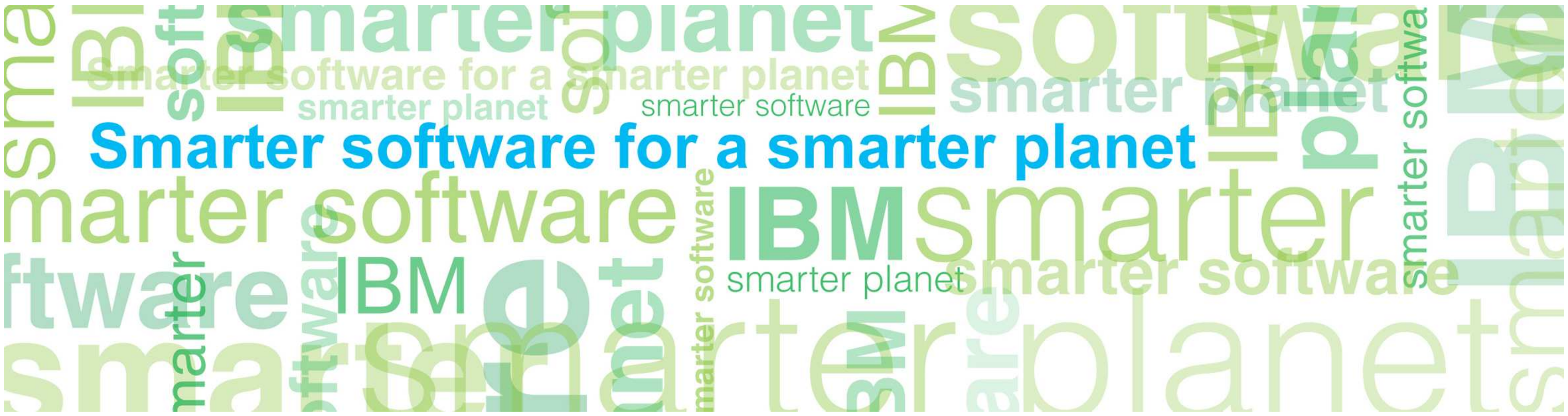
# LanguageWare Resource Workbench 7.2

# Create parsing rules



## Introduction

- **Course overview**
  - – How to create parsing rules?
  - – What types of parsing rules can be created, and what are their uses?
  - – Best practices
- **Target audience:**
  - – All audiences
- **Prerequisites:**
  - – Install LW
  - – Creating custom dictionaries and parsing rules database
  - – Create UIMA pipeline configuration
  - – Run an annotator configuration file.
- **Version Release Date:** LRW 7.2, ICA 2.2, released October, 2010

## Module objectives

After this module you will be able to:

- Understand phrase and aggregate parsing rules
- See how you can create annotations using parsing rules
- See the results of the annotations.

## Module roadmap

- **Create parsing rules.**

  How do rules work?

  What are the different parsing rules types and what are their characteristics?

  How to create rules?
- **Summary and best practices**
- **Sample exercises**

## Introduction to parsing rules

- **General**
  - Parsing rules is a powerful feature that create annotations over textual patterns.
  - Parsing rules are created and stored in a Parsing Rules Database. This database is then built into a Parsing rules file that can be used in a UIMA Pipeline to analyze text and annotate items of interest.
  - Parsing rules use the JFST (**J**ava™ **F**inite **S**tate **T**ransducer) technology, and translate it into a user friendly, intuitive and easy to modify tree-like charts that the user can modify without having to deal directly with the JFST grammar.
  - You can create parsing rules by dragging and dropping text fragments into the create rules pane, selecting match criteria, and adding annotations. All this without having to do any coding.

## Parsing rules - Different types in a nutshell

- There are three different types of rules:
  - Phrase rules: Rules that use textual patterns using custom dictionary entries and different parts-of-speech, in addition to some previously created annotations. The scope of these rules is sentence boundary.
  - Aggregate rules. The scope of these rules is not limited with sentence or paragraph boundaries. They are more powerful and have a larger cover than the Phrase rules.
  - Entities rules: Rules that create annotations over the highest level annotations created with other types of rules.

## How to create parsing rules?

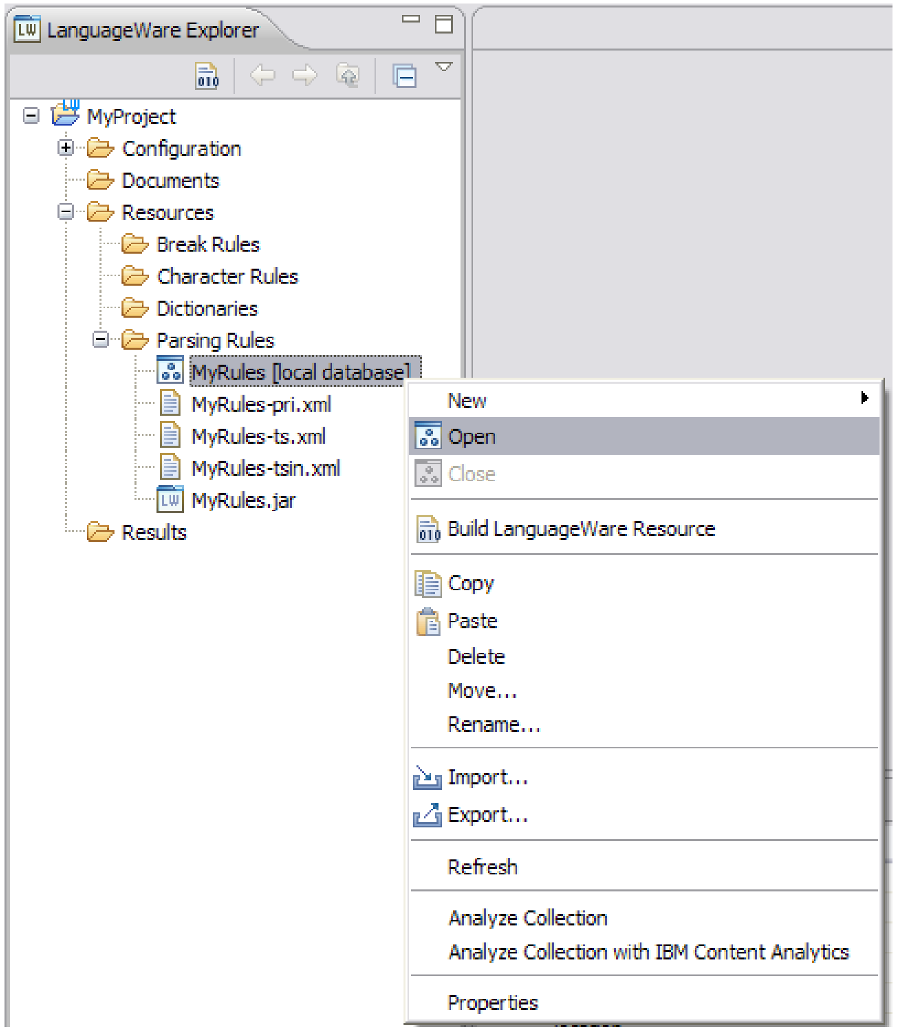
- Rules are usually written by dragging and dropping a section of text onto the Parsing rules Editor as an example. This section of text will already have been annotated by the UIMA Pipeline, and the Parsing rules Editor will use those annotations to display the pattern of annotations in the text.
- The editor is used to generalize the pattern so that it will match other similar occurrences of the same concept. Once you have identified the pattern that will be matched by the rule, you can then define one or more new annotations that will be created over all or some of the text identified by the rule.
- The rules are stored in the rules database, and when it is compiled/built, the .jar file is updated and it is re-run automatically on the text, so you can see the newly created annotations in the outline.
- The next section will guide you trough a simple example of creating a phrase rule.

## Let the modeling begin

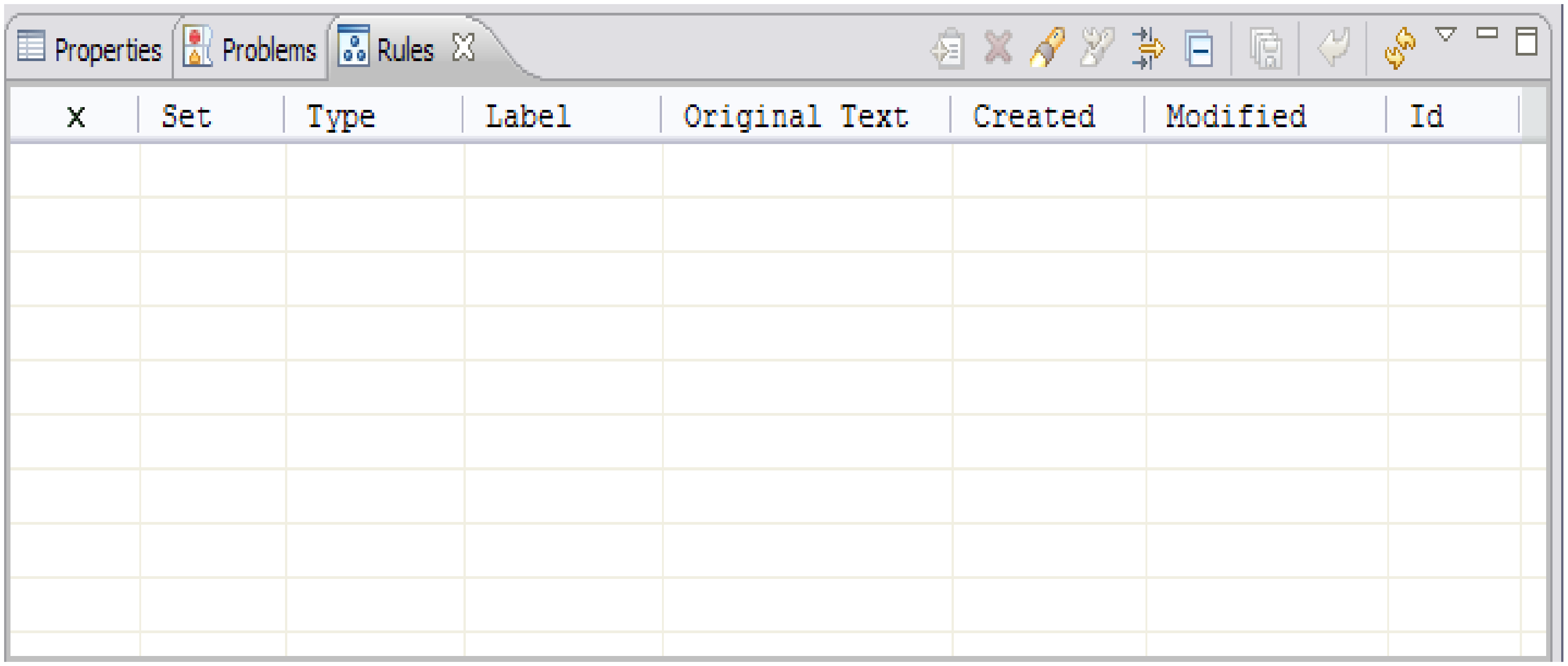
- Use a sample text containing the following two sentences with some people names:
  - Amine works for IBM Ireland.
  - John Doe, the CEO of Automatics Inc., has said in his speech that the company is looking for new investors.
- Use the UIMA pipeline configuration file developed earlier, with the following criteria:
  - Language Stage: Set it manually to English
  - Lexical stage: English Built-in dic and FirstName dictionary. The First Name dictionary contains two enties: “Amine” and “John”.
  - Parsing rules stage: MyRules.jar.

## Creating your first phrase rule

- Make sure you run the annotator on a text and check that you get the annotations in the outline.
- Open the Parsing rules database (double click the database link in the LanguageWare® Explorer, or right click it and select open). This will open two perspectives:

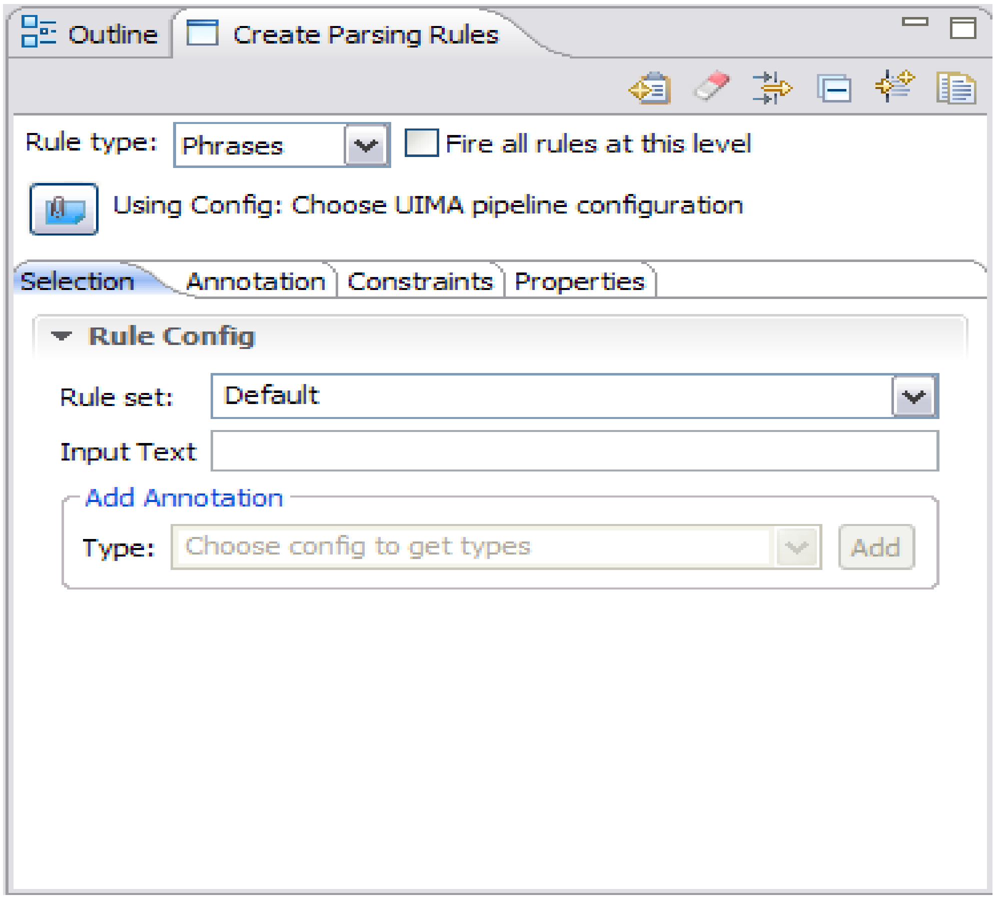


- The database viewer: this shows you the content of the database. It will be populated every time you add/modify/delete a rule.



| X | Set | Type | Label | Original Text | Created | Modified | Id |
|---|---|---|---|---|---|---|---|

- The rule creation pane: in this pane you will drag the fragment of text to create a rule, make the selection of the match criteria for the tokens, create annotations...



