



# Sviluppo di un'applicazione Java ontology-based per la configurazione di un capo d'abbigliamento

ST1414 Modellazione e Gestione della Conoscenza

Danilo Quattrini

Corso di laurea triennale in Informatica per la comunicazione digitale  
Università degli Studi di Camerino  
Anno Accademico 2023/24

# Overview del progetto

Il mondo della moda è in continua evoluzione e l'adozione di nuove tecnologie gioca un ruolo cruciale in questo cambiamento. Tra le innovazioni più interessanti e utili nel settore vi è il configuratore di vestiti. Questa relazione esamina l'importanza e l'uso del configuratore di vestiti nell'industria della moda, analizzando le sue caratteristiche principali e i benefici che offre.

Il configuratore di vestiti permette a chi lo utilizza di personalizzare i propri capi di abbigliamento in base alle loro preferenze individuali. Questa funzionalità include la selezione del tipo di vestito il tessuto il colore e la taglia che vogliamo attribuirgli.

I vantaggi che possiamo riscontrare nell'utilizzo di tale configuratore, per le aziende che sono competenti in questo ambito sono le seguenti:

- **Riduzione dei Costi:** Automazione e precisione nella produzione risparmiando tempo e spese inutili.
- **Maggiore Fedeltà del Cliente:** Offrire personalizzazione ampia e crea un legame più forte con il brand.
- **Innovazione e Competitività:** Adottare tecnologie avanzate posiziona l'azienda come leader nel settore.

I clienti che usufruiscono di tale prodotto non sono da meno, saranno coloro che ne trarranno più beneficio, questo perché avranno:

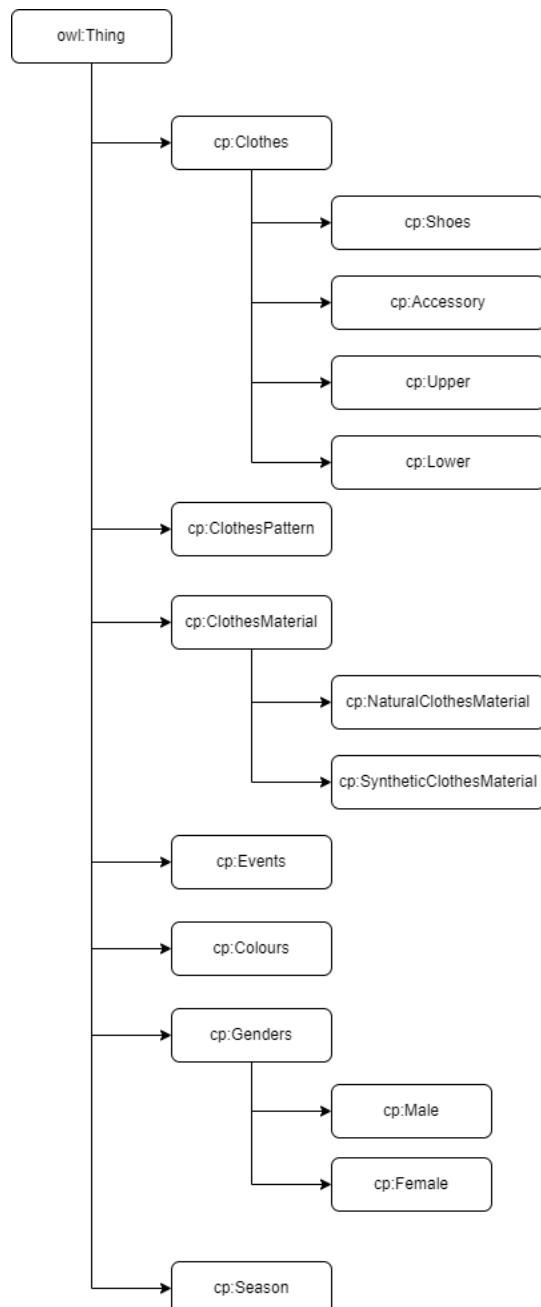
- **Esperienza di Acquisto Personalizzata:** I clienti possono ottenere capi unici e perfettamente adatti alle loro esigenze.
- **Maggiore Soddisfazione:** La capacità di visualizzare e modificare i vestiti in tempo reale aumenta la soddisfazione del cliente.
- **Contributo alla Sostenibilità:** I consumatori possono fare scelte più ecologiche grazie alla produzione su richiesta.

