europeana
connect

# D2.2.1 – Europeana Language Resources Repository

This deliverable is software.

co-funded by the European Union

The project is co-funded by the European Union, through the ***e*Content*plus*** programme

http://ec.europa.eu/econtentplus

Österreichische Nationalbibliothek

EuropeanaConnect is coordinated by the Austrian National Library


**ECP-2008-DILI-528001**

# EuropeanaConnect

## D2.2.1 – Europeana Language Resources Repository

| | |
|---|---|
| **Deliverable number/name** | *D2.2.1* |
| **Dissemination level** | *Public* |
| **Delivery date** | *31/07/2010* |
| **Status** | *v1.0* |
| **Author(s)** | *Aaron Kaplan, Anne Schiller* |
| **Contributors**[1] | *Alessio Bosca, Luca Dini (CELI); Jan Molendijk, Sjoerd Siebinga (EDL); Maria Gäde, Vivien Petras, Juliane Stiller (HUB); Nicola Ferro (UPD); Antoine Isaac (VUA); Aaron Kaplan, Anne Schiller (XEROX).* |

***e*Content*plus***

This project is funded under the *e*Content*plus* programme,
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.


EuropeanaConnect is coordinated by the Austrian National Library

[1] Contributors to the design and construction of the repository

# Description of software packaged/developed for Europeana within EuropeanaConnect

This deliverable is a collection of software and data for linguistic analysis of text. The resources were acquired from a number of different sources (with different owners and licensing conditions), and adapted to a common API to facilitate future integration into Europeana software.

| | |
|---|---|
| Link to software | https://europeanalabs.eu/core/svn/contrib/lrr/trunk |
| Login information | europeanalabs login and password |
| Development environment | Nothing IDE-specific |
| Programming language used | Resources exposed via Java and Web Service interfaces; underlying implementations are in a variety of languages. |
| Application server used | Some resources are available as Java classes that do local processing, no application server involved. For those that are web services, implementations are currently mixed: some based on JAX-WS, others on Axis2. |
| Database requirements | CELI TranslationDictionary implementation requires a DBMS for which a JDBC driver exists, and which supports UTF-8 collation. Tested with MYSQL. |
| Operating system requirements | See individual resource descriptors; all run on Linux, some can also run on other platforms. |
| Port requirements / default ports used | configurable |
| Interface | Both Java and Web Service interfaces provided |
| Licensing conditions | All of the resources are available for research use by EuropeanaConnect consortium partners during the course of the project. Conditions for use by others or after the end of the project are specific to each resource (see individual descriptors). |

# Table of Contents

Description of software packaged/developed for Europeana within EuropeanaConnect ..... 3
1 Introduction ..... 5
2 Accessing and using the repository ..... 7
2.1 Formats and APIs ..... 7
2.2 Repository structure ..... 8
3 Workflow ..... 10
4 Information about Language Resources ..... 11
4.1 General Information ..... 11
4.2 Resource description ..... 12
4.2.1 Stop word lists ..... 12
4.2.2 Language identifiers ..... 12
4.2.3 Morphological Analysers ..... 12
4.2.4 Named Entity Recognizers ..... 13
4.2.5 Translation dictionaries ..... 13
4.3 Evaluation information ..... 13
4.3.1 Stop word lists ..... 13
4.3.2 Language identifiers ..... 14
4.3.3 Morphological Analysers ..... 14
4.3.4 Named Entity Recognizers ..... 14
4.3.5 Translation dictionaries ..... 14
4.4 Technical Aspects ..... 14
4.5 Additional information ..... 15
4.6 Examples ..... 15
4.6.1 Xerox language tools ..... 15
4.6.2 XDXF bilingual dictionaries ..... 17
4.7 Registered Resources ..... 18
5 Publicly-accessible resource register ..... 19
Appendix A. XML Schema for LR Descriptions ..... 20
Appendix B. API Documentation ..... 22

# 1 Introduction

EuropeanaConnect work package 2 aims to provide multilingual access for international users. One task is to establish mappings between controlled vocabularies in different languages (T2.3); another is to set up translation modules or services for cross-lingual user queries. Such services rely on language resources: software and information for processing digital representations of human language in various ways. It is not in the scope of EuropeanaConnect to build such resources, but to collect and assess available resources and to adapt and maintain them as necessary.