Rule type: Phrases

Fire all rules at this level

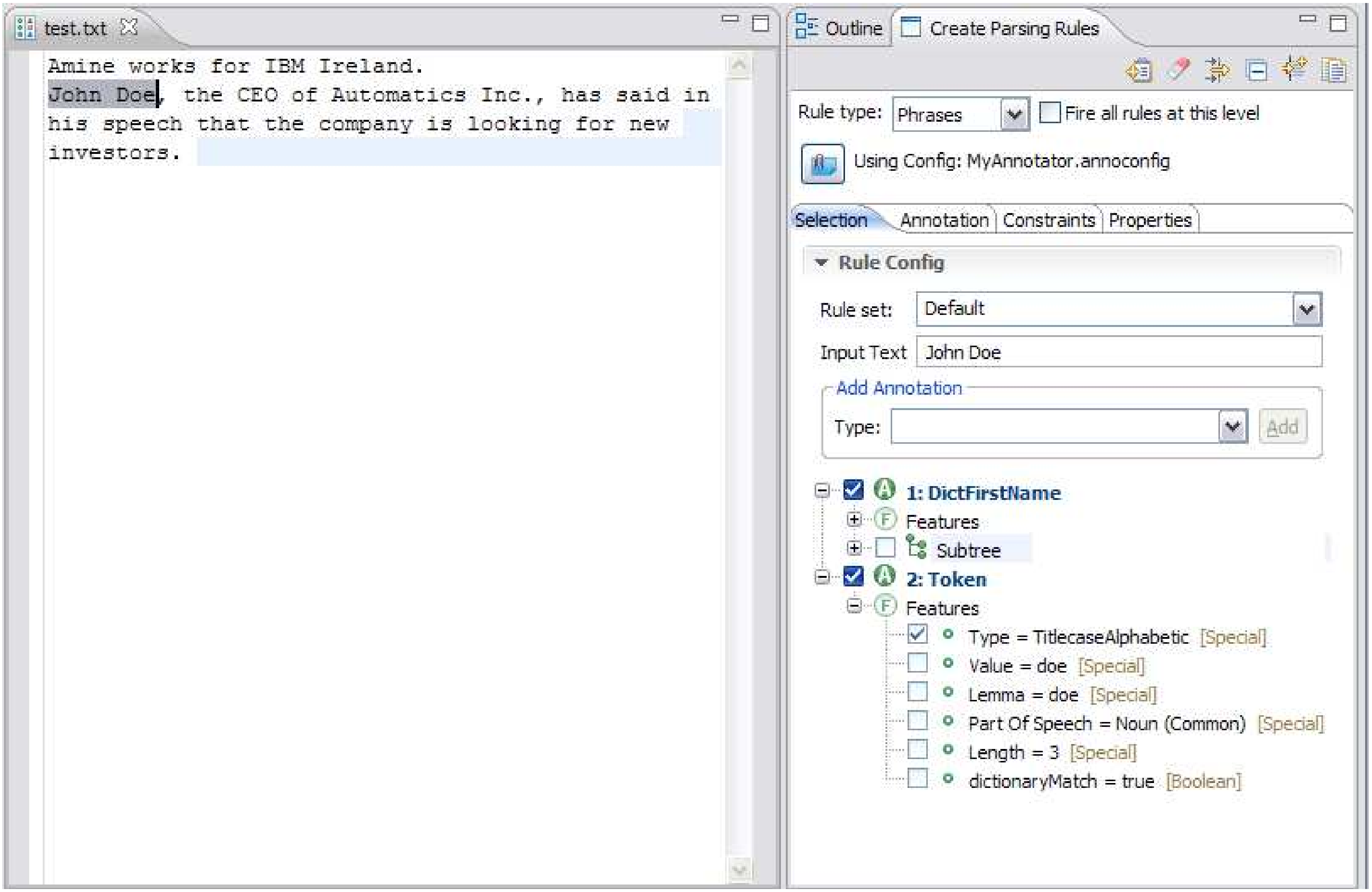
Using Config: Choose UIMA pipeline configuration

Rule set: Default

Input Text

Type: Choose config to get types

- Select the fragment of the text that will be used as a pattern to be matched. In this example, you will create a parsing rule that finds people names.
- A person Name is usually written in upper case, and the basic pattern consists of a first name followed by a family name.
- It is easier to have a dictionary of first names, this is not the case for Family names. So use the dictionary of first names that contains already two entries: “Amine” and “John”.
- So the pattern that you will use to find a person name will be a First Name followed by a word in upper case. This is represented by “John Doe” in the sample text.
- Select this fragment, drag it, and drop it into the “Create Parsing Rules” pane. Open on the right side.
- You should see an analysis tree that describes the syntactic and grammatical information related to the components of the fragment of text under analysis.



test.txt

```
Amine works for IBM Ireland.
John Doe, the CEO of Automatics Inc., has said in
his speech that the company is looking for new
investors.
```

Outline | Create Parsing Rules

Rule type: Phrases
[ ] Fire all rules at this level

Using Config: MyAnnotator.annoconfig

Selection | Annotation | Constraints | Properties

**Rule Config**

Rule set: Default

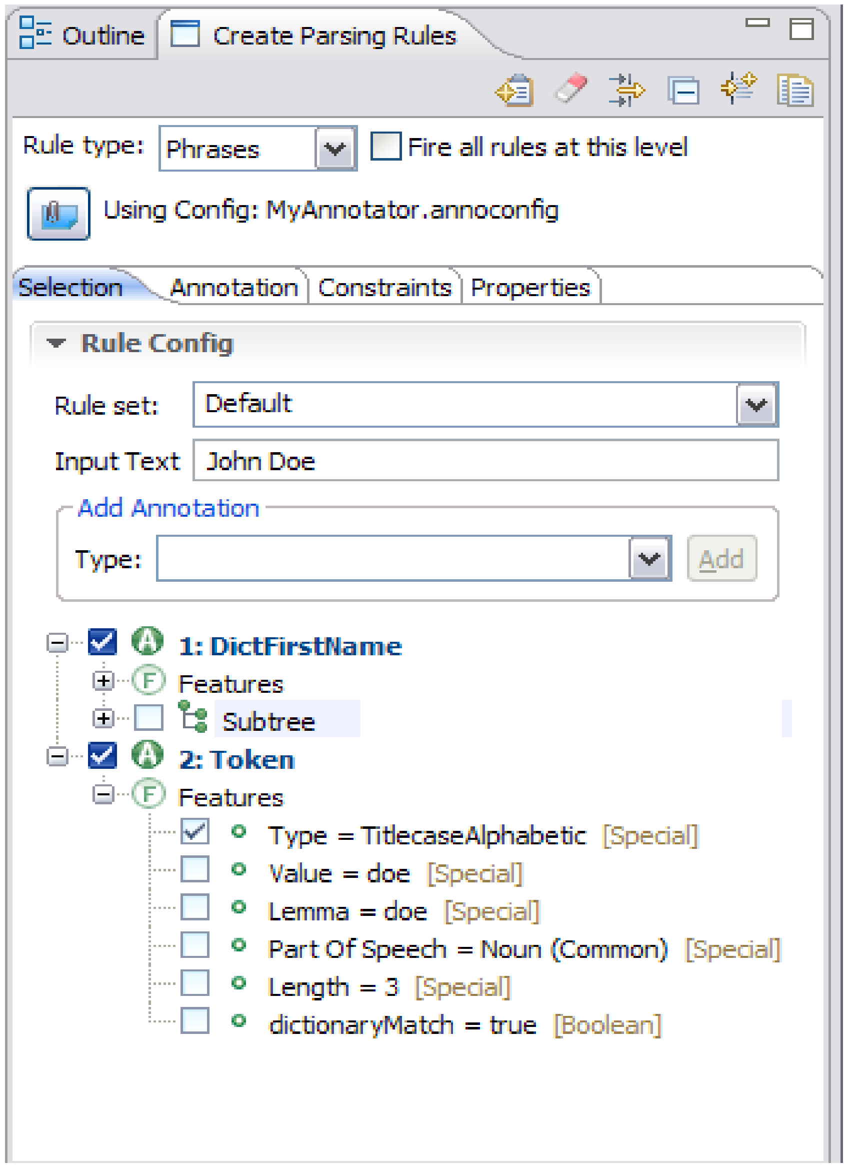
Input Text: John Doe

Add Annotation

Type: [ ] Add

- [✓] **1: DictFirstName**
  - Features
  - [ ] Subtree
- [✓] **2: Token**
  - Features
    - [✓] Type = TitlecaseAlphabetic [Special]
    - [ ] Value = doe [Special]
    - [ ] Lemma = doe [Special]
    - [ ] Part Of Speech = Noun (Common) [Special]
    - [ ] Length = 3 [Special]
    - [ ] dictionaryMatch = true [Boolean]

- The selection tab allows you to select the match criteria for the pattern in terms of syntactic and grammatical attributes of the tokens (components of the pattern).
- You can make the match criteria as general or as specific as you want. This process is made simple thanks to check boxes and tree nodes.
- In this example:
  - The first tree node is a **DictFirstName**: this is an annotation coming from the **FirstName** custom dictionary .
  - The second Node is a **Token**: this is the default annotation assigned to any entry that is not coming from a custom dictionary or previously created annotation.
- In the Token node, the tree shows the different features (criteria) that can be selected to match the pattern. According to the selection, my rule will match every First Name coming from the FirstName dictionary, followed by an upper case word.

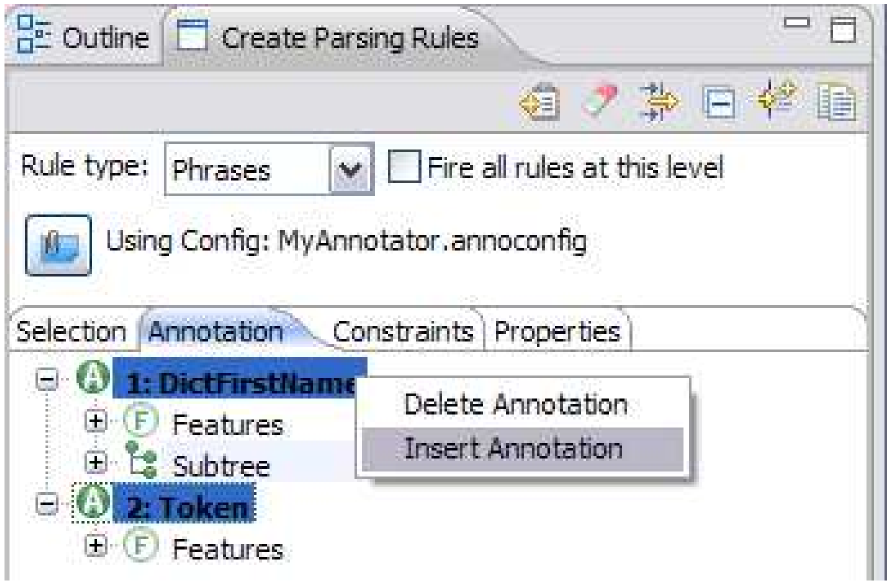


Rule type: Phrases
☐ Fire all rules at this level
Using Config: MyAnnotator.annoconfig
Rule Config
Rule set: Default
Input Text John Doe
Add Annotation
Type:

