# MappingMasterDSL

## *(Traducción al español de la  Documentación Reglas de Lenguaje)*

Traducción al español de la edición de Documentation of Transformation Rule Language de MappingMaster de Martin O'Connor de fecha de edición 28 de octubre de 2020 disponible en el link *MappingMasterDSL · protegeproject/mapping-master Wiki · GitHub*.

*Fecha de consulta: 6 de agosto de 2022.*
*Traducción por : Mora-Alvarez, Zaida Antonieta y Luque-Morales Ramón Alberto. Estudiantes del Doctorado en Manufactura Avanzada. Posgrados CIATEQ, A.C. Querétaro, México.*

MappingMaster es un lenguaje específico de dominio (DSL) que define las asignaciones del contenido de la hoja de cálculo a las ontologías OWL. Este lenguaje se basa en la sintaxis OWL de Manchester, que en sí misma es un DSL para describir ontologías OWL.

Se puede encontrar una introducción a la sintaxis de Manchester aquí . Puede encontrar un conjunto de ejemplos de expresiones de sintaxis de Manchester en la sección de Referencia rápida de ese documento.

La sintaxis de Manchester admite la especificación declarativa de los axiomas OWL.

Por ejemplo, una declaración de sintaxis de Manchester de una clase con nombre OWL `Gum` que es una subclase de una clase nombrada llamada `Product` se puede escribir usando una cláusula de declaración de clase como:

```
Class: Gum SubClassOf: Product
```

MappingMaster DSL amplía la sintaxis de Manchester para admitir referencias al contenido de la hoja de cálculo en estas declaraciones. MappingMaster introduce una nueva cláusula de referencia para hacer referencia al contenido de la hoja de cálculo. En este DSL, cualquier cláusula en una expresión de sintaxis de Manchester que indique una clase con nombre OWL, una propiedad OWL, un individuo OWL, un tipo de datos o un literal puede sustituirse por esta cláusula de referencia. Cualquier declaración que contenga dichas referencias se procese previamente y se importe el contenido de la hoja de cálculo relevante especificado por estas referencias. A medida que se procesa cada, se recupera el contenido de la hoja de cálculo

correspondiente a cada referencia. Este contenido se puede utilizar de cuatro maneras principales:

• Se puede utilizar para nombrar directamente entidades OWL que se crean bajo demanda.

• Se puede utilizar para anotar entidades OWL que se crean bajo demanda.

• El contenido puede hacer referencia a entidades OWL existentes, ya sea directamente como un URI o mediante una propiedad de anotación.

• Finalmente, el contenido puede usarse como un literal.

Usando uno de estos enfoques, cada referencia dentro de una expresión se resuelve durante el preprocesamiento en una entidad OWL, tipo de datos o literal. La expresión estándar puede ser ejecutada por un procesador de sintaxis Manchester.

# Tabla de contenido (links a Github)

# Referencias

Las referencias en MappingMaster DSL están precedidas por el carácter @. Por lo general, van seguidos de una referencia de celda al estilo de Excel. En la notación de celda estándar de Excel, las celdas se extienden desde A1 en la esquina superior izquierda de una hoja dentro de una hoja de cálculo hasta columnas y filas sucesivamente más altas, con caracteres alfabéticos que se refieren a columnas y valores numéricos que se refieren a filas.

## Uso básico de referencias

Por ejemplo, una referencia a la celda A5 en una hoja de cálculo se escribe de la siguiente manera:

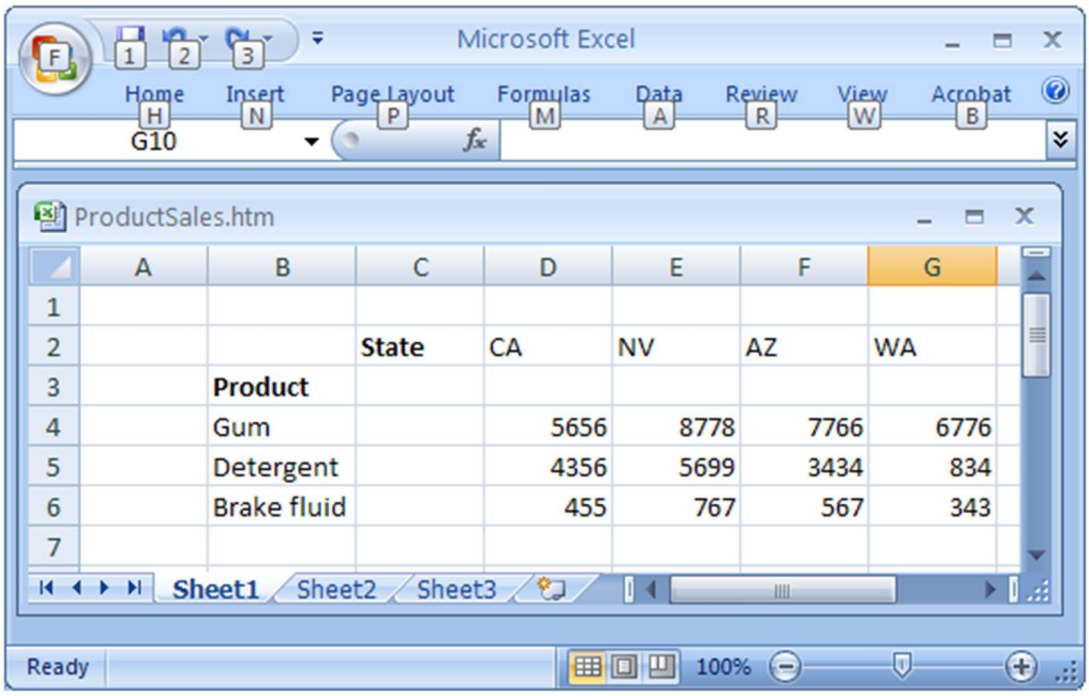
```
@A5
```

La especificación de celda anterior indica que la referencia es relativa, lo que significa que si una fórmula que contiene la referencia se copia en otra celda, los componentes de fila y columna de la referencia se actualizan adecuadamente.

Las hojas también se pueden especificar encerrando su nombre entre comillas simples y usando el "!" separador de caracteres entre el nombre de la hoja y la especificación de la celda:

```
@'A sheet'! A3
```

Por ejemplo, en la siguiente hoja de cálculo, las filas 4 a 6 de la columna B contienen categorías de productos; las columnas D a G de la fila 2 contienen identificadores de estado y el rango de cuadrícula D4 a G6 contiene cantidades de ventas.




| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | | | | | | | |
| 2 | | | State | CA | NV | AZ | WA |
| 3 | | Product | | | | | |
| 4 | | Gum | | 5656 | 8778 | 7766 | 6776 |
| 5 | | Detergent | | 4356 | 5699 | 3434 | 834 |
| 6 | | Brake fluid | | 455 | 767 | 567 | 343 |
| 7 | | | | | | | |

Estas referencias se pueden usar en el DSL de MappingMaster para definir construcciones OWL usando el contenido de la hoja de cálculo.

Por ejemplo, una expresión MappingMaster para declarar que una clase FlavouredGum es una subclase de la clase nombrada por el contenido de la celda B4 se puede escribir:

```
Class: FlavouredGum SubClassOf: @B4
```

Cuando se procese, esta expresión creará una clase con nombre OWL usando el contenido de la celda B4 ("Gum") como el nombre de la clase y declarará FlavouredGum ser su subclase. si la clase Gum ya existe, la relación de subclase simplemente se establecerá. Por tanto, las referencias se pueden usar tanto para definir nuevas entidades OWL como para referirse a entidades existentes.

Se puede escribir una expresión similar para declarar que la clase `SalesItem` es equivalente a la clase nombrada por el contenido de la celda B4:

```
Class: SalesItem

EquivalentTo: @B4
```

La sintaxis de Manchester también admite una cláusula de declaración individual para declarar individuos; los valores de propiedad se pueden asociar con los

individuos declarados utilizando una subcláusula de hechos, que contiene una lista de declaraciones de valores de propiedad.

Por ejemplo, una expresión para especificar que un individuo creado a partir del contenido de la celda D2 ("CA") tiene un valor de "California" para un valor de propiedad de datos `hasStateName` puede ser escrito:

```
Individual: @D2

Facts: hasStateName "California"
```

Aquí, un individuo `California` ser creado si es necesario y asociado con la propiedad de datos `hasStateName`, que recibirá el valor de cadena "California".
Usando la sintaxis estándar de Manchester, las propiedades de anotación también se pueden asociar con entidades declaradas.

Por ejemplo, una propiedad de anotación de tipo de datos de cadena existente llamada `hasSource` se puede utilizar para asociar a la persona de California declarada anteriormente con el documento fuente de la siguiente manera:

```
Individual: @D2

Facts: hasStateName "California"

Annotations: hasSource "DMV Spreadsheet 12/12/2010"
```

Las clases o propiedades se pueden anotar de la misma manera. Por ejemplo, una clase se puede anotar con `hasSource` propiedad de anotación de la siguiente manera:

```
Class: @D2

 Annotations: hasSource "DMV Spreadsheet 12/12/2010"
```

La sintaxis de Manchester también admite el uso de expresiones de clase OWL. En general, una expresión de clase puede ocurrir en cualquier lugar donde pueda ocurrir una clase nombrada.

