servlet

I metodi dell'interfaccia Servlet vengono invocati automaticamente dal servlet engine.
L'interfaccia dichiara quindi cinque metodi, di cui però non dà il body e quindi l'implementazione:

- void init(ServletConfig config): inizializza la servlet una volta sola durante il ciclo di esecuzione della servlet;
- ServletConfig getServletConfig(): restituisce un oggetto che implementa l'interfaccia ServletConfig. Questo oggetto fornisce accesso alle informazioni sulla configurazione della servlet, compresi i parametri di inizializzazione e il ServletContext, che rappresenta l'ambiente in cui la servlet viene eseguita;
- String getServletInfo(): viene definito in modo da restituire una stringa con informazioni quali autore e versione;
- void service(ServletRequest request, ServletResponse response): viene mandato in esecuzione in risposta alla richiesta mandata da un client alla servlet;

- void destroy(): invocato quando la servlet viene terminata; usato per rilasciare le risorse

Il metodo init() viene chiamato quando la servlet viene caricata in memoria, di solito alla prima richiesta per quella specifica servlet. Dopo l'esecuzione di init(), il server può soddisfare la richiesta del client invocando il metodo service(), che elabora la richiesta e prepara la risposta da inviare al client.

Il metodo service() viene chiamato per ogni nuova richiesta ricevuta.
Di solito, per ogni nuova richiesta, il servlet engine crea un nuovo thread di esecuzione in cui viene eseguito il metodo service(). Alla fine, quando il servlet engine termina la servlet, viene invocato il metodo destroy() per rilasciare le risorse della servlet.

5.2 HTTPServlet

La classe astratta HTTPServlet implementa una servlet rispetto al protocollo HTTP.
Ogni sottoclasse di HTTPServlet farà overriding di almeno un metodo, di solito uno dei seguenti:

- doGet, se la servlet viene richiamata col metodo HTTP GET;
- doPost, per richieste HTTP con metodo POST;
- doPut, per richieste HTTP con metodo PUT;
- doDelete, per richieste HTTP con metodo DELETE;
- init e destroy, per gestire eventuali risorse necessarie al funzionamento della servlet
- getServletInfo, usata dalla servlet per fornire informazioni riguardanti se stessa.

Una strategia tipicamente utilizzata è di non riscrivere il metodo service, ma piuttosto i metodi "doXXX". I metodi "doXXX" prendono in ingresso:

- un oggetto ServletRequest, corrispondente ad uno stream di ingresso, da cui leggere la richiesta del client;
- un oggetto ServletResponse, corrispondente ad uno stream di uscita, su cui scrivere la risposta.

In caso di problemi, viene lanciata una ServletException o una IOException.

In caso di richieste multiple, il metodo service() può essere eseguito in parallelo. Pertanto, può essere necessario che i programmatori sincronizzino l'accesso alle risorse utilizzando tecniche di sincronizzazione.

Il metodo service() viene sovrascritto per gestire le diverse richieste che arrivano da un client web. Questo metodo determina la tipologia di richiesta in arrivo e chiama il metodo appropriato per gestirla, come ad esempio doGet() o doPost().

5.3 L'interfaccia HTTPServletRequest & HTTPServletResponse

L'interfaccia HttpServletRequest

I metodi dichiarati da questa interfaccia provengono in parte dall'interfaccia `ServletRequest`.
Un'implementazione dell'interfaccia è data dalla classe HttpServletRequestWrapper.

- `String getParameter( String name )`: torna il valore del parametro *name* passato nella get/post;
- `Enumeration getParameterNames()`: returns an enumeration of string objects containing the names of the parameters contained in this request. "
- `String[] getParameterValues( String name )`: se name ha valori multipli
- `Cookie[] getCookies()`: memorizzati dal server nel client;
- `HttpSession getSession( boolean create )`: se l'oggetto non esiste e `create = true`, allora lo crea.

L'interfaccia HttpServletResponse

- `void addCookie( Cookie cookie )`: nell'header della risposta passata al client; essa viene memorizzata a seconda dell'età massima e dell'abilitazione o meno ai cookie disposta nel client;
- `ServletOutputStream getOutputStream()`: stream basato su byte per dati binari;
- `PrintWriter getWriter()`: stream basato su caratteri per dati testuali;
- `void setContentType( String type )`: specifica il formato MIME.