The set of 10 languages that should be supported within EuropeanaConnect is divided into two groups:

- **Core Languages**: English, French, German, Italian, Polish, Spanish
- **Secondary Languages**: Dutch, Hungarian, Portuguese, Swedish

At the beginning of the project a survey about language resources has been sent out to the partners in WP2 to find out what is available among the partners. Certain resources from the resulting list were selected to be included in the Europeana Language Resource Repository as described in the following sections. Other resources may be added in the future as we become aware of new resources, as evaluations demonstrate that certain resources are more appropriate than others for the tasks at hand, or as new needs present themselves.

The repository currently contains the following types of resources which are necessary for query translation and vocabulary mapping:

- **Stop word lists:** lists of ”non-content” words, such as articles, conjunctions, prepositions, which can be ignored for specific tasks of processing, especially for indexing, retrieval or even translation. This resource will be mainly used by the indexer.
- **Language identifiers:** a tool that is necessary whenever the language of the query is not explicitly known. It will be used by the indexer and by the query translation module.
- **Morphological analyzers:** software modules that perform tokenization and lemmatization, but also decompounding, multi-word detection and part of speech tagging
- **Named entity recognizers:** software modules that identify named entities, such as person names, geographic names, organisation names, etc.
- **Translation dictionaries**: mappings between terms in different languages.

For each of these categories we define a set of descriptive features to guide and inform the user about the resources that are available.

More categories can be added, for example, if at a later stage of the project other types of language resources turn out to be necessary or useful for new functionalities.

The above categorization is not identical to that used in the Description of Work, for two reasons. First, the resource types listed in the DoW were given merely as examples; part of the work done in this work package was to understand the needs of the “client” tasks (T2.3 and T2.4), and the current set of resource types reflects that understanding. Second, several different kinds of resources that were listed as distinct types in the DoW, namely stemmers, normalizers,

lemmatizers, and phrase (multi-word expression) recognizers, have been grouped under the single rubric of morphological analyzers. This was done because a single low-level linguistic processing module typically implements several of the above functions, not just one.

In addition to the applicative resources mentioned above, the repository contains a set of Europeana related **test corpora** which may serve to evaluate, modify or even create new language resources.

# 2 Accessing and using the repository

The repository is currently available to Europeana partners at the following URL:

https://europeanalabs.eu/core/svn/contrib/lrr/trunk

This is a password-protected subversion repository that can be accessed by Europeana developers using their EuropeanaLabs login and password.

Access to the repository is restricted because some of the resources it contains are proprietary, acquired by Europeana under conditions that do not allow them to be redistributed freely. There are also issues regarding ownership of the code written in the course of the project by EuropeanaConnect industrial partners. Once these latter issues are resolved, the repository will be split into an open-source section that is publicly accessible, and a proprietary section that is accessible only to Europeana developers.

Note that while a password is required in order to download resources, a full list of resources with detailed information about them is available to the public via the register (see Section 5).

## 2.1 Formats and APIs

The repository contains both linguistic data and software. Resources have been collected from a variety of sources, and each source provides data in its own representation format. To facilitate the integration of resources into Europeana, the repository includes a layer of code that exposes all resources of a given type via a common Java API. The code for reading a resource in its native format and exposing the information it contains via the common API is stored in the repository alongside the data. The common format is an API, rather than a declarative representation format such as an XML format, because this allows a wider range of resource types to be covered. For example, a stemmer or lemmatizer provided as a C library cannot meaningfully be converted to an XML file, but it can be provided via a Java API. (Stop word lists are an exception: they are simple enough in structure that a simple declarative format suffices.) Wrapping all resources with a procedural interface also allows a uniform integration of resources that are available as external web services.