Per la creazione del configuratore sono andato a creare un'ontologia che è servita come base per descrivere e interrelare tutte le caratteristiche essenziali dei vestiti, permettendo una personalizzazione accurata e intuitiva per gli utenti. I domini principali considerati nell'ontologia includono colore, tipo di materiale, misure e tipi di vestiti. Sono anche descritte nel dettaglio l'origine dei materiali, il codice colore in esadecimale, l'inizio e la fine delle varie stagioni e infine viene anche descritto a che tipo di target è più idoneo quel tipo di capo o no.

Si è venuto anche a descrivere i vari eventi sociali, in cui sarebbe opportuno indossare quel determinato capo, importando un'ontologia già presente, chiamatasi

*SocietalEvent* presente in dbpedia ed utilizzata in questo progetto per, riferirmi ad eventi di contesto sociale nella quale una persona deve indossare un particolare indumento invece che un altro. L'ontologia può essere sempre più ampliata per quanto riguarda i diversi tipi di materiale e di colori che sono presenti nel mondo, ma qui troverete quelli più di comune utilizzo, si è create anche una classe per la rappresentazione dei tipi di pattern che potrebbe assumere un dato indumento.

Qui sarà mostrata ora un view breve di cosa è presente nel file `TheClothesProject.rdf` con le varie classi che sono state rappresentate per il suo funzionamento.



# Ontologia e inferenza

L'ontologia e l'inferenza sono elementi chiave per garantire il buon funzionamento e l'efficacia di un configuratore di vestiti. L'ontologia fornisce una struttura organizzata delle informazioni, mentre l'inferenza permette di derivare nuove conoscenze e migliorare la personalizzazione. Questa sezione descrive come utilizzare correttamente l'ontologia e l'inferenza per ottimizzare il configuratore di vestiti. Cos'è però l'inferenza in sé per sé?

L'inferenza sarebbe il processo di fare deduzioni e trarre conclusioni logiche basate su conoscenze o informazioni già esistenti, utilizzando regole, assiomi e relazioni prestabiliti per ricavare nuove informazioni che potrebbero non essere esplicitamente dichiarate o rappresentate nei dati forniti. Rappresentando la conoscenza del dominio in modo formale, le ontologie consentono ai motori di ragionamento automatizzati (reasoner) di trarre conclusioni, fare deduzioni e rispondere a domande complesse sulla base delle informazioni disponibili.

## Esclusività

Per quanto riguarda l'esclusività, ogni classe nell'ontologia è progettata per avere un suo tipo specifico di funzionamento, evitando così che sottoclassi e individui vengano mescolati tra loro in modo diretto. Questo approccio permette a ciascuna classe di mantenere le proprie caratteristiche uniche e distintive.

Ad esempio, la classe *ClothesMaterial* è dedicata a rappresentare i materiali utilizzati per i capi di abbigliamento, come Cotton, Leather, e Polyester. Questi materiali non si confondono con le dimensioni dei capi, gestite dalla classe *ClothesSize* e dalle sue sottoclassi come AccessorySize, LowerSize, ShoeSize, e UpperClothingSize. Ogni classe relativa alle dimensioni ha le proprie proprietà e individui, specifici per il tipo di capo che rappresentano, evitando sovrapposizioni con altre classi.

Similmente, la classe *Colour* rappresenta i colori dei capi di abbigliamento con le relative proprietà di colore come hasColorHex, e non si mescola con le proprietà di dimensione o di materiale. Questa segregazione permette un'organizzazione chiara e un accesso efficiente ai dati, poiché ogni classe gestisce un aspetto specifico e non si sovrappone con altre.

Inoltre, le classi *Pattern* e *Season* gestiscono rispettivamente i motivi decorativi e le stagioni, con le proprie specificità. Ogni classe interagisce con le altre attraverso relazioni ben definite ma mantiene la propria esclusività funzionale. Questo design garantisce che l'ontologia rimanga coerente e che le informazioni possano essere gestite, interrogate e aggiornate in modo preciso e ordinato.