5.4 Multithreading

Le richieste dei client vengono soddisfatte dal servlet engine facendo uso di un pool di thread. Per evitare collisioni, occorre che il metodo service() sia thread-safe: ogni accesso a risorse comuni quali files e basi di dati va protetto col costrutto synchronized.

6 CMS

Un Content Management System (CMS) è un sistema per la gestione dei contenuti che facilita l'organizzazione del lavoro sui contenuti per un gran numero di utenti.

La maggior parte dei CMS utilizza un database relazionale per archiviare i metadati e i contenuti necessari al sistema. I contenuti sono spesso memorizzati come XML per favorirne il riutilizzo e consentire maggiore flessibilità.

Un CMS offre supporto per la creazione, modifica e pubblicazione dei documenti, consentendo anche la collaborazione tra più persone nella loro creazione. È possibile importare documenti esterni o legacy nel sistema.

Un aspetto importante di un CMS è la definizione e la gestione di diverse categorie di utenti in maniera tale da avere un controllo degli accessi.

Un'altra funzionalità chiave di un CMS è il supporto alla ricerca e al recupero dei documenti utilizzando metodi come query booleane o ricerca per parole chiave. I documenti devono essere organizzati in una repository con indicizzazioni adeguate per consentire il recupero efficace dei contenuti.

Molti CMS includono anche strumenti per il marketing personalizzato, consentendo la personalizzazione delle pagine per utenti specifici, programmi di fidelizzazione e altro ancora.

6.1 Web CMS

Un Web CMS è un CMS che funziona su web e consente l'accesso principalmente tramite un browser. Un esempio di CMS è una wiki.

Gli utenti non richiedono competenze informatiche avanzate, ma solo la conoscenza di base nell'uso del browser.

6.2 Web framework

Un framework non è né un'API, né una libreria, nè un'applicazione, ma uno strumento che offre supporto allo sviluppo di applicazioni, mettendo a disposizione una serie di strumenti e soluzioni integrate già pronte all'uso.

Un web framework, dunque, non è altro che un framework per lo sviluppo di applicazioni specifiche per il web, mettendo a disposizione sia strumenti per la programmazione lato server, che per la programmazione lato client.

I web framework semplificano la programmazione lato server fornendo funzionalità comuni come la gestione del database, la scelta del content-type della pagina e la gestione delle variabili di sessione. Questo riduce la duplicazione del codice e semplifica lo sviluppo.

Un web framework si occupa di standardizzare e risolvere i problemi comuni nello sviluppo web, consentendo agli sviluppatori di concentrarsi esclusivamente sulle caratteristiche significative dell'applicazione.

Il requisito di modularità è fondamentale importanza per la realizzazione di un web-framework.
Inoltre, i Web frameworks offriranno supporto per tutte quelle che sono caratteristiche tipiche di un'applicazione web, inclusi per esempio il supporto al session-tracking e all'autenticazione dell'utente.

6.3 Architettura delle applicazioni web

Un'architettura (software) descrive la struttura di un'applicazione, la decomposizione in componenti, le loro interfacce e interazioni.
L'architettura rappresenta il passaggio tra analisi e implementazione.
Le applicazioni Web sono un esempio di architettura multi-tier.
Per tier si intende un raggruppamento logico di funzionalità. I diversi tier possono stare sulla stessa macchina o su macchine diverse.
Le applicazioni web più semplici sono three-tier: information source $\longleftrightarrow$ server web $\longleftrightarrow$ user-agent.
Il tier intermedio controlla l'interazione tra client e sorgente di informazione e implementa:

- il meccanismo di controllo (controller logic): elabora la richiesta del client e si procura i dati da presentargli;
- il meccanismo di presentazione (presentation logic) elabora questi dati per presentarli all'utente (in HTML ad es.)
- le regole del dominio applicativo (business logic) e si assicura che i dati siano affidabili prima di usarli, le business rules regolano l'accesso dei client ai dati e l'elaborazione dei dati dell'applicazione.

6.4 ModelViewController

Il modello MVC è un pattern architetturale three-tier. Se usato nel web, non vi è alcuna relazione tra modello e vista, a causa del fatto che la comunicazione tra essi è veicolata da richieste HTTP che è un protocollo stateless.