The formats for the five resource types are defined as follows:

- **Stop word lists**: a stop word list is represented as a simple text file, one word per line, encoded in UTF-8.
- **Language identifiers**: a language identifier must implement the LanguageIdentifier interface.
- **Morphological analysis components:** implement the MorphoAnalyzer interface.
- **Named entity recognition components:** implement the NamedEntityRecognizer interface.
- **Translation dictionaries**: implement the TranslationDictionary interface.

These interfaces are documented in Appendix B. The latest documentation can be generated from the source code by going to the directory `common/api` in the repository and running `mvn javadoc:javadoc`.

## 2.2 Repository structure

The top level of the repository contains a directory called `common` where files common to all resource types are stored. In particular, it has a subdirectory called `api` that contains the Java interface and class definitions that make up the APIs for using resources.

The `common` directory has one sibling for each resource type, currently:

```
language-identifiers
morphological-analyzers
named-entity-recognizers
stopword-lists
translation-dictionaries
```

Within each of these directories is one subdirectory per resource group. A resource group is a set of resources that share some files, typically because they come from the same provider or are based on the same platform. Each resource group directory contains a file called `Europeana-LR-descriptor.xml` that provides information about the group in a standard format that is defined in Section 4 below. The information from all the descriptor files in the repository will be aggregated and presented via the web as the Europeana Linguistic Resource Register (see Section 5). Alongside `Europeana-LR-descriptor.xml` is a `README` file providing technical details about the work that was done to adapt the resource to the standard API, and information that developers need in order to use the resource in a program. Information of potential interest to users of the resource goes in `Europeana-LR-descriptor.xml`, whereas information of interest only to the maintainer of the resource goes in `README`.
Besides the `Europeana-LR-descriptor.xml` and `README` files, a resource group directory contains one directory for each resource in the group. A resource directory is a maven project and conforms to the maven directory structure: at the top level is the file `pom.xml` that is used to build the resource, and a directory called `src` with subdirectory `main`. Below `src/main` are the following subdirectories:

- `orig` stores files in their original format and with their original file names, as received from the provider. Only the latest version of the files is present in a checkout of `orig`; previous versions can be retrieved using the version control mechanism.
- `modif` contains, if necessary, modified versions of the original files, or versions that have been transformed into other formats. Files that can be generated automatically from the originals should not be stored here; rather, the maven build process should generate them at compile time. Only if manual intervention is necessary should the results be stored here.[2]
- `java` contains java source code that exposes the resource via the Europeana API.

[2] This is an ideal that has sometimes proved difficult to attain, but we try to automate the build as much as is practical.

Whenever there are files that need to be shared between multiple subdirectories, those subdirectories have a sibling called `common`, which is itself a maven project, or contains multiple maven projects if necessary.

Versioning of APIs and resources is handled using the maven conventions. A subversion tag will be created for each non-SNAPSHOT version of an artifact.

# 3 Workflow

The following steps are involved in integrating a new resource into the repository:

- The data are stored, in their original format as received from the provider, with any associated code and documentation.
- The data are explored to evaluate their quality and understand how they can be used, and code is written to expose them via a standard API. This may or may not include code for transforming the data into a different format. During the course of the EuropeanaConnect project, this is being done by WP2 partners, on a set of resources sufficient for the needs of the project (query translation and vocabulary mapping for six core languages and four secondary languages). **To maintain and expand Europeana after the end of the project, Europeana Foundaton will need either to have staff with the necessary skills, or to contract with external service providers.** For each language covered, there is the need of a computational linguist (someone with both linguistic and programming skills) who is a native speaker of the selected language. For each translation pair there is the need of a translator with computational skills, good knowledge of the source language, and excellent knowledge of the target language. (Of course these two people might be the same in some cases.)
- In some cases, the data might be modified, e.g. to correct errors or to add new words.
- At a later time, a data provider might deliver a new version of an existing resource. This new version will be stored, appropriate conversion processes run again, and local modifications integrated into the new version.
- A developer will retrieve the data and associated code in order to deploy it on a test or production system.