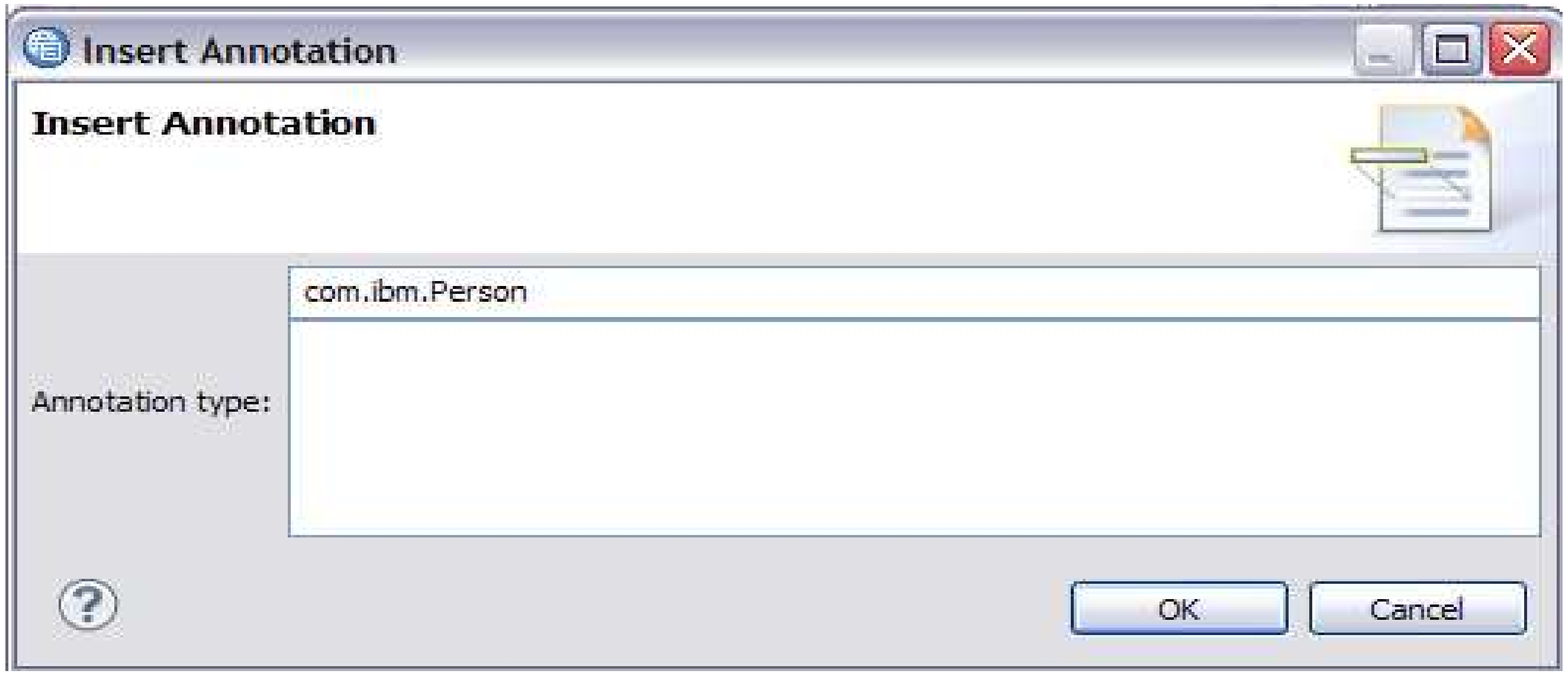
✓ 1: DictFirstName
- Features
- ☐ Subtree

✓ 2: Token
- Features
  - ✓ Type = TitlecaseAlphabetic [Special]
  - ☐ Value = doe [Special]
  - ☐ Lemma = doe [Special]
  - ☐ Part Of Speech = Noun (Common) [Special]
  - ☐ Length = 3 [Special]
  - ☐ dictionaryMatch = true [Boolean]

- After the selection is made, move to the annotation tab. This will allow you to create an annotation. An annotation can cover the whole text fragment or a part of it.
- In this example, you will create an annotation Person over this pattern.
- Select the two nodes of the tree, right click, and select **Insert Annotation.**

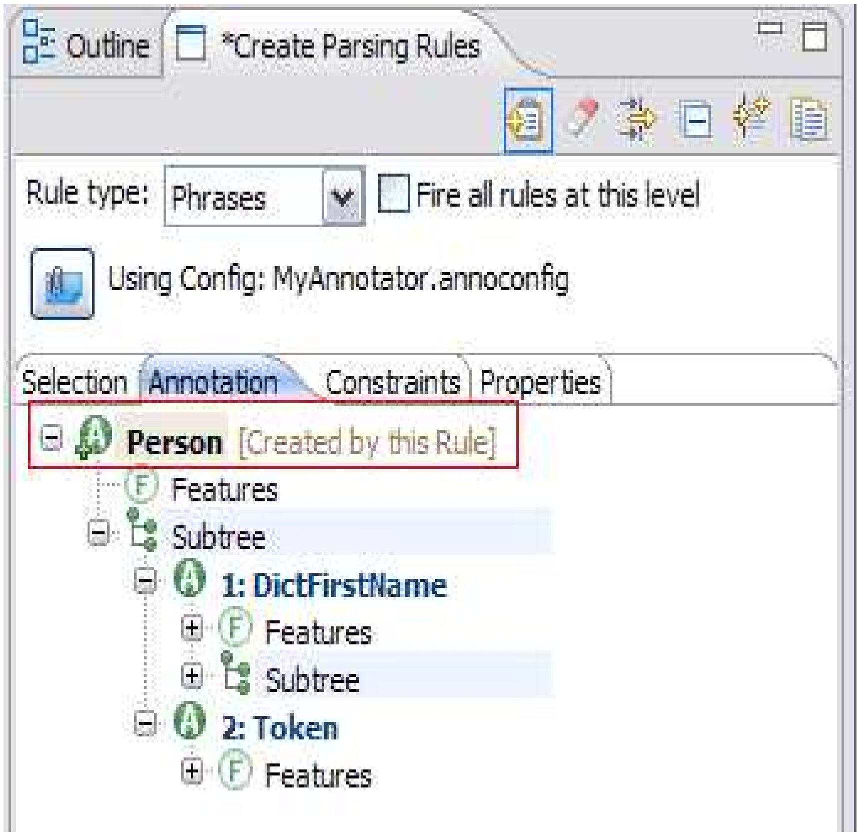


- This will prompt you to enter the name of the annotation. Make sure the Name of the annotation starts with an uppercase letter. Click OK
- NB: the UIMA prefix used (com.ibm) is the default that was specified when creating the project.



Annotation type: com.ibm.Person

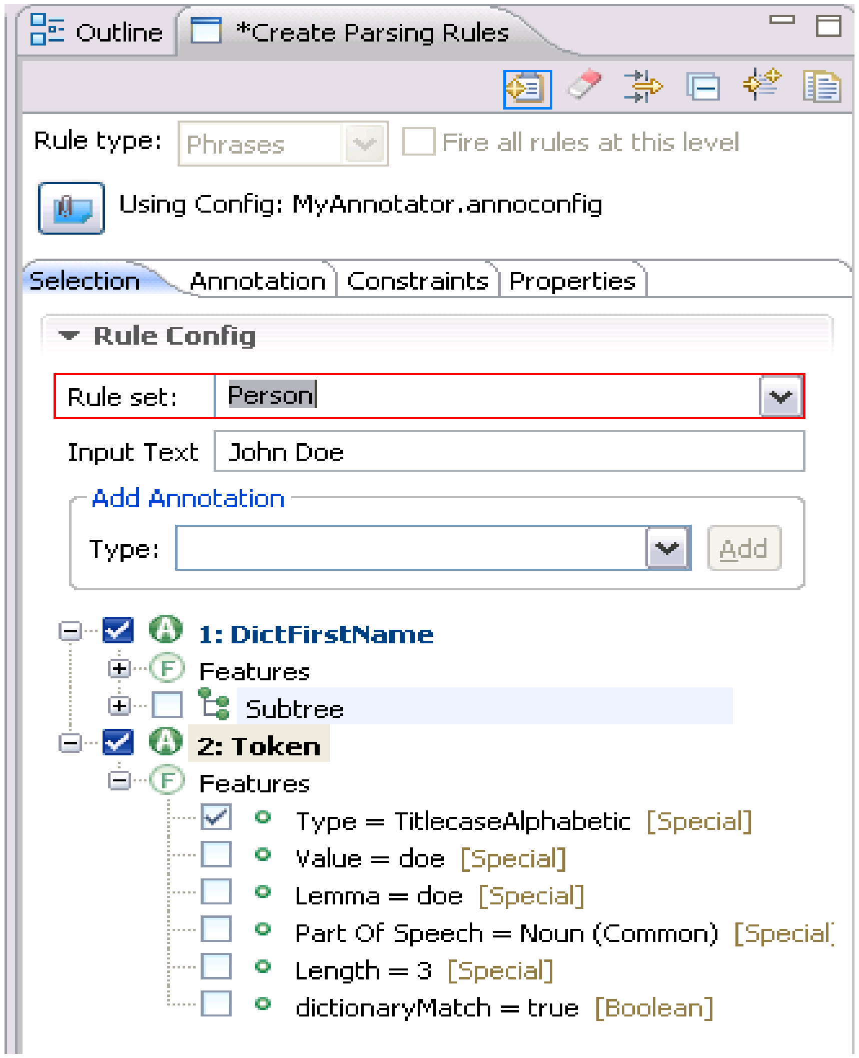
- The **Person** annotation has been added to the tree;



Rule type: Phrases
[ ] Fire all rules at this level
Using Config: MyAnnotator.annoconfig

```
⊟ Person [Created by this Rule]
     Features
  ⊟ Subtree
     ⊟ 1: DictFirstName
        ⊞ Features
        ⊞ Subtree
     ⊟ 2: Token
        ⊞ Features
```

- It is better to use a rule set name to remember which rule is doing what. This is also very important for Aggregate rules (will be covered later in this tutorial).
- Click the save rule icon (highlighted in blue).
- You can see the rule details in the database viewer on the next slide

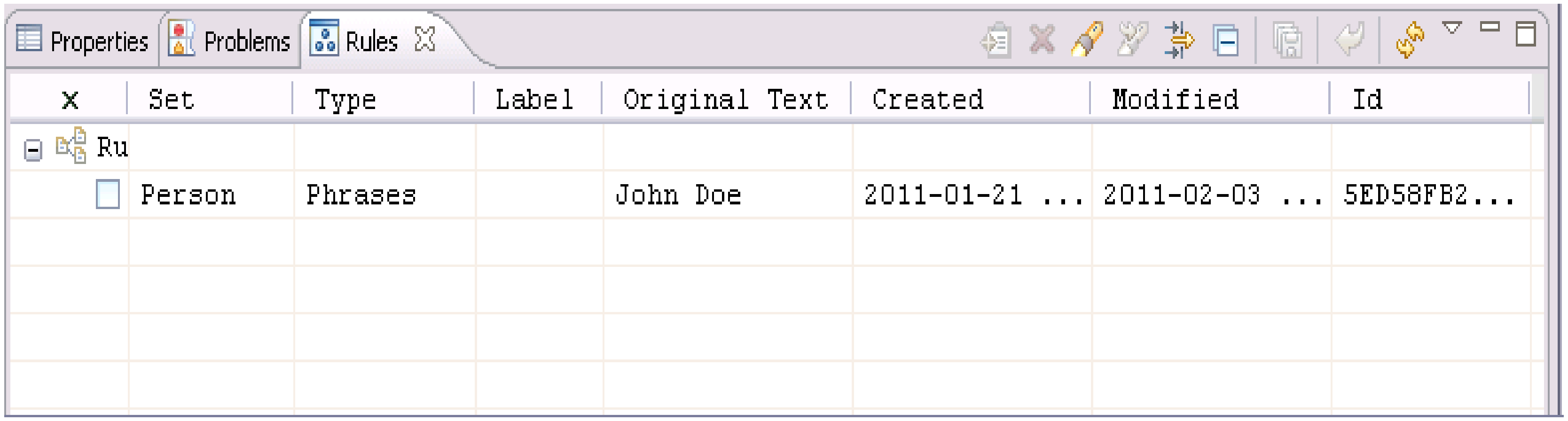


Rule type: Phrases
Fire all rules at this level
Using Config: MyAnnotator.annoconfig

Rule Config

Rule set: Person
Input Text John Doe
Type:

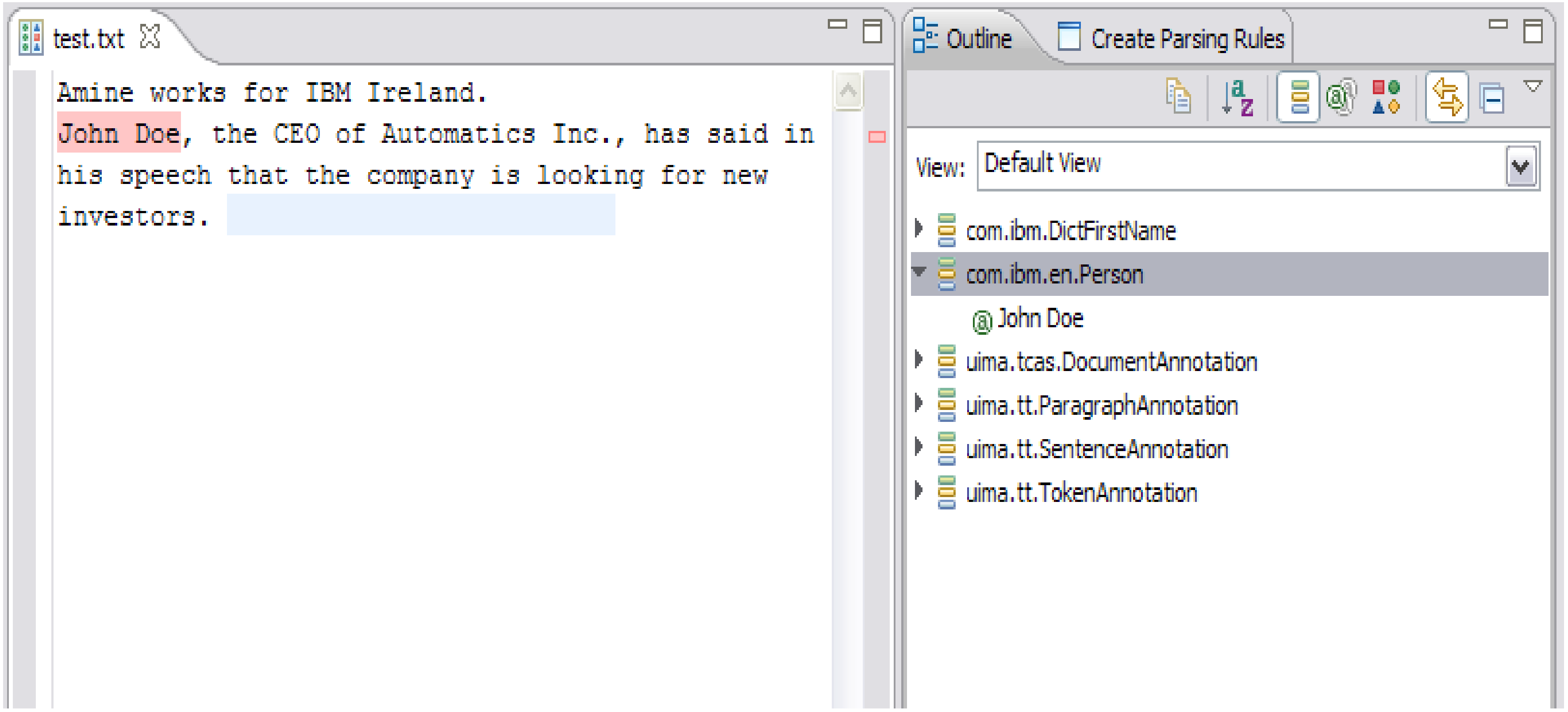
1: DictFirstName
Features
Subtree
2: Token
Features
Type = TitlecaseAlphabetic [Special]
Value = doe [Special]
Lemma = doe [Special]
Part Of Speech = Noun (Common) [Special]
Length = 3 [Special]
dictionaryMatch = true [Boolean]