I tre livelli di astrazione del pattern WebMVC sono delegati ad assolvere specifiche funzionalità:

- Model: centralizza il controllo e l'accesso ai dati. Usualmente lo scambio dei messaggi tra il livello "model" e "controller" è veicolato da un ORM (Object Relational Mapping) responsabile del mapping tra modello dei dati e corrispondente modello ad oggetti dell'applicazione.
- View: è responsabile della composizione e visualizzazione delle risorse, spesso rappresentate da pagine HTML, ma anche JSON, XML, etc. A tale scopo, questo livello utilizza librerie javascript per la programmazione lato client.
- Controller: ha la responsabilità di acquisire le richieste HTTP, gestire la comunicazione con il modello dei dati e di trasferire i dati alla "vista" per la compilazione delle risorse. Inoltre si occupa dell'indirizzamento degli URL e della gestione dei meccanismi di sicurezza.

6.5 Fullstack & microstack framework

1. Full-stack framework: sono quei framework che offrono supporto attraverso componenti implementate o integrate da strumenti di terze parti, per ciascun livello dello stack software;
2. Micro-stack framework: a differenza dei primi, non offrono supporto out-of-the-box per i livelli di modello e vista, ma offrono solo l'infrastruttura in grado di gestire le richieste HTTP per l'applicazione web. Un esempio di micro-stack framework è Flask, basato su python.

Tuttavia, entrambe le tipologie di web framework, aggiungono allo stack software un web server di sviluppo integrato da impiegare solo per lo sviluppo.

Tutti i web-framework, a prescindere dalla tipologia condividono alcuni principi, in particolare:

- Convention over configuration: principio che favorisce l'adozione di un insieme di convenzioni piuttosto che la configurazione nel dettaglio dell'applicazione. Un esempio: una classe di modello denominata ExampleModel dovrà necessariamente corrispondere un relativo ExampleModelController. Tale associazione, dunque, non è configurabile, ma è imposta dal framework.
- DRY (Don't repeat yourself): principio che sconsiglia la duplicazione dell'informazione e insiste nel mantenere ogni parte dell'informazione in un unico punto del sistema
- Accoppiamento lascio: i diversi livelli del framework non dovrebbero avere conoscenza l'uno dell'altro se non in casi particolari.

- Minor quantità di codice: non dovrebbe essere necessario scrivere codice per parti scontate e ripetitive.
- Sviluppo veloce: velocizzare lo sviluppo sul web è possibile riducendo o evitando il più possibile gli aspetti noiosi e ripetitivi.

7 WebUML

La maggior parter dei sistemi basati su web considerano utenti anonimi, che non hanno alcuna abilità o competenza informatiche. Risulta quindi essenziale un'interfaccia intuitiva e dal comportamento predicibile. Nella progettazione sono quindi molto importanti i casi d'uso.

Il linguaggio di modellazione UML è stato ideato per l'analisi e la progettazione di sistemi orientati agli oggetti e permette di rappresentare graficamente i sistemi o mediante diagrammi strutturali, basati su classi, componenti, oggetti, o mediante diagrammi comportamentali per i casi d'uso, statechart, etc.

Gli schemi architetturali per le applicazioni web usano le pagine web come elemento di base. Poiché uno degli obiettivi nella fase di design è catturare tutte le componenti del sistema, risulta cruciale catturare le pagine web come elementi principali e utilizzarle insieme alle classi e alle altre componenti del sistema.

Le pagine client possono essere considerate come ogni altra interfaccia utente, tuttavia le pagine server, ovvero le pagine web che hanno script elaborati lato server costituiscono un problema. Non essendo un approccio orientato agli oggetti, UML non riesce a modellare bene questo importante aspetto dei sistemi web. L'unica soluzione è quindi usare gli strumenti UML per costruire un'estensione che permetta di includere determinati aspetti dei sistemi web utili per descrivere le pagine server.

Le estensioni sono possibili attraverso un meccanismo che mette insieme:

- stereotipi: meccanismo di estenzione che consente di classificare i model element di UML (Sia S uno stereotipo del modello M, S conserva tutte le caratteristiche di M ma soddisfa dei vincoli addizionali e può avere nuovi tagged values)
- tagged values: rappresentano un'estenzione delle proprietà associate ad un elemento del modello. Sono costituite da una coppia che permette di aggiungere informazioni arbitrarie ad ogni elemento del modello (Tag, Value).
- vincoli che modificano la semantica: rappresentano un'estenzione della semantica del linguaggio che permettono di attribuire un ruolo ad un elemento del modello che restringe la semantica.