# 4 Information about Language Resources

Each of the resources in the repository is accompanied by a descriptor in which we have compiled information about the resource and how it can be used. In this section we explain the fields of that descriptor.

## 4.1 General Information

For any type of resource the following information about ownership and legal aspects must be provided.

The Repository will contain both proprietary and open source language resources. Any usage restrictions or associated costs are stated. Anyone who would like to re-use non-free proprietary resources outside the Europeana context will have to directly contact[3] the provider.

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| name | name or identifier | a unique name that allows to identify the language resource | mandatory |
| type | “stop word list”, “language identifier”, “language analysis”, “translation dictionary” | one of the categories available in the Europeana repository | mandatory |
| place | URL | Access path in the Europeana repository | mandatory[4] |
| abstract | short text | a short general functional description | mandatory |
| license | “proprietary”, “open source” | Free or proprietary LR | mandatory |
| licence_note | “GPL v2.0”, “free for research” | More information about IPR aspects | recommended |
| origin | name or URL | Author or owner of the LR (CELI, Xerox, http://xxx.yyy.eu) | mandatory |
| contact | name, phone number, URL, ... | address, phone number or URL of the person to contact for request for licensing | mandatory if proprietary |
| fee | amount or “free” | cost for usage of LR | mandatory |
| API | “yes” OR “no” | Adapted to standard API or not | mandatory |
| status | “not yet evaluated”, “evaluated and rejected”, “to be adapted” | Information about the status of the LR with respect to its integration and access in the repository. | mandatory if API=no |

[3] The Web-interface for searching and browsing the language resources may include a simple contact form to be filled in and sent to the resource provider.

[4] This field will be generated automatically when converting the descriptor XML file into a public register web page (cf. section 5)

## 4.2 Resource description

### 4.2.1 Stop word lists

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| language | ISO-639-1 language code[5] | language covered by the list | mandatory |
| size | number | number of words in the list | recommended |

### 4.2.2 Language identifiers

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| language | ISO-639-1 language codes | languages that can be identified | mandatory |

### 4.2.3 Morphological Analysers

This category groups different types or levels of basic linguistic analysis. It covers word segmentation and stemming, simple lemmatization or more complex morphological analysis. It may also include decompounding (e.g. for German, Dutch, Hungarian) and part of speech disambiguation.

The function also determines the shape of the provided output: the output of a stemmer is a string representing the stem of a word which corresponds to a substring of the word form. A lemmatizer instead provides a base form or “lemma” which corresponds to the citation form found in a dictionary. A morphological analyser returns a lemma plus morphological information such as grammatical category (noun, adjective, verb, ...) and other features (tense, number, gender, etc.).

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| language | ISO-639-1 language code | language to be analyzed | mandatory |
| function | “segmentation”, “stemming”, “lemmatization”, “POS tagging” | describes the level or components of the analysis | mandatory |
| size | number | number of lemmas or full forms in the lexicon | recommended |

[5] We use two-letter codes (Cf. http://www.loc.gov/standards/iso639-2/php/code_list.php) which are sufficient to cover the Europeana languages.

### 4.2.4 Named Entity Recognizers

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| language | ISO-639-1 language code | language to be analyzed | mandatory |
| NE types | “person”, “country”, “river”, ... | list or description of the types (and granularity) of the recognized entities[6] | mandatory |

### 4.2.5 Translation dictionaries

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| language | ISO-639-1 language codes | source and target language(s) | mandatory |
| Size | Number | number of translation pairs (or tuples) | recommended |
| ambiguity | number | average number of translations per source word | optional |

## 4.3 Evaluation information

The choice of a language resource will usually not just depend on its functionality, but very much on its quality. As discussed in a Memo[7] about evaluation criteria it is difficult to provide all quality indicators that one would wish to see. The tables below contain a minimum set of criteria that can be provided by the authors or other sources, or can be computed on the basis of a test corpus made of **user queries** from the Europeana ClickStreamLogs and prototype **data sets** from Europeana. More indicators, such as speed, maintainability, etc., can be included if available. Test corpora for the 10 Europeana languages are available in the language repository as well.