Por ejemplo, una expresión para definir una condición necesaria y suficiente de una clase `Sale` usó el contenido de la celda D4 como relleno de un axioma `owl:hasValue` con la propiedad `hasAmount` puede ser escrito:

```
Class: Sale

SubClassOf: (hasAmount value @D4)
```

En general, las entidades OWL nombradas explícitamente en una expresión MappingMaster (a diferencia de las resueltas a través de una referencia) ya deben existir en la ontología de destino. En estos ejemplos, las clases Sale, SalesItem and FlavouredGum, y propiedade hasAmount, hasStateName y hasSource ya deben existir.

## Especificar el tipo de una referencia

En la expresión

```
Class: @A5

SubClassOf: Drug
```

`@A5` se refiere claramente a una clase OWL. Sin embargo, el tipo de referencia no siempre se puede inferir sin ambigüedades.
Por ejemplo, en la expresión

```
Class: Sale

SubClassOf: (@A3 value @D4)
```

la referencia `@A3` podría referirse a una propiedad de objeto, dato o anotación, y hacer referencia `@D4` podría ser un individuo OWL o un literal.

Para hacer frente a esta situación, Mapping Master admite la especificación explícita del tipo de entidad. Específicamente, una referencia puede estar opcionalmente seguida por una especificación de tipo de entidad entre paréntesis para declarar explícitamente el tipo de entidad referenciada. Esta especificación puede indicar que la entidad es una clase con nombre OWL, una propiedad de anotación o datos OWL, un individuo con nombre OWL o un tipo de datos. Las palabras clave de MappingMaster para especificar los tipos son las palabras clave estándar de la sintaxis de Manchester Class, ObjectProperty, DataProperty, AnnotationProperty e `Individual`, más cualquier nombre de tipo XSD (p. ej.,`xsd:int`).

Usando esta especificación, la declaración anterior de medicamentos, por ejemplo, se puede escribir:

```
Class: @A5(Class)

 SubClassOf: Drug
```

Una declaración de un individuo de la celda B5 con un valor de propiedad asociado de la celda C5 que es de tipo flotante se puede especificar de la siguiente manera:

```
Class: @A5(Class)

 SubClassOf: Drug
```

Si `hasSalary` la propiedad de datos ya está declarada como de tipo `xsd:float` entonces no se necesita la calificación de tipo explícita. También se puede especificar un tipo predeterminado global para los literales en el caso de que el tipo de la propiedad de datos asociada sea desconocido o no especificado, o si no se proporciona un tipo explícito en la referencia.

Las referencias a propiedades e individuos de OWL pueden calificarse de la misma manera.

# Resolución de referencia

Las referencias pueden especificar entidades OWL (es decir, clases, propiedades, individuos o tipos de datos) o literales. Cuando una referencia específica una entidad OWL, el valor de referencia puede resolverse en una entidad OWL existente o puede usarse para nombrar una entidad OWL que se crea a pedido.

## Resolución de referencia básica

Se admite una variedad de estrategias de resolución de nombres al crear o hacer referencia a entidades OWL. Las tres estrategias principales son:

- Usando `rdf:IDs` para crear o resolver entidades OWL.
- Usar anotaciones `rdfs:label` para crear o resolver entidades OWL
- Cree entidades OWL basadas en la ubicación de una celda ignorando el valor de referencia resuelto

Con `rdf:ID` codificación, y a la entidad OWL generada a partir de una referencia se le asigna su `rdf:ID` directamente del valor de referencia resuelto. Obviamente, este

contenido debe representar un identificador válido (los espacios no están permitidos en `rdf:ID`s por ejemplo).

Usando `rdfs:label` codificación, una entidad OWL resuelta a partir de una referencia recibe un URI generado automáticamente y su `rdfs:label` el valor de la anotación se establece en el valor de referencia resuelto.
Con la codificación de ubicación, una entidad OWL generada a partir de una referencia también recibe un URI generado automáticamente, pero en este caso no se utiliza el valor de referencia resuelto.

La codificación de nombres predeterminada utiliza el `rdfs:label` propiedad de anotación. El valor predeterminado también se puede cambiar globalmente.