7.1 Server page

La pagina web è basata su script eseguiti dal server, i quali interagiscono con risorse lato server come database, logica di business e sistemi esterni.

Le operazioni della classe definiscono le funzioni dello script, mentre gli attributi rappresentano le variabili visibili all'interno della pagina.
Un vincolo importante è che la pagina può comunicare solo con oggetti presenti sul server.

7.2 Client page

Si tratta di una pagina web che può essere visualizzata direttamente da un user agent, come un browser. La pagina può contenere script interpretati dall'user agent stesso.

Le operazioni corrispondono alle funzioni degli script, mentre gli attributi corrispondono alle variabili.
La pagina può avere associazioni con altre pagine, client o server, senza alcun vincolo specifico.

7.3 Link

Questo stereotipo descrive un'associazione tra pagine client e altre pagine, che possono essere sia lato server che lato client. Anche se il collegamento tra due pagine client non coinvolge direttamente il server, viene comunque presupposta una richiesta HTTP al server per gestire il collegamento tra le pagine.

7.4 Redirect

Un'associazione importante è rappresentata dalla redirect, che indica il trasferimento del controllo da una pagina a un'altra. In questo caso, sia la pagina di origine che quella di destinazione possono essere pagine client o pagine server. Se l'origine è una pagina client, il browser richiederà automaticamente la pagina di destinazione.

8 JSP e Struts 2

Le servlet presentano un problema principale: la creazione di pagine da inviare al client richiede istruzioni complesse che riguardano il formato di presentazione della pagina stessa. Questo comporta un aumento del rischio di errori e richiede la ricompilazione della servlet per ogni minima modifica.

Per affrontare questa problematica, sono state introdotte le JSP. Esse consentono di implementare pagine in cui le istruzioni Java e il template statico coesistono in modo più chiaro, rendendoli più leggibili e facili da manutenere.

La compilazione avviene in modo automatico la prima volta che arriva una richiesta per la pagina e successivamente ogni volta che la pagina viene modificata.
È importante notare che non è necessaria la ricompilazione se le modifiche riguardano solo il template statico.

Una pagina JSP è formata da:

- un template statico
- istruzioni Java

La prima volta che una pagina JSP viene richiamata, viene automaticamente tradotta in una servlet. Successivamente, come qualsiasi altra servlet, essa viene eseguita dal servlet engine, che carica la classe utilizzando un class loader e la esegue.
Viene quindi prodota una pagina web da mandare al browser per essere visualizzata.

Si possono avere due tipi di errore:

- errori durante la traduzione
- errori durante l'elaborazione della richiesta

8.1 Elementi pagina JSP

Una pagina JSP ha più elementi:

- direttive
- dichiarazioni
- scriptlet
- espressioni
- librerie di etichette

Il resto della pagina è chiamato fixed-template data, si usano le servlet se questa parte fissa è poca o le JSP in caso contrario.

In particolare, le direttive sono dei meccanismi JSP per:

- specificare le impostazioni della pagina (ad esempio se è una pagina di errore, il tipo di linguaggio usato, etc.): **page**
- includere contenuti da altre risorse: **include**, che verranno tradotti come se fossero originariamente nelle JSP
- specificare librerie di etichette personalizzate: **taglib**

Le direttive, verranno elaborate durante la traduzione in servlet, inoltre, non producono nessun output immediato, dal momento che vengono elaborate dalla singola richiesta.