# Per che cosa la utilizziamo?

In questo progetto si viene ad utilizzare il ragionamento logico che fa l'inferenza, sull'ontologia, in modo da far sì che si riesca a comporre nel modo corretto un determinato capo d'abbigliamento.

Analizzeremmo le proprietà che sono presenti e i valori che possiedono alcuni individui.

- Dividere per categoria i tipi di vestiti che si possono configurare (Upper, Lower, Accessory e Shoes) che ognuno di loro avrà il tipo di abbigliamento che si può configurare, che verrà scelto dall'utente, nei quali si potranno trovare alcuni vestiti che avranno delle stagioni di cui sono adatti indossarli e il target della quale fa riferimento.
- Ogni tipo di materiale ha una sua origine di provenienza da quello naturale a quello sintetico.
- Ogni colore avrà un suo codice HEX definita con l'opportuna DataProperty hasColorHEX, che l'utente durante la configurazione potrà scegliere liberamente
- La classe ClothesSize dove ci saranno delle sottoclassi che indicheranno, tramite i loro label, a quale categoria di indumento appartiene, quella specifica misura.
- Le stagioni presenti come classe, elencate tramite i suoi individui, permettono di far vedere all'utente, quale determinato vestito è adatto in una specifica stagione.

# Che strumenti sono stati utilizzati?

Per implementare la gestione dell'ontologia e l'inferenza descritta precedentemente nell'applicazione, sono state impiegate le seguenti API:

- Apache Jena: un framework open source per lo sviluppo di applicazioni che gestiscono linked data. Fornisce varie funzionalità, tra cui creazione e lettura di grafi RDF, esecuzione di query SPARQL, interazione con ontologie RDFS e OWL e motori di inferenza.
- Pellet (OpenLLET): un reasoner OWL-DL open source, compatibile con Apache Jena, che fornisce funzionalità di verifica della coerenza delle ontologie, computazione della gerarchia delle classi, spiegazione delle inferenze e risposte alle query SPARQL.

L'utilizzo di un reasoner esterno si è reso necessario dal momento che, come spiegato nella documentazione ufficiale di Apache Jena, nessuno dei reasoner OWL

forniti dalla versione corrente di Jena è completo in senso tecnico e le prestazioni (in particolare l'uso della memoria) del reasoner più completo.

## Limitazioni e Completezza dell'Ontologia

È importante sottolineare che l'ontologia sviluppata per il configuratore di vestiti non rappresenta la totalità della conoscenza nel campo della moda. Essa copre solo una parte delle informazioni e delle relazioni esistenti in questo ambito. Sebbene l'ontologia includa categorie fondamentali come colori, tipi di materiali, misure e tipi di vestiti, e sia progettata per essere esaustiva e funzionale, non è possibile catturare completamente tutte le variabili e le dinamiche in continua evoluzione del settore della moda.

## Utilizzo effettivo dell'ontologia

Le query SPARQL sono utilizzate per interrogare l'ontologia del configuratore di vestiti, permettendo di estrarre informazioni dettagliate sulle classi, le sottoclassi, gli individui e le loro proprietà.

Il prefisso che viene utilizzato sarebbe questo

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX cp: <https://www.unicam.it/cs/daniloquattrini/TheClothesProject#>
PREFIX dbo: <http://dbpedia.org/ontology/>
```

Di seguito invece viene fornita una descrizione delle principali query SPARQL utilizzate nel progetto e del loro utilizzo.

## Visualizzazione dei target

Con questa query l'utente potrà scegliere all'inizio il target cui si vuole creare quel determinato vestito selezionando le sottoclassi di cp: Target.

```
SELECT DISTINCT ?label ?value
```

WHERE?

```
{  
    ?target rdfs:subClassOf cp:Target.  
    BIND((? target) AS ?label).  
    ?target rdfs:label ?value
```

```
}
```

## Consulta lista dei vestiti

Dopo la scelta del target l'utente verrà indirizzato nella scelta di categoria di vestiti di interesse, che sono divise in Upper, Lower, Shoes e Accessory.