Se proporciona una cláusula de codificación de nombres para especificar explícitamente una codificación deseada para una referencia particular. Al igual que con las especificaciones de tipo de entidad, esta cláusula está entre paréntesis después de la referencia de celda. Las palabras clave para especificar los tres tipos de codificación son`mm:location`, `rdf:ID` , y `rdfs:label`.

Usando esta cláusula, una especificación de `rdf:ID` La codificación para el ejemplo de droga anterior se puede escribir:

```
Class: @B4(rdf:ID)

SubClassOf: Drug
```

Como se mencionó, MappingMaster también admite la creación de entidades donde se ignoran los valores de las celdas. En este caso, la palabra clave `mm:location` se puede utilizar entre paréntesis después de una referencia. Por ejemplo, se puede escribir una expresión para crear un individuo para la celda D4 mientras se ignora el contenido de la celda:

```
Individual: @D4(mm:location)
```

De forma predeterminada, los nombres de las entidades OWL se resuelven o generan utilizando el namespace de la ontología actualmente activa. El lenguaje incluye cláusulas `mm:prefix` y `mm:namespace` para anular este comportamiento predeterminado.

Por ejemplo, una expresión para indicar que un individuo creó o resolvió a partir del contenido de la celda A2 (suponiendo que `rdfs:label` resolución) debe utilizar el namespace identificado por el prefijo "clinical", se puede escribir:

```
Individual: @A2(mm:prefix="clinical")
```

De manera similar, una expresión para indicar que debe usar el namespace "http://clinical.stanford.edu/Clinical.owl#" puede ser escrito:

```
Individual: @A2(mm:namespace="http://clinical.stanford.edu/Clinical.owl#")
```

La calificación explícita de espacios de nombres o prefijos en la referencia permite eliminar la ambigüedad de las etiquetas duplicadas en una ontología.

## Resolución de referencia utilizando valores de anotación

Para admitir referencias directas a valores de anotación en expresiones, DSL de MappingMaster adopta el mecanismo de sintaxis de Manchester de encerrar estas referencias entre comillas simples.

Por ejemplo, si la clase OWL `Product` tiene un valor de anotación `rdfs:label` 'A sellable product' (Un producto vendible) se puede referir de la siguiente manera:

```
Class: @B4

SubClassOf: 'A sellable product'
```

Un sellable product se resolverá mediante una anotación de valor a la clase `Product` cuando se procesa esta expresión.

## Opciones de configuración de resolución de referencia

Documente las siguientes opciones:

mm:defaultPrefix, mm:defaultNamespace, mm:defaultLanguage, mm:ResolveIfOWLEntityExists, mm:SkipIfOWLEntityExists, mm:WarningIfOWLEntityExists, mm:ErrorIfOWLEntityExists, mm:CreateIfOWLEntityDoesNotExist, mm:SkipIfOWLEntityDoesNotExist, mm:WarningIfOWLEntityDoesNotExist, mm:ErrorIfOWLEntityDoesNotExist, mm:ProcessIfEmptyLabel, mm:ErrorIfEmptyLabel, mm:WarningIfEmptyLabel, mm:SkipIfEmptyLabel

# Contenido de la celda de procesamiento

El comportamiento predeterminado es utilizar directamente el contenido de la celda a la que se hace referencia. Sin embargo, este valor predeterminado se puede anular mediante una *value specification clause* opcional.

Esta cláusula generalmente se indica con el carácter '=' inmediatamente después de la palabra clave de especificación de codificación y va seguida de una lista de *value*

*specifications* entre paréntesis y separados por comas, que se agregan entre sí. Estas especificaciones de valor pueden ser referencias de celda, valores entrecomillados, expresiones regulares que contienen grupos de captura o funciones de procesamiento de texto incorporadas.

## Procesamiento de contenido de celda básico

Por ejemplo, una expresión que extiende una referencia para especificar que la entidad creada a partir de la celda A5 debe usar `rdfs:label` la codificación del nombre y que el nombre debe ser el valor de la celda precedida por la cadena "Sale:" se puede escribir de la siguiente manera:

```
Class: @A5(rdfs:label=("Sale:", @A5))
```

Las referencias de especificación de valores no están restringidas a la propia celda referenciada y pueden indicar celdas arbitrarias. También se puede especificar más de una codificación para una referencia en particular, por lo que, por ejemplo, se pueden generar valores de anotación de etiquetas e identificadores separados para una entidad en particular utilizando los contenidos de diferentes celdas.