Properties | Problems | Rules

| x | Set | Type | Label | Original Text | Created | Modified | Id |
|---|---|---|---|---|---|---|---|
| Ru | | | | | | | |
| ☐ | Person | Phrases | | John Doe | 2011-01-21 ... | 2011-02-03 ... | 5ED58FB2... |

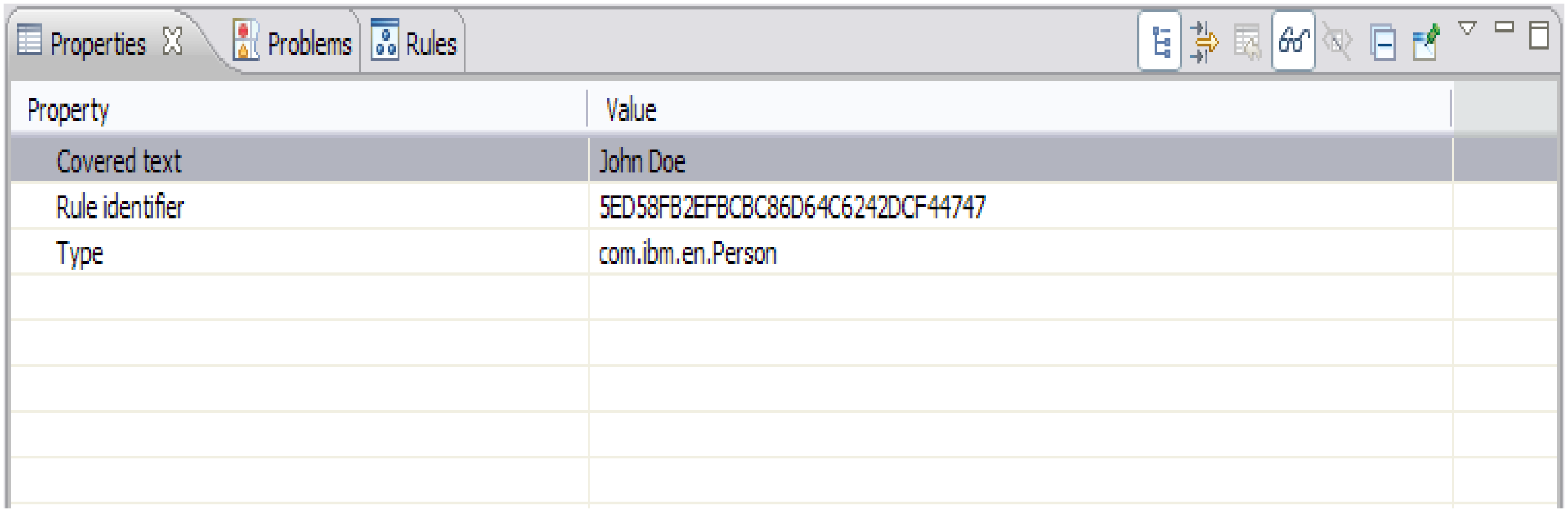
- Compile the Parsing rules Database by right clicking on it and selecting Build LanguageWare resources, or selecting the database and clicking the icon on the LanguageWare Explorer tool bar.
- The new annotation will appear in the outline (on the right side), and when you select it, the text that is matched gets highlighted in the document.



```
Amine works for IBM Ireland.
John Doe, the CEO of Automatics Inc., has said in
his speech that the company is looking for new
investors.
```

```
com.ibm.DictFirstName
com.ibm.en.Person
    John Doe
uima.tcas.DocumentAnnotation
uima.tt.ParagraphAnnotation
uima.tt.SentenceAnnotation
uima.tt.TokenAnnotation
```

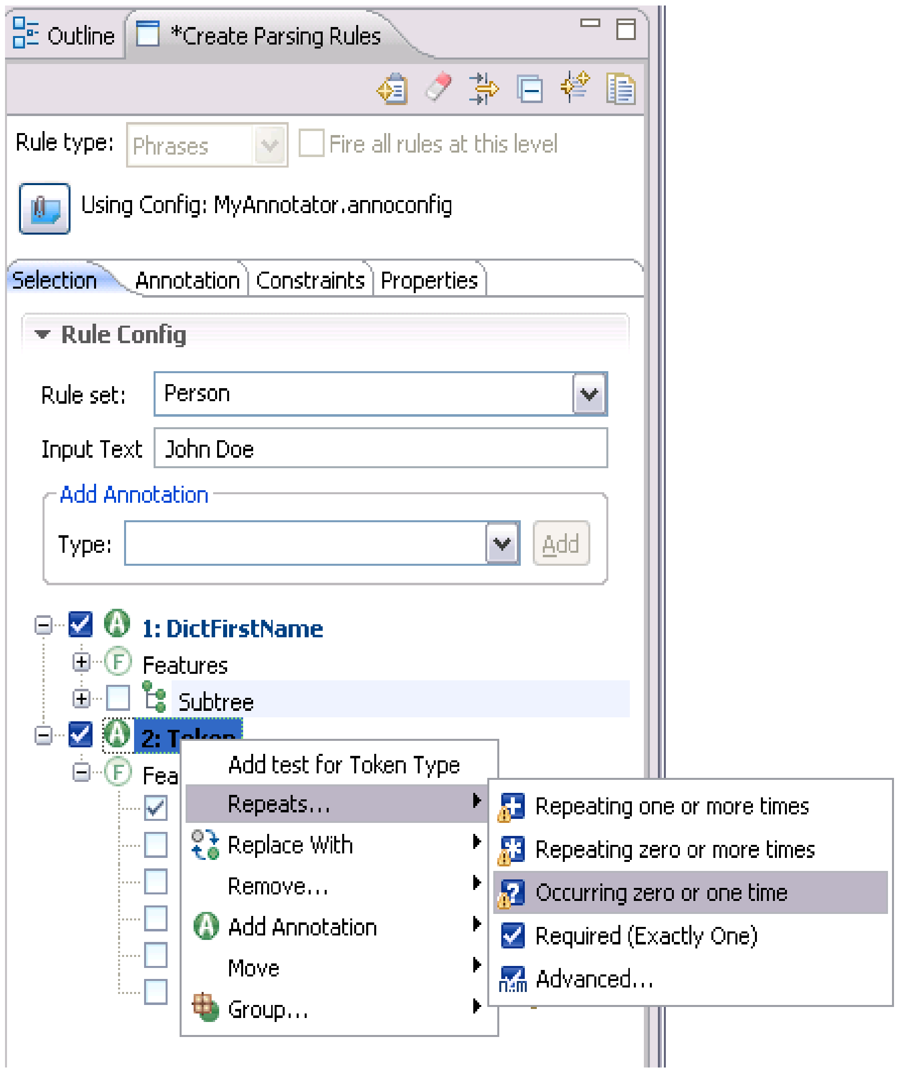
- Select the annotated text (clicking on it either on the outline or in the document), this will show the details of the highlighted text in the Properties tab.



| Property | Value |
|---|---|
| Covered text | John Doe |
| Rule identifier | 5ED58FB2EFBCBC86D64C6242DCF44747 |
| Type | com.ibm.en.Person |

## Make it more general

- The rule that you created will only match the instances of text where there is a FirstName followed by an upper-cased word. However, in some cases, you find only a first name, with no family name.
- This means that my rule should be made more general to cover both patterns. To do this, specify that the second token is optional, so if an upper case token is not found after the first name, the rule will still match the first name and highlight it as person.
- Working on the same rule, edit the selection tree, right click the token node, select Repeat/Occurring zero or one time.



Outline | *Create Parsing Rules

Rule type: Phrases | Fire all rules at this level

Using Config: MyAnnotator.annoconfig

Selection | Annotation | Constraints | Properties

Rule Config

Rule set: Person

Input Text: John Doe

Add Annotation

Type: | Add

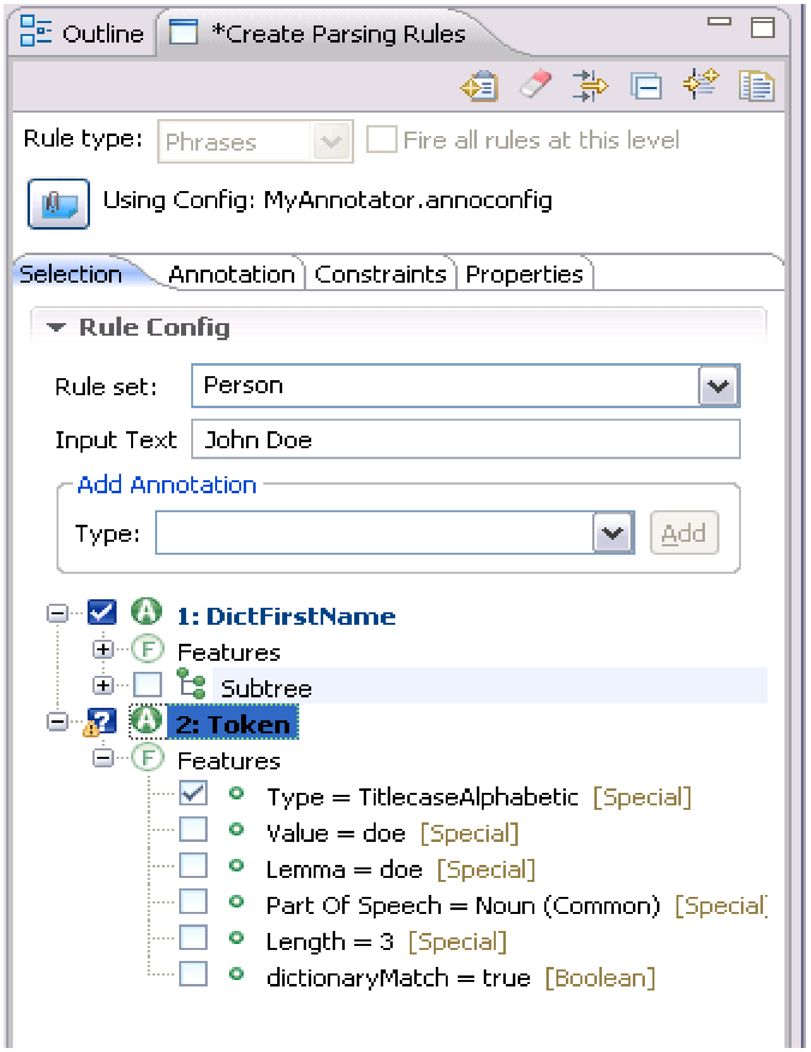
1: DictFirstName
- Features
- Subtree

2: Token
- Fea

Add test for Token Type
Repeats...
Replace With
Remove...
Add Annotation
Move
Group...

Repeating one or more times
Repeating zero or more times
Occurring zero or one time
Required (Exactly One)
Advanced...

- You can see that the icon in front of the Token node has changed. Each icon has a different repeat criteria. See the help for more information about the repeats options.
- Nothing will change in the annotation tab, because you only changed some match criteria in the selection tab.
- Save the rule and compile/build the parsing rules database.