8.2 Elementi di scripting

Possono essere usati non solo per produrre pagine dinamiche, ma anche per produrre pagine statiche da restituire solo se sono soddisfatti determinati requisiti.
Sono componenti di scripting:

- scriptlets: le istruzioni contenute in uno scriptlet vengono messe in esecuzione durante l'elaborazione della richiesta http
- commenti
- espressioni: che contengono espressioni java, che il contenitore converte in un oggetto String
- dichiarazioni: per la definizione di variabil ie metodi, che diventano membri della classe ottenuta dalla traduzione della JSP
- sequenze di escape

8.3 Oggetti impliciti

Nelle JSP, gli oggetti impliciti sono oggetti predefiniti che sono disponibili senza la necessità di dichiararli o inizializzarli esplicitamente. Alcuni degli oggetti impliciti sono:

- scope di applicazione: application (il contenitore all'interno del quale viene eseguita la servlet)
- scope di pagina: config, exception, out, page, response, etc.
- scope di richiesta: request (rappresenta la richiesta HTTP corrente e fornisce metodi per accedere ai parametri della richiesta)
- scope di sessione: session (rappresenta la sessione HTTP corrente)

8.4 Svantaggi JSP

Le JSP presentano indubbi vantaggi, ma continuano a presentare problemi come:

- avere il codice Java e il template statico frammisto nello stesso file presenta problemi sia per la leggibilità che per la manutenzione
- non favoriscono il riuso del codice
- è possibile inserire tutti i metodi Java in una JSP che verrà inclusa da tutte le JSP che devono usare uno o più di quei metodi: questo modo di programmare, tuttavia, si allontana dalla programmazione orientata agli oggetti
- debugging e sviluppo più difficile a causa degli IDE pensati per analizzare codice di una classe, non di una JSP.
- etc.

Per cercare di ovviare, almeno in parte, a questi problemi, nelle versioni successive della JSP sono stati studiati dei meccanismi basati su azioni standard <jsp:action>:

- <jsp:include>, dinamico rispetto alla direttiva di include
- <jsp:forward>, inoltra la richiesta a un'altra JSP, servlet o pagina statica facendo terminare l'esecuzione della JSP corrente
- <jsp:plugin>, permette di aggiungere componenti plugin sottoforma di elementi HTML object o embed specifici di quel browser
- <jsp:param>, specifica i parametri per include, forward e plugin

8.5 JavaBeans

In informatica le JavaBean (letteralmente, chicchi di Java) sono classi scritte in linguaggio di programmazione Java secondo una particolare convenzione. Sono utilizzate per incapsulare più oggetti in un oggetto singolo (il bean), cosicché tali oggetti possano essere passati come un singolo oggetto bean invece che come multipli oggetti individuali.

Al fine di funzionare come una classe JavaBean, una classe di un oggetto deve obbedire a certe convenzioni in merito ai nomi, alla costruzione e al comportamento dei metodi. Queste convenzioni rendono possibile avere tool che possono usare, riusare, sostituire e connettere JavaBean.

8.6 Etichette personalizzate

Anche i JavaBeans, però, presentano dei problemi. Ad esempio, i nomi dei metodi devono seguire convenzioni piuttosto rigide, dando luogo a volte a nomi lunghi, complicati e alla fin fine poco chiari. Inoltre, per passare degli argomenti ai metodi occorre comunque ricorrere agli scriptlet. Un'ulteriore proposta di soluzione più flessibile dei JavaBeans è rappresentata dalle etichette personalizzate.

In molti casi, le etichette personalizzate rappresentano un'alternativa all'uso dei JavaBeans, ma questi ultimi non possono manipolare il contenuto della pagina, e comunque il loro uso richiede una qualche conoscenza di Java.

Pur ovviando ad alcuni dei problemi che caratterizzando JSP e JavaBeans, le etichette personalizzate sono piuttosto complicate da implementare ed hanno un ciclo di vita abbastanza complesso.

8.7 Apache Struts 2

Apache Struts 2 è un framework open-source per lo sviluppo di applicazioni web basate su Java.

Struts 2 si basa sul concetto di azioni (actions) e risultati (results). Le azioni rappresentano le unità di elaborazione delle richieste dell'utente e sono responsabili di gestire l'input, la logica di business e la navigazione dell'applicazione. I risultati rappresentano le varie opzioni per la visualizzazione dei dati o l'esecuzione di azioni successive.

Questo framework semplifica notevolmente la vita di un programmatore, sia in fase di sviluppo, che di revisione e di configurazione delle proprie applicazioni.

L'utilizzo di Struts supporta vantaggi significativi in termini del progetto:

- Modularità e Riusabilità: i diversi ruoli dell'applicazione sono affidati a diversi componenti. Ció 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
- Rapidità di sviluppo: è possibile sviluppare in parallelo le varie parti dell'applicazione, logica di business e di view