Por ejemplo, podemos extender el ejemplo anterior para asignar el `rdf:ID` de clases generadas a la celda B5 de la siguiente manera:

```
Class: @A5(rdf:ID=@B5 rdfs:label=("Sale:", @A5))
```

Si la lista de asignaciones incluye un solo valor, se pueden omitir los paréntesis de apertura y cierre:

```
Class: @A5(rdf:ID=@B5 rdfs:label=("Sale:", @A5))
```

El lenguaje incluye varios métodos de procesamiento de texto incorporados que se utilizan en las especificaciones de valor. En la actualidad, se admiten varios métodos. Éstos incluyen
mm:replace, mm:replaceAll, mm:replaceFirst, mm:prepend, mm:append, mm:toLowerCase, mm:toUpperCase, mm:trim, mm:reverse, and mm:printf, mm:decimalFormat. Estos métodos toman cero o más argumentos y devuelven un valor. Los argumentos proporcionados pueden ser cualquier combinación de cadenas o referencias entrecomilladas.

Se puede escribir una expresión para convertir el contenido de la celda A5 a mayúsculas antes de la asignación de etiquetas:

```
Class: @A5(mm:toUpperCase(@A5))
```

Un método también puede tener un primer argumento explícito omitido si el argumento se refiere al valor de ubicación actual. Por lo tanto, la expresión anterior también se puede escribir:

```
Class: @A5(mm:toUpperCase)
```

Las funciones de procesamiento de valores también se pueden usar fuera de las cláusulas de especificación de valores, pero solo si estas cláusulas no se usan en una referencia y solo se puede usar una sola función.

## decimalFormat y printf

`decimalFormat` y `printf` admite el formateo de contenido textual y numérico. Su comportamiento sigue las especificaciones estándar de Java para la clase decimalFormat clase y el método `String.format`.
`mm:decimalFormat` se puede utilizar de la siguiente manera:

```
Individual: Fred Facts: hasSalary @A1(mm:decimalFormat("###,###.00", @A1))
```

Cuando el valor de la celda `A1` es "23000.2" esto se representará:
Individual: Fred Facts: hasSalary "23,000.20"

Aquí hay un ejemplo de `mm:printf`:

```
Class: @A1(mm:printf("A_%s", @A1))
```

Cuando el valor de la celda `A1` es "Car" esto representará:

```
Class: A_Car
```

Cualquier parámetro se puede reemplazar con una cláusula de referencia. Estas funciones trabajarán también con asignaciones explícitas `rdf:ID` y `rdfs:label`. Tenga en cuenta que si solo se proporciona un parámetro, se supone que el segundo es la ubicación de referencia envolvente.

Asi que

```
Individual: Fred Facts: hasSalary @A1(mm:decimalFormat("###,###.00"))
```

es equivalente a:

```
Individual: Fred Facts: hasSalary @A1(mm:decimalFormat("###,###.00", @A1))
```

Y

```
Class: @A1(mm:printf("A_%s"))
```

es equivalente a:

```
Class: @A1(mm:printf("A_%s", @A1))
```

Que también es equivalente a:

```
Class: @A1(rdf:ID=mm:printf("A_%s", @A1))
```

## Funciones de anidamiento (Nesting Functions)

Tenga en cuenta que las funciones no se pueden anidar directamente dentro de otras funciones. Por ejemplo, no podemos convertir directamente un valor a mayúsculas usando la función `mm:toUpperCase` y luego use una enclosing function (función envolvente) `mm:prepend` :

```
Class: @A1(mm:prepend("_", mm:toUpperCase)) # THIS IS NOT ALLOWED
```

Sin embargo, los parámetros de las funciones pueden incluir referencias, que a su vez pueden contener funciones, por lo que podemos anidar indirectamente. Por ejemplo, la mayúscula anterior seguida de una operación antepuesta podría escribirse de la siguiente manera usando este enfoque:

```
Class: @A1(mm:prepend("_", @A1(mm:toUpperCase)))
```

La función `mm:printf` también puede ser muy útil cuando se realizan combinaciones complejas de operaciones. Por ejemplo, la mayúscula anterior seguida de una operación antepuesta podría escribirse de la siguiente manera usando `mm:printf` :

```
Class: @A1(mm:printf("_%s", @A1(mm:toUpperCase)))
```

## Sustitución de caracteres

Los `mm:replace` y `mm:replaceAll` funciones se derivan de los métodos asociados en el estándar de la case String de Java.