Rule type: Phrases
Fire all rules at this level
Using Config: MyAnnotator.annoconfig

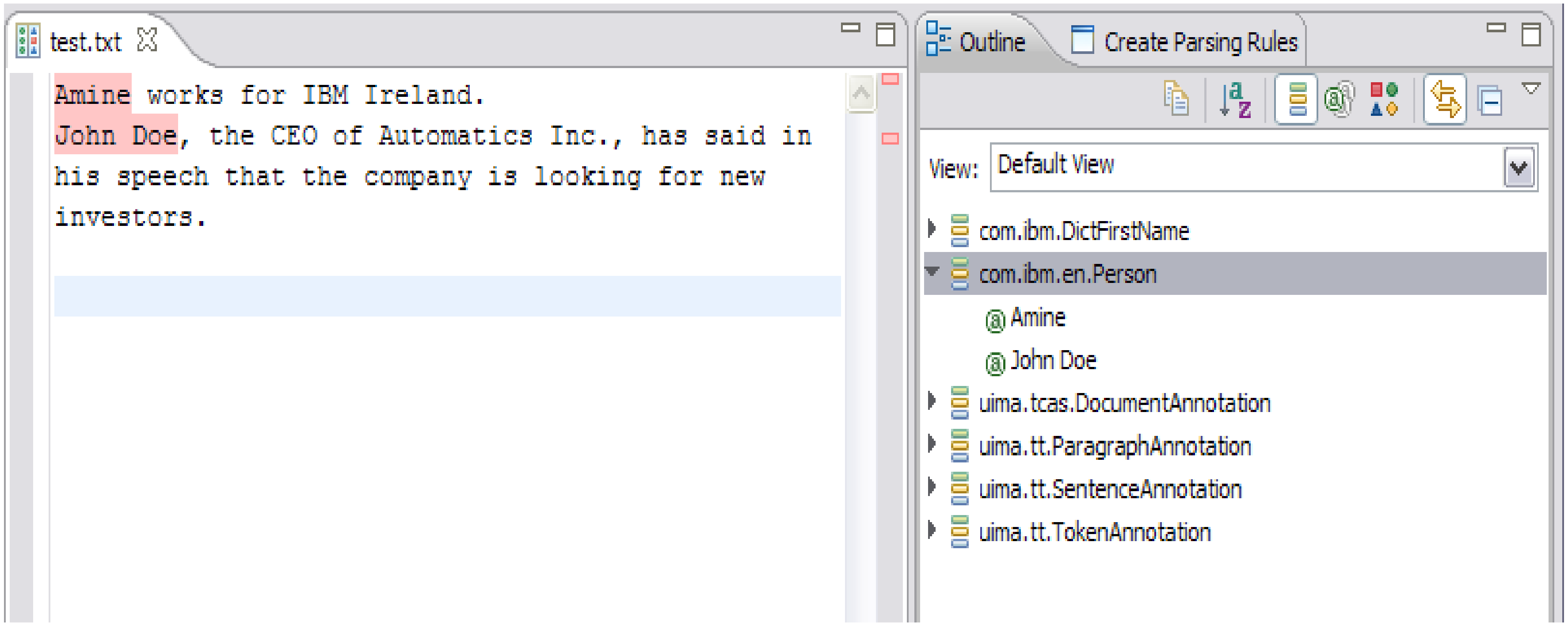
Rule Config

Rule set: Person
Input Text John Doe

Add Annotation
Type:

1: DictFirstName
Features
Subtree
2: Token
Features
Type = TitlecaseAlphabetic [Special]
Value = doe [Special]
Lemma = doe [Special]
Part Of Speech = Noun (Common) [Special]
Length = 3 [Special]
dictionaryMatch = true [Boolean]

- Switch to the Outline view, and select the Person Annotation.
- The rule now also covers the first name, because you specified that the possible family name part is optional.

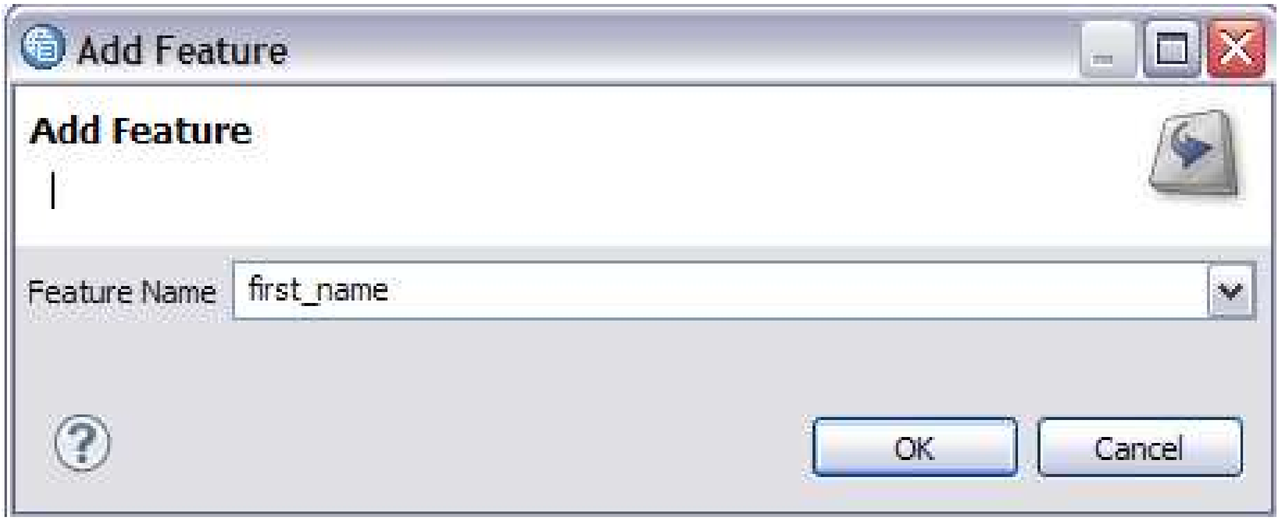


```
Amine works for IBM Ireland.
John Doe, the CEO of Automatics Inc., has said in
his speech that the company is looking for new
investors.
```

```
com.ibm.DictFirstName
com.ibm.en.Person
    Amine
    John Doe
uima.tcas.DocumentAnnotation
uima.tt.ParagraphAnnotation
uima.tt.SentenceAnnotation
uima.tt.TokenAnnotation
```

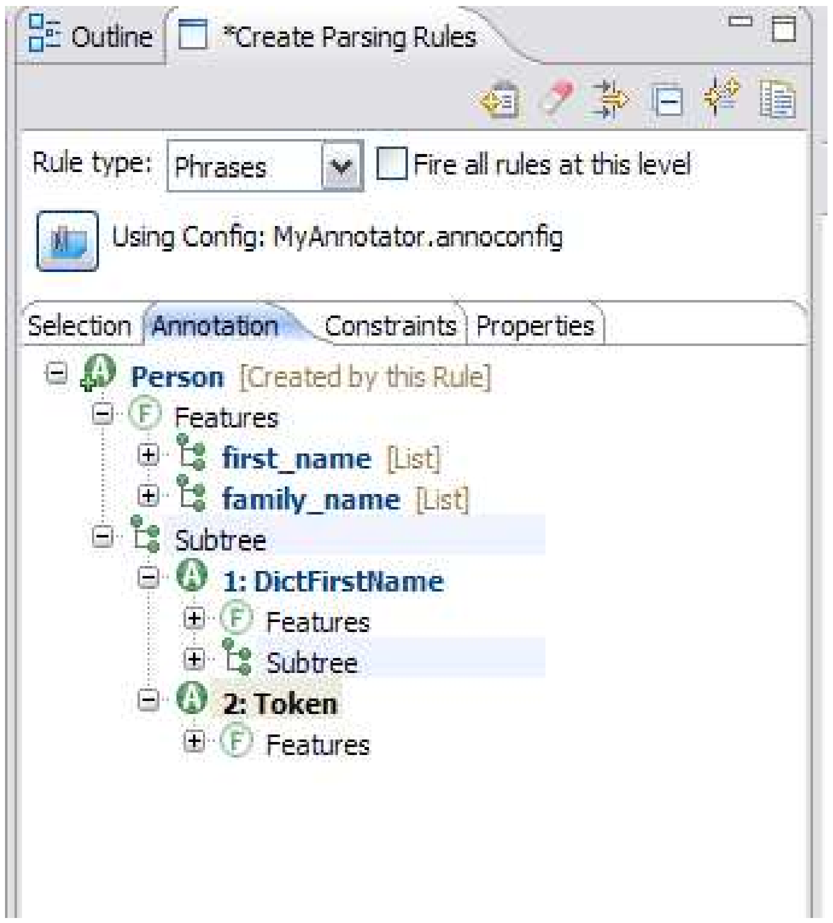
## Make it more interesting

- Now to introduce the concept of **Features** created by rules. Features add information to the components of the annotated text. It gives more granularity to the annotations. They are a type of (sub-) annotation that is created inside the main annotation.
- In this example, you will detail the Person annotation by flagging the first name and the family name.
- In the Annotation Tab, select the DictFirstName node and drag it to the feature node right underneath the Person annotation. A dialog box will pop over, and you can enter the Feature Name. Make sure it starts with a lower case letter (UIMA naming conventions). Click OK.
- Do the same thing for the token node (drag it into the Features node), and name it family_name.



Feature Name: first_name

- You should have something that looks like the screen capture.
- After adding the features, save the rule, and compile the database.
- Go to the outline, and select the Person annotation, and select the “John Doe” instance. You will see the information in the Properties tab (next slide).



Outline | *Create Parsing Rules

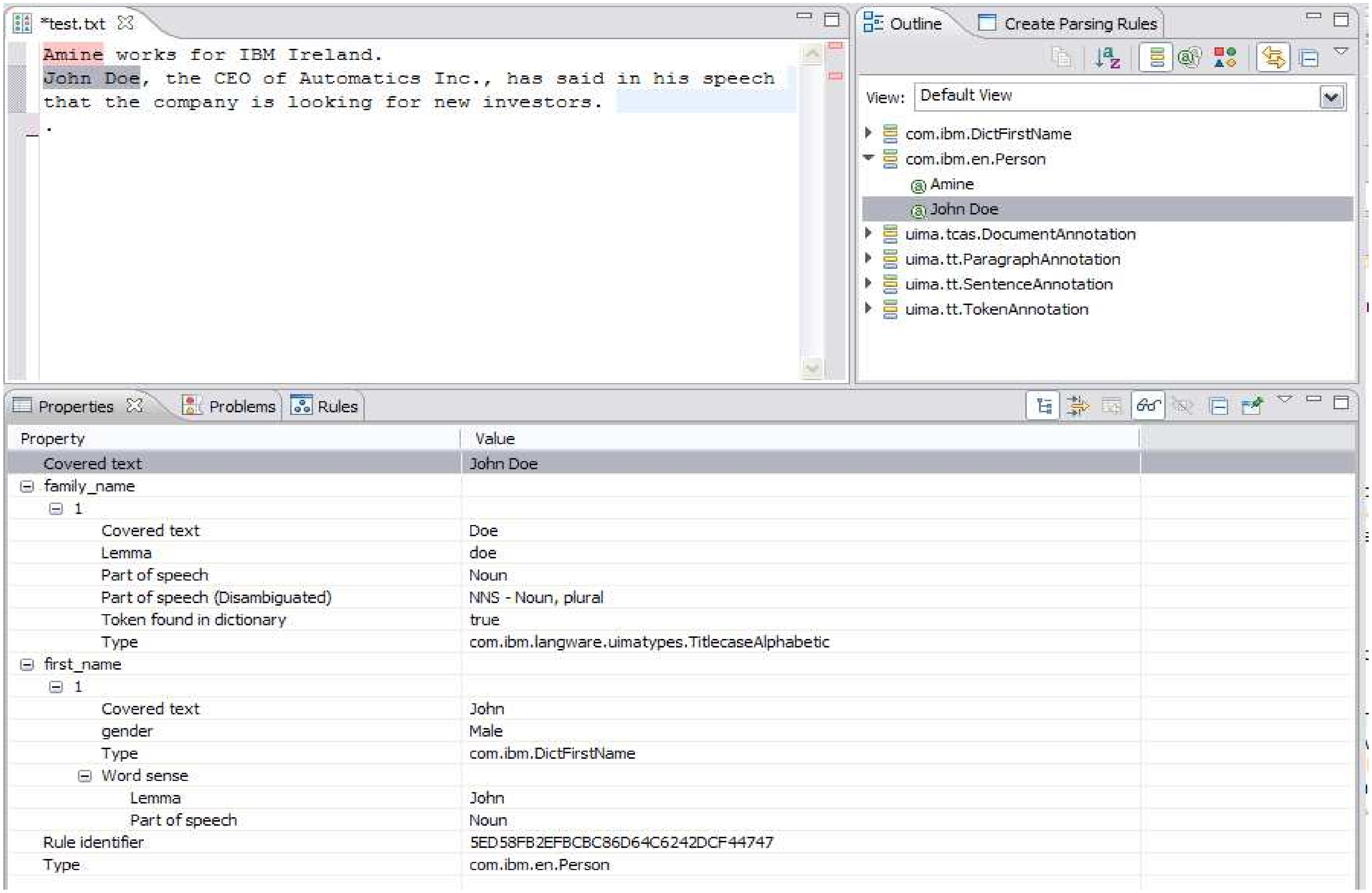
Rule type: Phrases

☐ Fire all rules at this level

Using Config: MyAnnotator.annoconfig

Selection | Annotation | Constraints | Properties

```
⊟ Person [Created by this Rule]
    ⊟ Features
        ⊞ first_name [List]
        ⊞ family_name [List]
    ⊟ Subtree
        ⊟ 1: DictFirstName
            ⊞ Features
            ⊞ Subtree
        ⊟ 2: Token
            ⊞ Features
```



```
Amine works for IBM Ireland.
John Doe, the CEO of Automatics Inc., has said in his speech
that the company is looking for new investors.
.
```

com.ibm.DictFirstName
com.ibm.en.Person
- Amine
- John Doe

uima.tcas.DocumentAnnotation
uima.tt.ParagraphAnnotation
uima.tt.SentenceAnnotation
uima.tt.TokenAnnotation

| Property | Value |
|---|---|
| Covered text | John Doe |
| family_name | |
| 1 | |
| Covered text | Doe |
| Lemma | doe |
| Part of speech | Noun |
| Part of speech (Disambiguated) | NNS - Noun, plural |
| Token found in dictionary | true |
| Type | com.ibm.langware.uimatypes.TitlecaseAlphabetic |
| first_name | |
| 1 | |
| Covered text | John |
| gender | Male |
| Type | com.ibm.DictFirstName |
| Word sense | |
| Lemma | John |
| Part of speech | Noun |
| Rule identifier | 5ED58FB2EFBCBC86D64C6242DCF44747 |
| Type | com.ibm.en.Person |