```
SELECT ?label ?value
```

```
WHERE?
```

```
{
```

```
?clothes rdfs:subClassOf cp:Clothes.  
BIND((?clothes) AS ?label) .  
?clothes rdfs:label ?value .
```

```
}
```

## Lista dei colori presenti

Le scelte seguenti saranno sempre le stesse riguardano sempre la visualizzazione delle varie sottoclassi presenti e di interesse, in questo caso però invece si andrà a visualizzare gli individui della classe cp:Colours

```
SELECT ?label ?value
```

```
WHERE
```

```
{
```

```
?individual rdf:type cp:Colour.  
BIND((?individual) AS ?label) .  
?individual rdfs:label ?value .}
```

```
}
```

Le altre query che sono presenti all'interno della classe `SelectDataQueries` e come detto in precedenza hanno il principale compito di stampare i dati presenti nell'ontologia a video, permetter all'utente di svolgere delle scelte ed utilizzarli nell'applicazione finale per poter configurare un capo d'abbigliamento.

# Responsabilità individuate

Nella fase di progettazione del software per il configuratore di vestiti, sono state individuate le seguenti responsabilità, strettamente legate all'uso dell'ontologia e delle tecnologie correlate:

## 1. Modellazione dell'ontologia

- Costruzione del modello RDF: Creazione del modello RDF per rappresentare le classi, le proprietà e le relazioni dell'ontologia dei vestiti.
- Caricamento dell'ontologia: Importazione dell'ontologia creata in un ambiente RDF su Protégé.
- Creazione del modello inferito: Utilizzo di reasoner per derivare conoscenze implicite dal modello RDF caricato.

## 2. Gestione delle query SPARQL

- Memorizzazione delle query SPARQL: Conservazione delle query SPARQL predefinite necessarie per vari scenari d'uso.
- Parametrizzazione delle query: Definizione dei parametri per personalizzare le query SPARQL in base alle esigenze dell'utente.

## 3. Esecuzione delle query SPARQL

- Preparazione delle query: Configurazione delle query SPARQL per l'esecuzione sul modello dell'ontologia.
- Esecuzione delle query: Esecuzione delle query sul modello RDF per estrarre dati specifici.

## 4. Interpretazione dei risultati delle Query

- Parsing dei risultati: Decodifica dei risultati delle query SPARQL in formato accessibile con JSON.
- Validazione dei dati: Verifica dell'accuratezza e della coerenza dei dati estratti.

## 5. Strutturazione ed Esposizione dei Dati

- Organizzazione dei dati: Strutturazione dei dati interpretati per una facile accessibilità.
- Esposizione delle informazioni: Creazione di API o servizi per esporre i dati strutturati alle applicazioni client.

## 6. Gestione del modello dell'ontologia

- Interrogazione del modello: Esecuzione di operazioni di lettura sul modello per recuperare informazioni specifiche.
- Verifica del modello: Esecuzione di controlli di coerenza e integrità sul modello dell'ontologia e sulla sua consistenza.

## 7. Presentazione dei Dati sull'Interfaccia

- Rendering dei dati: Visualizzazione dei dati interpretati negli elementi dell'interfaccia utente sul terminale in questo caso.
- Aggiornamento dinamico: Aggiornamento in tempo reale dell'interfaccia utente in risposta alle interazioni dell'utente.

# Come sono state implementate

## Costruzione del modello dell'Ontologia

La costruzione del modello dell'ontologia è gestita dall'interfaccia `RDFModelBuilder`, che definisce il contratto per la creazione di modelli RDF. La classe `TheMainBuilder` implementa questa interfaccia, creando e caricando modelli RDF standard. Un'estensione di questa classe, `TheInferredModel`, aggiunge funzionalità per la costruzione di modelli inferiti, particolarmente rilevanti per l'inferenza semantica necessaria in questo progetto.

## Mantenimento delle query SPARQL

Le query SPARQL, definite per vari scenari di utilizzo, sono mantenute attraverso l'interfaccia `SPARQLqueries`. Questa interfaccia è implementata dalla classe enum `SelectDataQueries`, che gestisce rispettivamente le query per la visualizzazione dei dati nell'ontologia. Questo approccio semplifica la gestione e l'accesso alle query necessarie al funzionamento del configuratore.