Por ejemplo, para eliminar todos los caracteres no alfanuméricos de una celda antes de la asignación, el `mm:replaceAll` la función se puede utilizar de la siguiente manera:

```
Individual: @A5

Facts: hasItems @B5(mm:replaceAll("[^a-zA-Z0-9]",""))
```

Del mismo modo, el`mm:reemplazar`El método se puede usar para reemplazar las comas con puntos al procesar literales:

```
Individuo: @A2

Datos: hasSalary @A3(xsd:float mm:replace(",", "."))
```

## Anteponer y agregar (Prepending and Appending)

El método `mm:prepend` se puede utilizar de la siguiente manera para simplificar el ejemplo anterior:

```
Class: @A5(rdfs:label=mm:prepend("Sale:"))
```

La expresión se puede simplificar aún más omitiendo la calificación explícita `rdfs:label` si es la predeterminada:

```
Class: @A5(mm:prepend("Sale:"))
```

El método `append` funciona de manera similar.
Por ejemplo, asumiendo por defecto la codificación `rdfs:label`, la cadena "_MM" se puede agregar a una etiqueta generada de la siguiente manera usando la función `mm:append` :

```
Individual: @A2(mm:append("_MM"))
```

## Extracción de valores usando expresiones regulares

Se puede utilizar un enfoque similar para extraer valores de forma selectiva de las celdas a las que se hace referencia. Una expresión regular capturing groups se proporciona una cláusula y se puede utilizar en cualquier posición en una cláusula de especificación de valor. Esta cláusula está contenida en una cadena entre comillas encerrada entre paréntesis cuadrados. Por ejemplo, si la celda A5 de una hoja de cálculo contiene la cadena "Pfizer:Zyvox", pero solo se utilizará el texto que sigue al carácter ':' en la codificación de la etiqueta, se podría escribir una expresión de captura adecuada como:

```
Class: @A5(rdfs:label=[":(\S+)"])
```

Tenga en cuenta que los paréntesis alrededor de las subexpresiones en una cláusula de expresión regular especifican grupos de captura e indican que se deben extraer

las cadenas coincidentes. En algunos casos, más de un grupo puede coincidir con un valor de celda, en cuyo caso las cadenas coincidentes se extraen en el orden en que coinciden y se agregan entre sí.

Los grupos de captura también se pueden utilizar para generar literales. Por ejemplo, si la celda A2 en una hoja de cálculo tiene el nombre, la inicial del segundo nombre y el apellido de una persona separados por un solo espacio, se pueden usar tres expresiones de captura para extraer selectivamente cada parte del nombre y asignarlas por separado a diferentes propiedades de la siguiente manera:

```
Individual: @A2

Types: Person

Facts: hasForename @A2(["(\S+)"]),

      hasInitial @A2(["\S+\s(\S+)"]),

      hasSurname @A2(["\S+\s\S+\s(\S+)"])
```

Un ejemplo similar para extraer por separado dos enteros separados por espacios de una celda se puede escribir como:

```
Individual: @A2

 Types: Person

 Facts: hasMin @A2(xsd:int ["(\d+)\s+"]),

      hasMax @A2(xsd:int ["\s+(\d+)"])
```

Si las propiedades `hasMan` y `hasMax` son de tipo `xsd:int` entonces la calificación explícita no es requerida aquí.

La captura de expresiones también se puede invocar a través de la función `mm:capturing` :

```
Individual: @A2

Types: Person

Facts: hasForename @A2(mm:capturing("(\S+)")
```

La sintaxis de captura de expresiones sigue la admitida por la clase Patrón de Java.

## Literales

Mapping Master actualmente admite los siguientes tipos de datos:

xsd:string, xsd:boolean, xsd:byte, xsd:short, xsd:int, xsd:long, xsd:float, xsd:double, xsd:integer, xsd:decimal, xsd:dateTime, xsd:date, xsd:time, xsd:Duration, rdf:PlainLiteral, rdf:XMLLiteral

## IRIs

Mapping Master tiene varias directivas para personalizar el proceso de creación de IRI.

| Directiva | Explicación |
|---|---|
| `mm: iri` | Utilice el valor de referencia resuelto para generar un IRI. Se arrojará un error si el valor generado no representa un IRI válido. |
| mm:camelCaseEncode | |
| mm:snakeCaseEncode | |
| mm:uuidEncode | |
| mm:hashEncode | |

## Gestión de valores perdidos (Missing Value Handling)

Para hacer frente a los valores de celda que faltan, los valores predeterminados también se pueden especificar en las referencias. Se proporciona una cláusula de valor predeterminado para asignar estos valores. Esta cláusula está indicada por las palabras clave mm:DefaultLocationValue, mm:DefaultLiteral, mm:DefaultLabel, and mm:DefaultID seguido de una asignación para una cadena. Por ejemplo, la siguiente expresión usa esta cláusula para indicar que el valor "Unknown" (desconocido) debe usarse como la etiqueta de clase creada si la celda A5 está vacía:

```
Class: @A5(rdfs:label mm:DefaultLabel="Unknown")
```