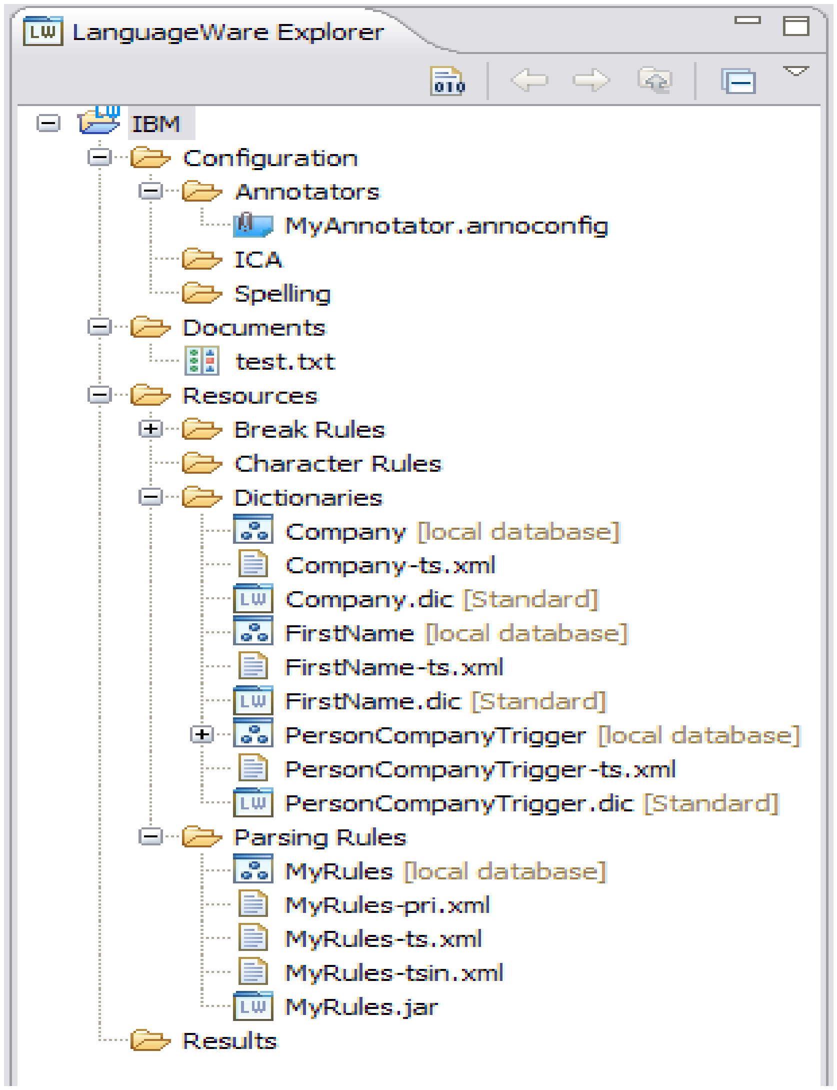
## Aggregate rules

- Aggregate rules behave differently from other rules. They skip over most annotation types when trying to match a rule.
- Phrase rules only work on a sentence scope, and it will check all tokens/annotations in the analysis. Aggregate rules are more flexible/powerful; i) you can set the scope to be the sentence, the paragraph or the whole document; ii) they only see annotations (coming from dictionaries, parsing rules..), and will skip over other tokens or punctuation.
- This will be explained through the following example.

## Creating the aggregate rule

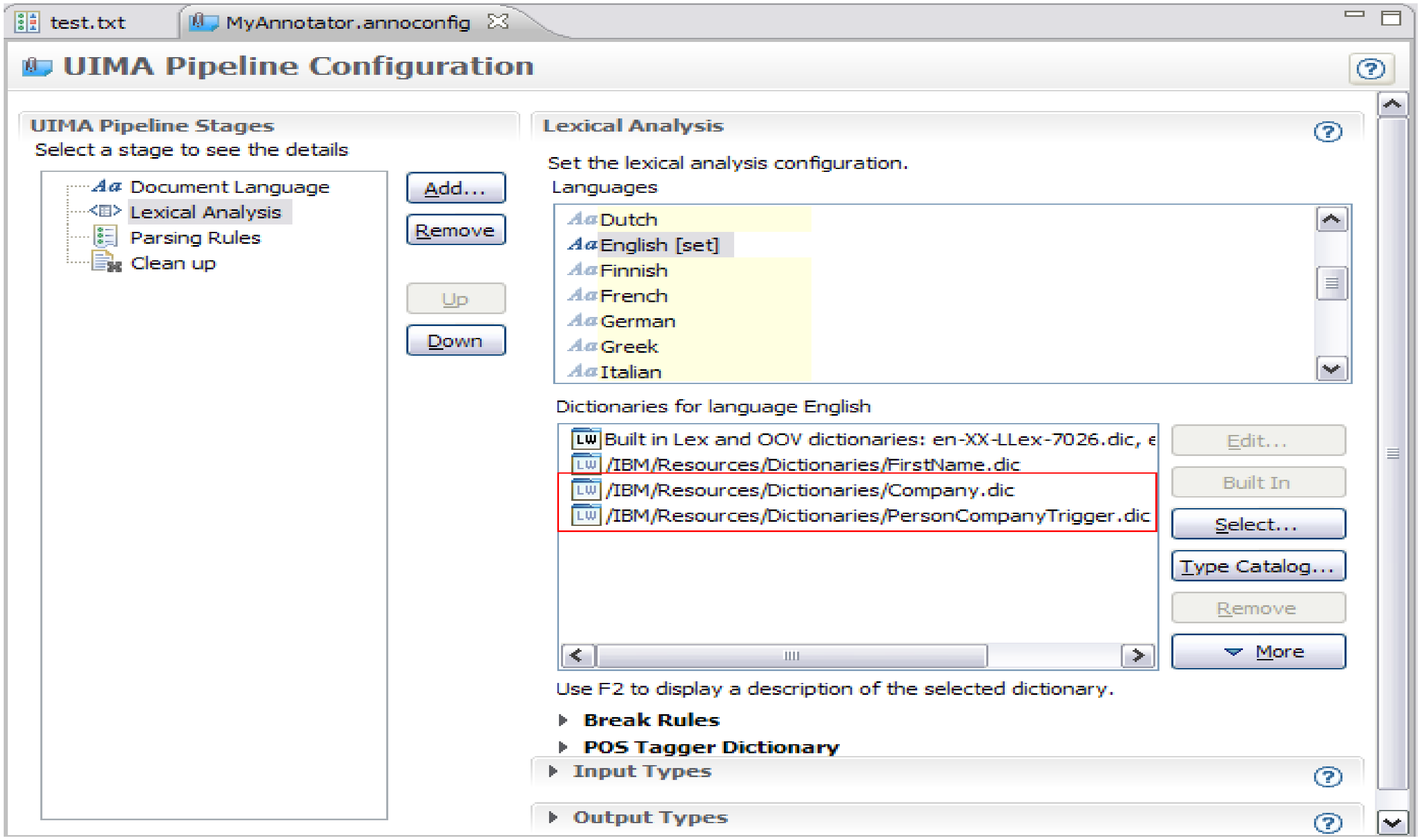
- Create a rule (pattern) that matches a relation between a person and a company.
- First figure out the best way to cover this pattern.
- Looking at the two sentences in the sample document, there is a pattern of a Person name, followed by a work relation term, and a company name. There are some intervening punctuation and tokens.
- As a good practice, always think about creating a custom dictionary when you need to match words/phrases, instead of using string matching in the rule (by checking the "**lemma**" or "**value**" criteria in the tree). The advantages of this approach can be noticed on two levels:
  - i) expandability: the words or phrase list can be expanded and developed at later stages, and more entries can be added without having to create a rule for each instance.
  - ii) performance: it's better than having regular expressions to match the strings in the rules, as regular expressions greatly affect performance.

- First, create two new custom dictionaries (See the 'create custom dictionary' section):
  - a dictionary for Company names (**Company**, with Type **DictCompany**), and add the entries "IBM" and "Automatics Inc." to it.
  - a dictionary for the Person Company relation (**PersonCompanyTrigger**, with Type **DictPersonCompanyTrigger**), and add the entries "CEO of" and "works for" to it.



```
LanguageWare Explorer
IBM
  Configuration
    Annotators
      MyAnnotator.annoconfig
    ICA
    Spelling
  Documents
    test.txt
  Resources
    Break Rules
    Character Rules
    Dictionaries
      Company [local database]
      Company-ts.xml
      Company.dic [Standard]
      FirstName [local database]
      FirstName-ts.xml
      FirstName.dic [Standard]
      PersonCompanyTrigger [local database]
      PersonCompanyTrigger-ts.xml
      PersonCompanyTrigger.dic [Standard]
    Parsing Rules
      MyRules [local database]
      MyRules-pri.xml
      MyRules-ts.xml
      MyRules-tsin.xml
      MyRules.jar
  Results
```

- Build the dictionaries and add them to the Lexical stage of the UIMA pipeline configuration, and save it.



Set the lexical analysis configuration.

Languages

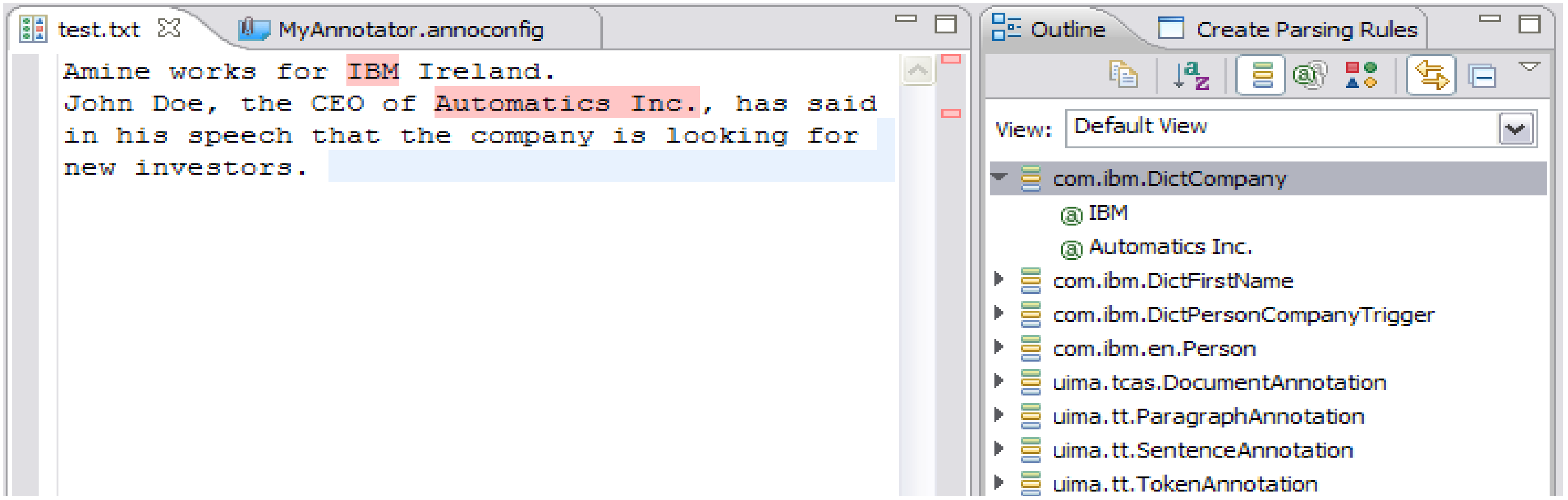
- Dutch
- English [set]
- Finnish
- French
- German
- Greek
- Italian

Dictionaries for language English

- Built in Lex and OOV dictionaries: en-XX-LLex-7026.dic, e
- /IBM/Resources/Dictionaries/FirstName.dic
- /IBM/Resources/Dictionaries/Company.dic
- /IBM/Resources/Dictionaries/PersonCompanyTrigger.dic

Use F2 to display a description of the selected dictionary.