## Esecuzione delle query SPARQL

L'esecuzione delle query SPARQL è gestita dall'interfaccia `ExecuteQueries`, implementata dalla classe `QueryExecutor`. Questa classe utilizza le API Jena per eseguire le query SPARQL sul modello dell'ontologia, garantendo un'interazione con i dati RDF e il funzionamento del tutto.

## Parsing dei risultati delle query

L'interfaccia `QueryOntologyExecutor` definisce il contratto per l'interpretazione dei risultati delle query SPARQL. La classe `JSONParser` implementa questa interfaccia, convertendo i risultati delle query in una struttura dati JSON. Questa classe gestisce anche il parsing dei singoli nodi RDF e delle etichette delle risorse.

## Strutturazione ed Esposizione dei Dati

L'interfaccia `ParsedData` definisce le modalità di accesso e presentazione dei dati interpretati dalle query SPARQL. La classe `JSONData` implementa questa interfaccia, esponendo i dati in un formato JSON-like, facilitando l'integrazione con l'interfaccia utente e altre componenti dell'applicazione.

## Gestione del Modello dell'Ontologia

La classe `ControllerOfOntology` è responsabile della gestione del modello dell'ontologia. Questa classe mantiene il modello RDF istanziato, il costruttore del modello e l'esecutore delle query SPARQL, fungendo da intermediario tra il modello dell'ontologia e il controller dell'applicazione.

## Presentazione dei Dati sull'Interfaccia

L'interfaccia `TheConfigurator` presente dentro il package `configurator` definisce il contratto per l'output dei dati sul terminal e l'interazione con l'utente. Le implementazioni specifiche includono una lista di classi per presentare i dati in output e gestiscono rispettivamente la visualizzazione della lista di capi di abbigliamento e la scelta svoltasi dall'utente. Queste classi assicurano che i dati siano presentati in modo chiaro e intuitivo. Alla fine di ciò la classe `FinalClothes` viene a combinare le scelte fatte e visualizzate a video, in modo che l'utente sappia come ha configurato il suo capo.

# Conclusioni

Il progetto del configuratore di vestiti basato sull'ontologia sviluppata rappresenta un'innovazione significativa nel settore della moda e della gestione dell'abbigliamento. Grazie alla costruzione di un modello ontologico dettagliato e ben strutturato, è possibile gestire e configurare capi di abbigliamento in maniera precisa e personalizzata, rispondendo alle diverse esigenze degli utenti finali.

L'adozione di classi specifiche e ben definite in precedenza, ognuna con le proprie proprietà distintive, ha permesso di mantenere l'esclusività e la coerenza del modello. Questa struttura ontologica consente di rappresentare accuratamente i vari aspetti

dell'abbigliamento, dai materiali e le dimensioni ai colori e ai motivi decorativi, facilitando la configurazione e la personalizzazione dei capi.

L'integrazione di query SPARQL permette di estrarre informazioni specifiche e dettagliate dal modello ontologico, garantendo un accesso rapido e efficiente ai dati. Questa capacità di interrogazione, unita alla gestione inferita del modello, rende il configuratore estremamente flessibile e potente, in grado di adattarsi rapidamente ai cambiamenti e alle nuove esigenze del mercato.

La gestione centralizzata del modello ontologico, combinata con un'interfaccia utente intuitiva e ben progettata, assicura che gli utenti possano interagire facilmente con il sistema, ottenendo risultati accurati e rilevanti. La separazione delle responsabilità tra costruzione del modello, esecuzione delle query, interpretazione dei risultati e presentazione dei dati contribuisce a mantenere il sistema modulare e scalabile, pronto per future espansioni e miglioramenti.

In conclusione, il configuratore di vestiti basato su questa ontologia rappresenta un passo avanti significativo verso una gestione più efficiente e personalizzata dell'abbigliamento. Grazie a una progettazione accurata e a una struttura solida, il sistema è in grado di offrire un'esperienza utente ottimale, rispondendo in maniera precisa alle diverse esigenze dei clienti e del mercato della moda.