También se admiten comportamientos adicionales para tratar los valores de celda que faltan. El comportamiento predeterminado es omitir una expresión completa si contiene referencias con celdas vacías. Se proporcionan cuatro palabras clave para modificar este comportamiento. Estas palabras clave indican que:

- Se debe generar un error si falta un valor de celda y se debe detener el proceso de mapeo (mm:ErrorIfEmptyLocation)
- Las expresiones que contienen referencias con celdas vacías deben omitirse (mm:SkipIfEmptyLocation)
- Las expresiones que contienen referencias con celdas vacías deben generar una advertencia además de omitirse (mm:WarningIfEmptyLocation)
- Las expresiones que contienen tales celdas vacías deben procesarse (mm:ProcessIfEmptyLocation).

La última opción permite el procesamiento de hojas de cálculo que pueden contener una gran cantidad de valores faltantes. La opción indica que el procesador del lenguaje debería, si es posible, descartar de manera conservadora la subexpresión que contiene la referencia vacía en lugar de descartar la expresión completa.
Considere, por ejemplo, la siguiente expresión que declara un individuo de la celda A5 de una hoja de cálculo y asocia una propiedad `hasAge` con él usando el valor en la celda A6:

```
Individual: @A5

 Facts: hasAge @A6(mm:ProcessIfEmptyLocation)
```

Aquí, usando la acción de comportamiento de omisión predeterminada (default skip behavior), un valor faltante en la celda A5 hará que se omita la expresión. Sin embargo, la directiva de proceso para el valor de la propiedad `hasAge` en la celda A6 eliminará solo la subexpresión que la contiene si esa celda está vacía. Entonces, si la celda A5 contiene un valor y la celda A6 está vacía, la expresión resultante seguirá declarando un individuo.

Con un enfoque similar, también se admite un manejo de valores vacíos más detallado para especificar diferentes comportamientos de manejo de valores vacíos para valores `mm:Literal`, `rdf:ID` y `rdfs:label`.

Aquí, las directivas de la etiqueta son
mm:ErrorIfEmptyLabel, mm:SkipIfEmptyLabel, mm:WarningIfEmptyLabel, y mm:ProcessIfEmptyLabel , con palabras clave equivalentes para el identificador RDF y el manejo literal. Estos son
mm:ErrorIfEmptyID, mm:SkipIfEmptyID, mm:WarningIfEmptyID, mm:ProcessIfEmptyID and mm:ErrorIfEmptyLiteral, mm:SkipIfEmptyLiteral, mm:WarningIfEmptyLiteral, mm:ProcessIfEmptyLiteral.

## Cambio de ubicación (Location Shifting)

Se proporciona una opción adicional para tratar con valores de celdas vacías. Esta opción está dirigida al caso común en muchas hojas de cálculo donde se proporciona un valor a una celda en particular y se supone que todas las celdas vacías debajo de ella tienen el mismo valor. En este caso, cuando se procesan estas celdas vacías, su ubicación debe cambiarse a la ubicación superior que contiene un valor. Por ejemplo, la siguiente expresión usa esta palabra clave para indicar que la llamada A5 no contiene un valor para el nombre de la clase declarada, entonces el número de fila debe desplazarse hacia arriba hasta que se encuentre un valor:

```
Class: @A5(mm:ShiftUp)
```

Si no se encuentra ningún valor, se aplica el procesamiento normal de manejo de valores vacíos. Directivas similares prevén cambios hacia abajo (`mm:ShiftDown`), y para permitir el desplazamiento a la izquierda (`mm:ShiftLeft`), o a la derecha (`mm:ShiftRight`).

# Iterando sobre un rango de celdas en una referencia

Obviamente, la mayoría de las asignaciones no solo harán referencia a celdas individuales, sino que repetirán un rango de columnas o filas en una hoja de cálculo. El carácter comodín '*' se puede usar en las referencias para hacer referencia a la columna y/o fila actual en una iteración. MappingMaster proporciona una interfaz gráfica para especificar estos rangos. (Pronto serán compatibles con DSL).