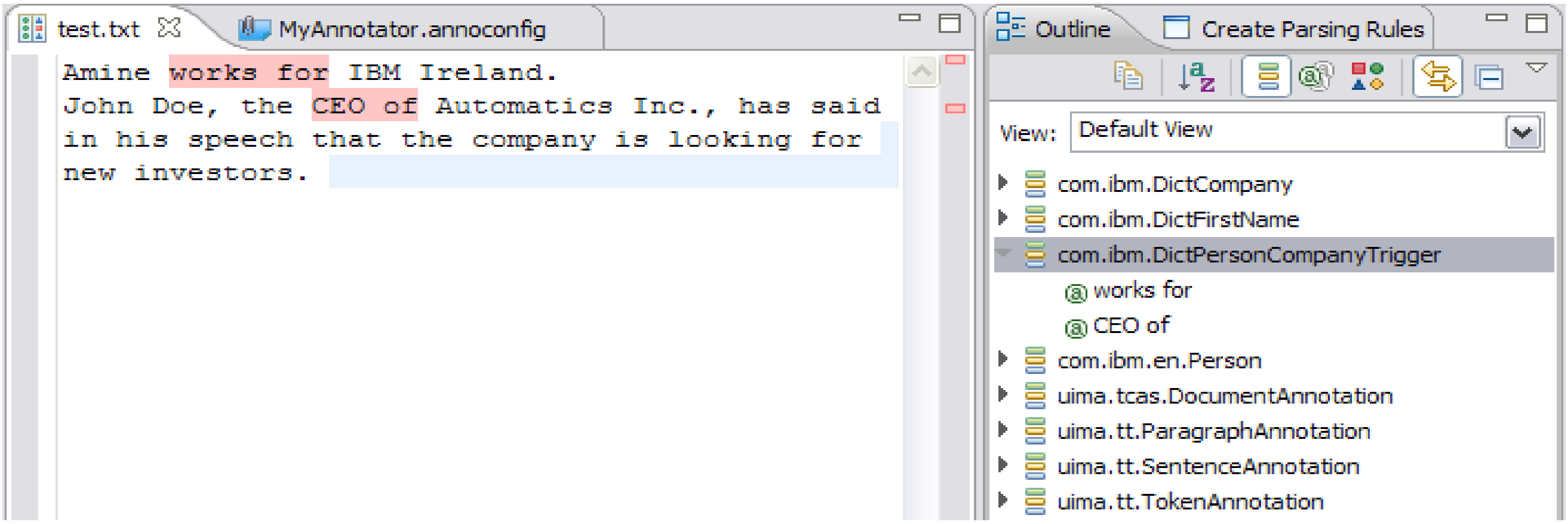
- Run the Annotator configuration on the text, and switch to the outline view, the new types should appear and you can see them highlighted in the text.



```
Amine works for IBM Ireland.
John Doe, the CEO of Automatics Inc., has said
in his speech that the company is looking for
new investors.
```

View: Default View

- com.ibm.DictCompany
  - IBM
  - Automatics Inc.
- com.ibm.DictFirstName
- com.ibm.DictPersonCompanyTrigger
- com.ibm.en.Person
- uima.tcas.DocumentAnnotation
- uima.tt.ParagraphAnnotation
- uima.tt.SentenceAnnotation
- uima.tt.TokenAnnotation

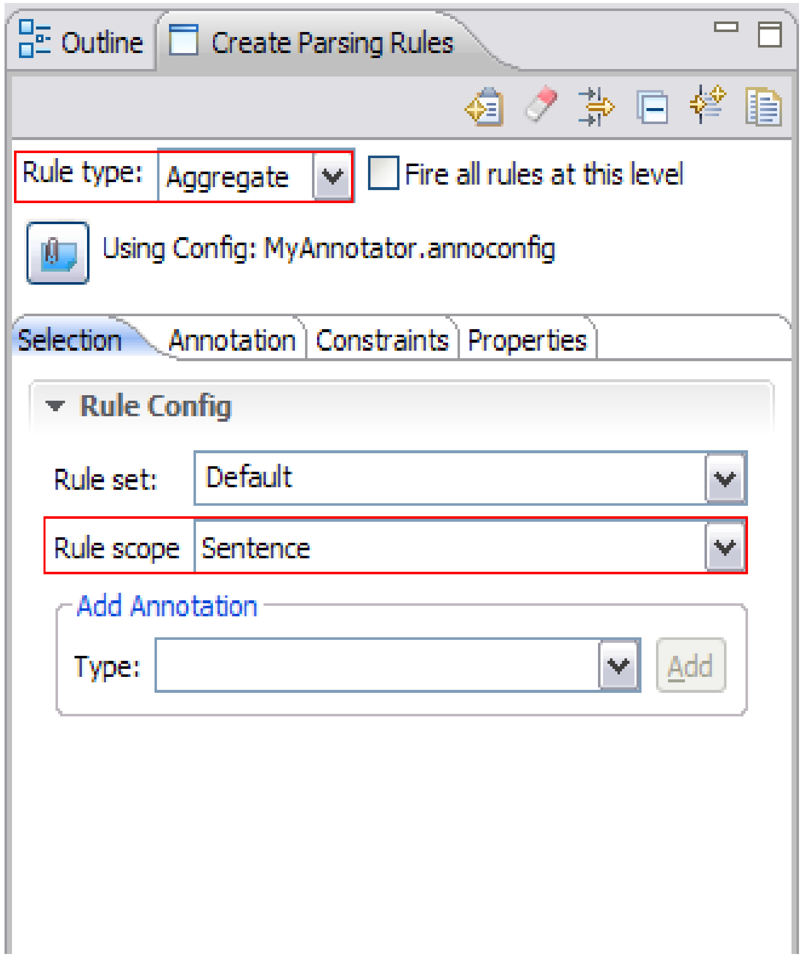


```
Amine works for IBM Ireland.
John Doe, the CEO of Automatics Inc., has said
in his speech that the company is looking for
new investors.
```

View: Default View

- com.ibm.DictCompany
- com.ibm.DictFirstName
- com.ibm.DictPersonCompanyTrigger
  - works for
  - CEO of
- com.ibm.en.Person
- uima.tcas.DocumentAnnotation
- uima.tt.ParagraphAnnotation
- uima.tt.SentenceAnnotation
- uima.tt.TokenAnnotation

- To create an aggregate rule, open the Parsing rules database, and change the rule type to Aggregate in the Create parsing rules view (as shown in the screen capture).
- If the rule editor is not empty, click the erase button (beside the save rule button)
- Note that a new field appeared replacing the “Input text” for Phrase rules: Rule scope (set by default to “sentence”).


Outline | Create Parsing Rules

Rule type: Aggregate

Fire all rules at this level

Using Config: MyAnnotator.annoconfig

Selection | Annotation | Constraints | Properties

Rule Config

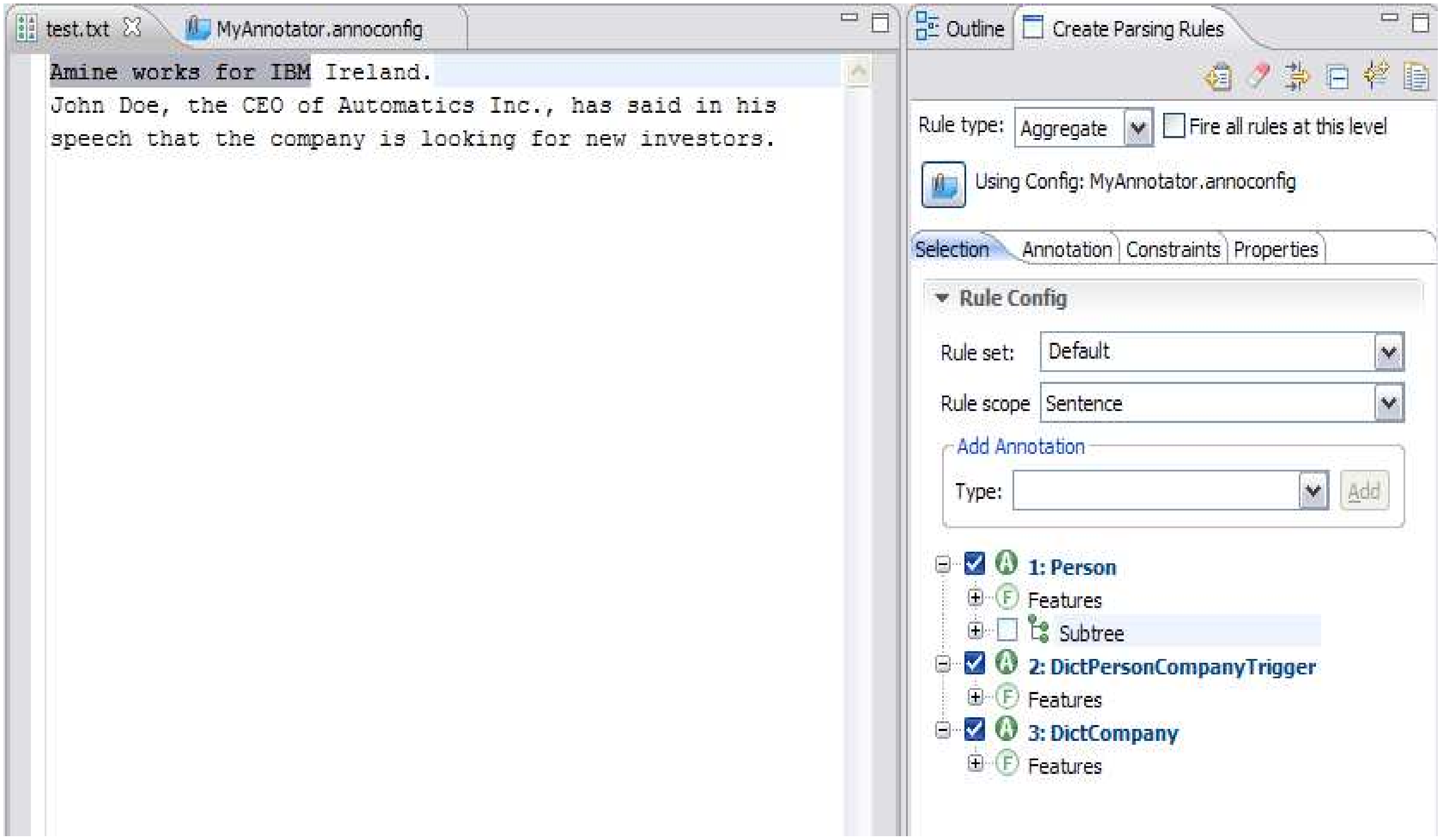
Rule set: Default

Rule scope: Sentence

Add Annotation

Type:

- The same drag-and-drop mechanism is used for aggregate rules. Select the fragment of text that contains the pattern you want to match, drag it to the "Create Parsing Rules" pane. This is the output you should see.

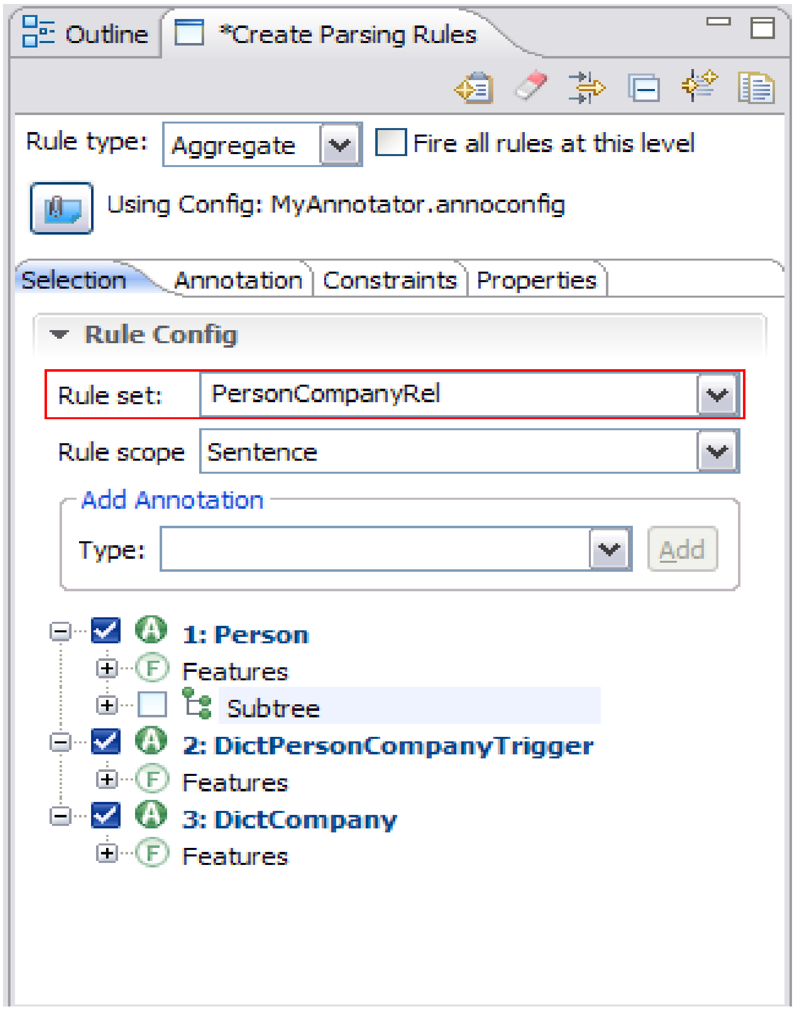


```
Amine works for IBM Ireland.
John Doe, the CEO of Automatics Inc., has said in his
speech that the company is looking for new investors.
```

Rule type: Aggregate
Fire all rules at this level
Using Config: MyAnnotator.annoconfig
Rule set: Default
Rule scope: Sentence
Type:

- ✓ 1: Person
  - Features
  - Subtree
- ✓ 2: DictPersonCompanyTrigger
  - Features
- ✓ 3: DictCompany
  - Features

- The selection of the components of the tree should not be changed; that is, the pattern that you want to match is a Person annotation, followed by a Person Company trigger and finally a company name.
- As explained before, the aggregate rule will not see the elements intervening between the annotations.
- Make sure you define the rule set, this is useful in the ordering of the rules. See the help for more information about "*Rules ordering*" for aggregate rules.

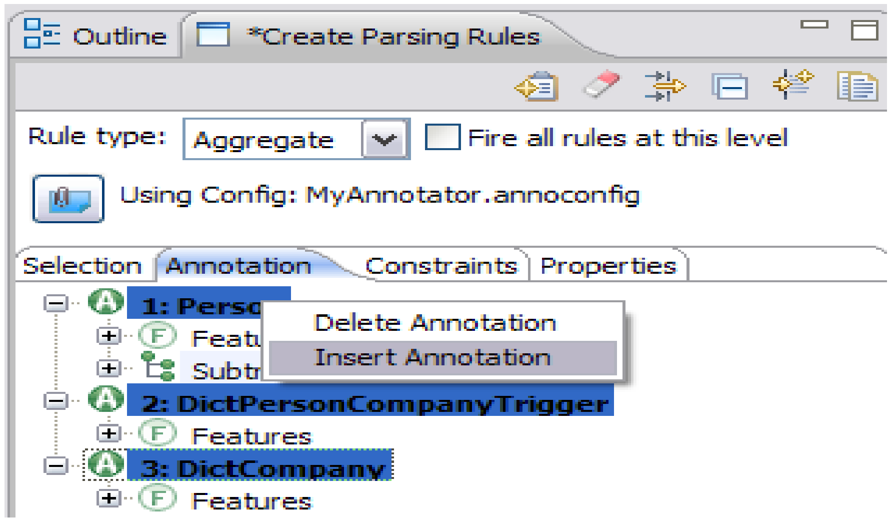


Rule type: Aggregate
Fire all rules at this level
Using Config: MyAnnotator.annoconfig

Rule set: PersonCompanyRel
Rule scope Sentence

1: Person
Features
Subtree
2: DictPersonCompanyTrigger
Features
3: DictCompany
Features

- Switch to the annotation Tab, select the three elements, then right click, and select insert annotation.