Las referencias de ejemplo que utilizan esta notación comodín incluyen:

- @A3
- @A*
- @**

Por ejemplo, una expresión que itera sobre la cuadrícula D4 a G6 para crear un individuo de clase `Sale` para cada celda se puede escribir:

```
Individual: @**

Types: Sale
```

Esta expresión se puede extender para asignar valores de propiedad a estos individuos:

```
Individual: @**
```

```
Types: Sale

Facts: hasAmount @**,

    hasProduct @B*,

    hasState @*2
```

# Cobertura de sintaxis de Manchester

El DSL no es compatible con toda la sintaxis de Manchester. Las siguientes cláusulas **no se admiten actualmente**:

- Declaraciones de propiedades de objetos OWL (OWL object property declarations)
- Declaraciones de propiedades de datos OWL (OWL data property declarations)
- Declaraciones de propiedad de anotación OWL (OWL annotation property declarations))
- Declaraciones de tipos de datos OWL (OWL datatype declarations)
- Cualificación de tipo literal OWL (OWL literal type qualification)
- Clases disjuntas de OWL (OWL disjoint classes)
- Propiedades equivalentes y disjuntas de OWL (OWL equivalent and disjoint properties)
- Aserciones de propiedad negativas de OWL (OWL negative property assertions)
- OWL tiene llave (OWL has key)

# Opciones de configuración

Se puede especificar un conjunto de valores predeterminados globales para las directivas de referencia. El lenguaje tiene una serie de cláusulas para especificar estos valores predeterminados.

Los siguientes ejemplos ilustran el uso de estas cláusulas junto con los valores predeterminados actuales.

- mm:DefaultReferenceType el valor predeterminado actual es `Class`. Otros valores posibles incluyen NamedIndividual, ObjectProperty, DataProperty, AnnotationProperty, y cualquier tipo de datos XSD.

- mm:DefaultPropertyType el valor predeterminado actual es `ObjectProperty`. Otros posibles valores son `DataProperty` y `AnnotationProperty`.
- mm:DefaultPropertyValueType el valor predeterminado actual es `xsd:string` . Si esperamos un valor de propiedad (datos o anotaciones), use `xsd:string`.
- mm:DefaultDataPropertyValueType el valor predeterminado actual es `xsd:string`. Otros valores posibles incluyen cualquier tipo de datos XSD.
- mm:DefaultValueEncoding el valor predeterminado actual es `rdf:ID` . Otros valores posibles son `rdfs:label`, `mm:Literal` y `rdfs:Location`.
- mm:DefaultIRIEncoding el valor predeterminado actual es `mm:CamelCaseEncoding`. Otros valores aceptables son mm:NoEncode, mm:NoSnakeCaseEncode, mm:UUIDEncode y mm:HashEncode.
- mm:DefaultShiftSetting el valor predeterminado actual es mm:NoShift. Otros posibles valores son mm:ShiftUp, mm:ShiftDown, mm:ShiftLeft, y mm:ShiftRight.
- mm:DefaultEmptyLocationSetting el valor predeterminado actual es mm:WarningIfEmptyLocation.
- mm:DefaultEmptyLiteralSetting el valor predeterminado actual es mm:WarningIfEmptyLiteral.
- mm:DefaultEmptyRDFIDSetting el valor predeterminado actual es mm:WarningIfEmptyRDFID.
- mm:DefaultEmptyRDFSLabelSetting el valor predeterminado actual es mm:WarningIfEmptyRDFSLabel.
- mm:DefaultIfOWLEntityExistsSetting el valor predeterminado actual es mm:ResolveIfOWLEntityExists.
- mm:DefaultIfOWLEntityDoesNotExistSetting el valor predeterminado actual es mm:CreateIfOWLEntityDoesNotExist.
- mm:DefaultLocationValue el valor predeterminado actual es "".
- mm:DefaultLiteralValue el valor predeterminado actual es "".
- mm:DefaultRDFID el valor predeterminado actual es "".
- mm:DefaultRDFSLabel el valor predeterminado actual es "".
- mm:DefaultLanguage el valor predeterminado actual es "".
- mm:DefaultPrefix el valor predeterminado actual es "".
- mm:DefaultNamespace el valor predeterminado actual es "".

# Resumen

MappingMaster DSL permite crear axiomas y entidades OWL a partir del contenido de una hoja de cálculo. El uso de la sintaxis de Manchester permite que estas entidades OWL se relacionen entre sí de formas complejas.

La especificación declarativa de asignaciones de esta manera tiene varias ventajas. La escritura de estas asignaciones no requiere ninguna experiencia en programación o secuencias de comandos. Estas asignaciones se pueden compartir fácilmente

usando la Interfaz gráfica de usuario de MappingMaster (Mapping Master GUI), que puede guardar y cargar estas asignaciones. Las asignaciones también se pueden ejecutar repetidamente en diferentes hojas de cálculo con la misma estructura.