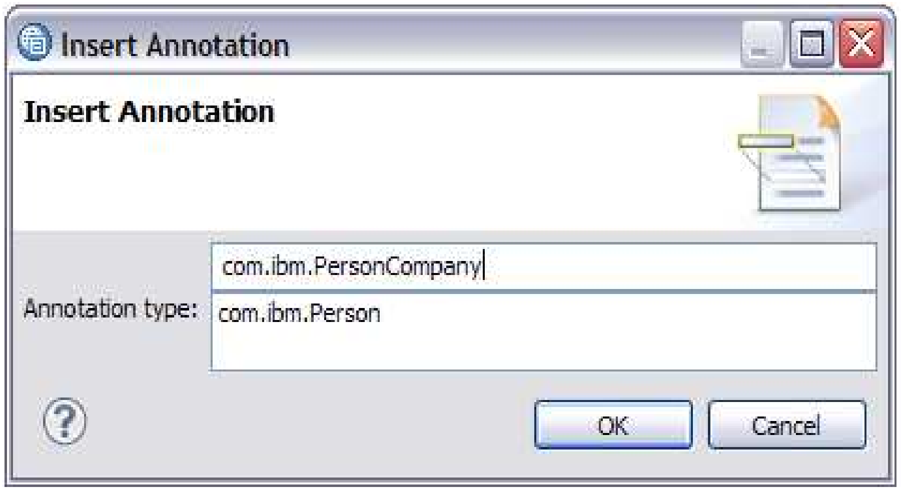
Rule type: Aggregate

Fire all rules at this level

Using Config: MyAnnotator.annoconfig

1: Perso
Featu
Subtr
2: DictPersonCompanyTrigger
Features
3: DictCompany
Features

- Specify the annotation name: **PersonCompany** in the dialog box that pops up, and click OK

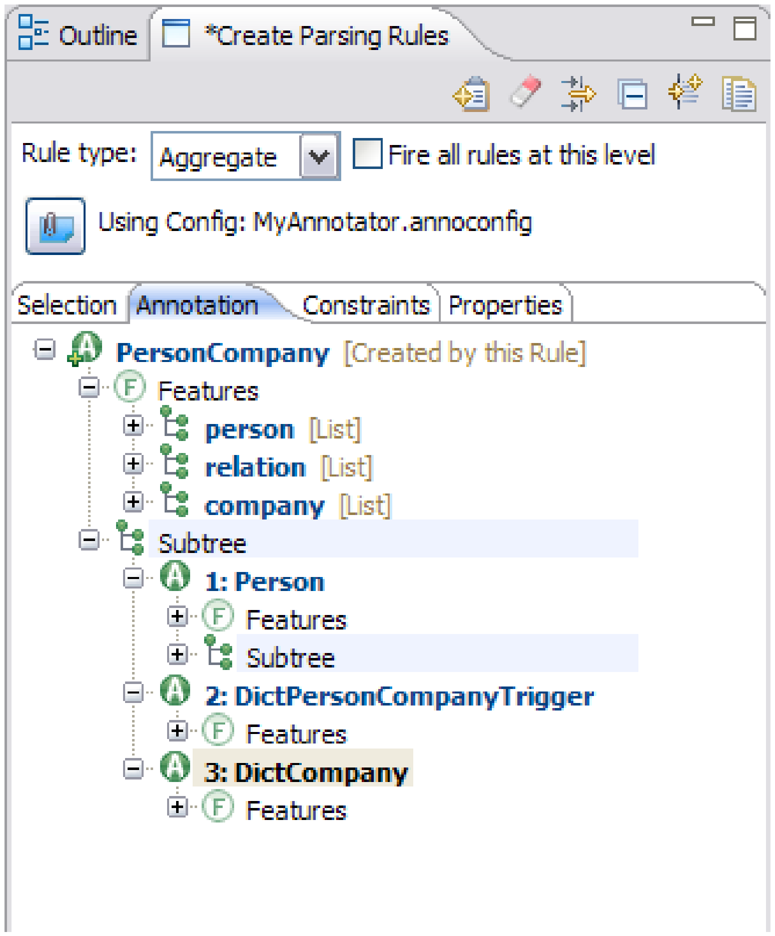


Annotation type: com.ibm.PersonCompany

com.ibm.Person

- Also create features for this annotation by dragging the sub-annotations into the features node and name them as follows:
  – Person => person
  – DictPersonCompanyTrigger => relation
  – DictCompany => company

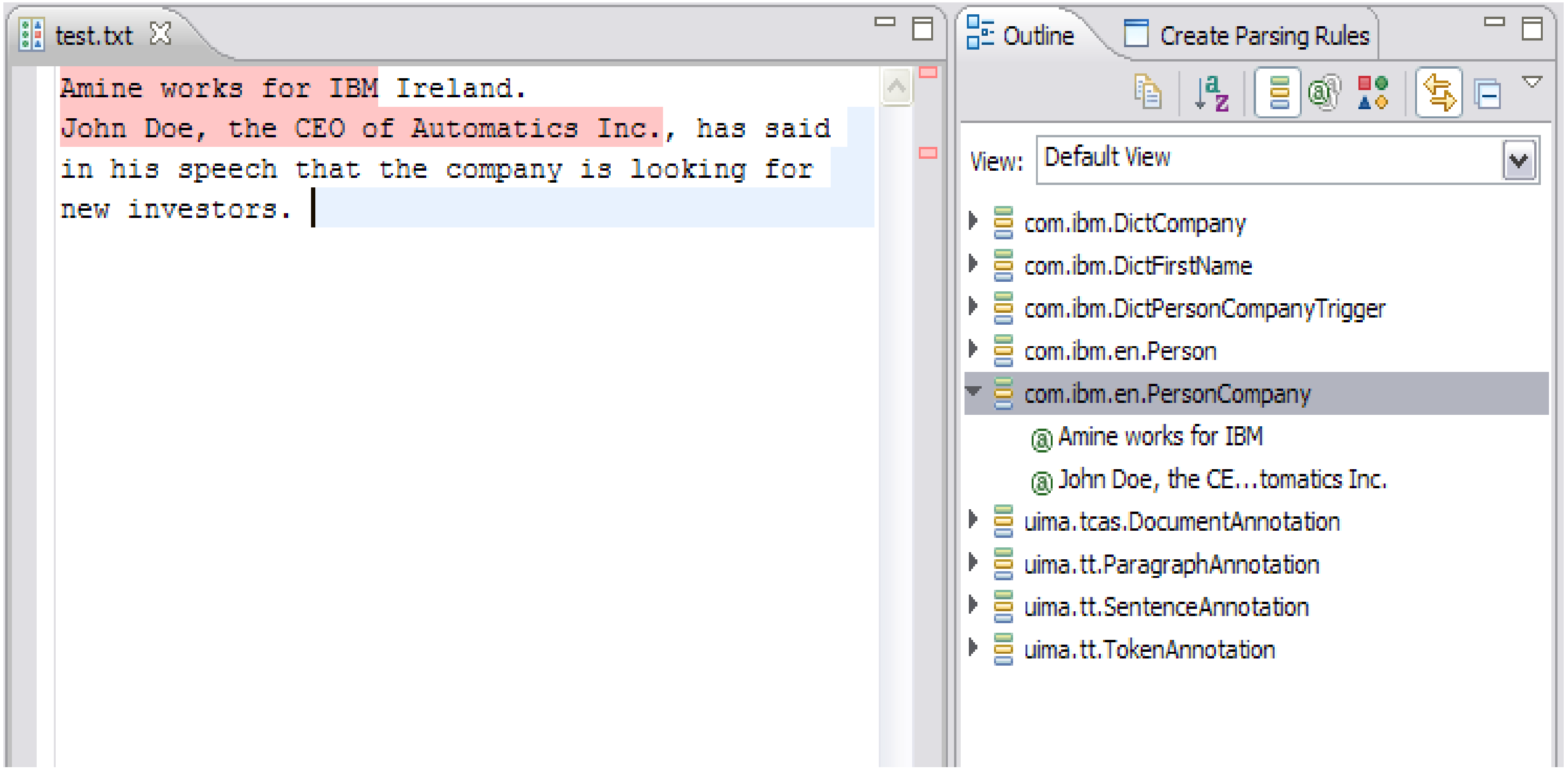
- Save the rule (click the icon) and build/compile the Parsing rules database



Rule type: Aggregate
Fire all rules at this level
Using Config: MyAnnotator.annoconfig

```
PersonCompany [Created by this Rule]
  Features
    person [List]
    relation [List]
    company [List]
  Subtree
    1: Person
      Features
      Subtree
    2: DictPersonCompanyTrigger
      Features
    3: DictCompany
      Features
```

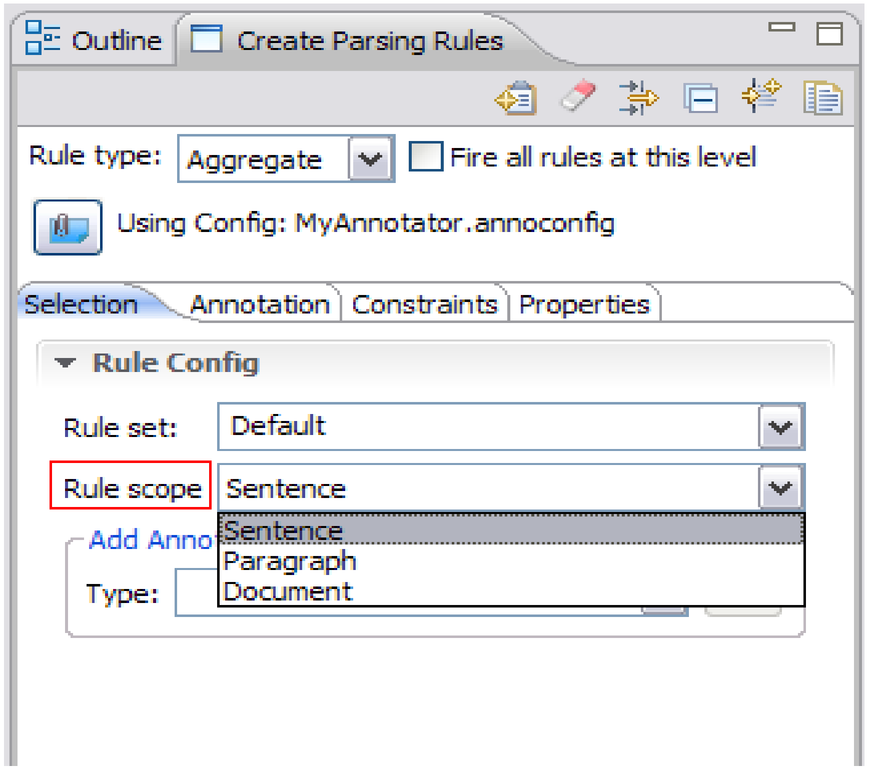
- Switch to the outline and see the new PersonCompany annotation. You can notice that two instances have been highlighted, so the rule matched the second sentence as the aggregate rule does not see the intervening elements..



```
Amine works for IBM Ireland.
John Doe, the CEO of Automatics Inc., has said
in his speech that the company is looking for
new investors.
```

```
com.ibm.DictCompany
com.ibm.DictFirstName
com.ibm.DictPersonCompanyTrigger
com.ibm.en.Person
com.ibm.en.PersonCompany
    Amine works for IBM
    John Doe, the CE...tomatics Inc.
uima.tcas.DocumentAnnotation
uima.tt.ParagraphAnnotation
uima.tt.SentenceAnnotation
uima.tt.TokenAnnotation
```

- Aggregate rules scope can be changed to cover a paragraph, or the entire document by setting the scope of the aggregate rule as shown in the screen capture.



Rule type: Aggregate
Fire all rules at this level
Using Config: MyAnnotator.annoconfig
Rule set: Default
Rule scope: Sentence
Sentence
Paragraph
Document

## Module roadmap

- **Create parsing rules.**
  - How do rules work?
  - What are the different parsing rules types and what are their characteristics?
  - How to create rules?
- **Summary and best practices**
- **Sample exercises**

## Module summary

You have completed this module and can:

- Create Annotations using Phrase rules
- Create Annotations using Aggregate rule
- Create Features
- See the result in the Outline and the Properties views.

See the LanguageWare help for more tips and advanced use cases.

## Best practices

- This Module covered two types of parsing rules: Phrase and Aggregate rules.
- Before starting to create rules, try to figure out the annotations you want to find, and the intermediary elements (indicators and triggers) that may be used to make the job easier.
- For string matching, it is best to create a dictionary for the Type (even if it contains only one entry).
- Make sure you specify the rule set for every rule. Group the rules by sets so they fire when appropriate and they do not break each other.

## Module roadmap

- **Create parsing rules.**
  - How do rules work?
  - What are the different parsing rules types and what are their characteristics?
  - How to create rules?
- **Summary and best practices**
- **Sample exercises**

## Practice exercises

- Create some rules in the "IdentifyQuestion" database to identify the questions in the "SampleQuestion" document.

## Contacts

- If you have any questions, comments or suggestions, contact us using the LanguageWare email address *EMEALAN@ie.ibm.com* or on the developerWorks® forum.

## Trademarks, copyrights, and disclaimers