### 4.3.1 Stop word lists

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| - none - | | | |

[6] We have not defined a standard set of named entity types. Each extractor defines its own set of types, and if extractors with different types are to be used together in a system, their types will have to be mapped to a common (application-specific) set. The Tagset class in the Europeana API helps formalize the types used by each resource—see the javadoc for more details.

[7] Anne Schiller, *Europeana Language Resources - Evaluation Criteria.* https://version1.europeana.eu/c/document_library/get_file?p_l_id=16989&folderId=26117&name=DLFE-9226.doc

### 4.3.2 Language identifiers

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| accuracy | percentage | correctly identified languages | recommended |

### 4.3.3 Morphological Analysers

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| coverage | percentage[8] | Tokens of a test corpus that can be analysed | recommended |
| accuracy | percentage | tokens that are correctly analysed | optional |

### 4.3.4 Named Entity Recognizers

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| coverage | percentage | tokens that are analysed as named entities | recommended |
| accuracy | percentage | tokens that are correctly analysed as NEs | optional |
| ambiguity | number | average number of different NE types per token | optional |

### 4.3.5 Translation dictionaries

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| coverage | percentage | query words that have a translation | recommended |
| accuracy | percentage | query words that are correctly analysed | optional |

## 4.4 Technical Aspects

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| platform | “web service”, “Windows”, “Linux”, “platform independent”, ... | Where or how to use the LR | mandatory |
| speed | number | analysis time (may depend on the platform) | optional |

[8] Including a reference or some information about the corpus in which the coverage has been tested.

## 4.5 Additional information

Examples and comments can be added for all categories of Language Resources.

| Feature | Value(s) | Description | Status |
|---|---|---|---|
| example | text | Samples to illustrate the entries (of textual resources) or input/output (of analysis tools) - mainly for resources that are not (yet) available in the standard API. | recommended |
| comment | text | Any additional useful information about the resource, its use, availability or evaluation. | optional |
| internal | text | Technical notes for or about the integration of the LR into the repository. They should be part of the description in the repository, but not displayed on the wiki for the users. | optional |

## 4.6 Examples

### 4.6.1 Xerox language tools

#### 4.6.1.1 XIP for English

| Feature | Value(s) |
|---|---|
| name | XIP for English |
| type | morphological analyser, named entity recognizer |
| abstract | The tool integrates a finite-state morphological analyser, an HMM tagger and rules for chunking, dependency extraction and named entity recognition to perform different levels of text analysis. |
| license | proprietary |
| origin | Xerox |
| contact | Mathieu.Chuat@xerox.com or info@xrce.xerox.com |
| fee | (negotiations pending) |
| status | in process of being adapted |
| language | EN |
| function | segmentation, lemmatization, named entity recognition, dependency parsing |
| coverage | 100 % |
| NE type | person, location, organisation, date, time |
| comment | Full coverage is due to the fact that XIP is a robust parser and always provides some output for any input. |
| platform | Windows, Linux, Solaris, MacOs |

#### 4.6.1.2 XIP for Hungarian

| Feature | Value(s) |
|---|---|
| name | XIP for Hungarian |
| type | morphological analyser |
| abstract | A tool for morphological analysis including derivation and compounding |
| license | proprietary |
| origin | Xerox |
| contact | Mathieu.Chuat@xerox.com or info@xrce.xerox.com |
| fee | (negotiations pending) |
| status | to be adapted to standard API |
| language | HU |
| function | segmentation, lemmatization, morphological analysis |
| coverage | 100 % |
| comment | XIP includes a guesser for non-lexicalized words. Therefore full coverage is guaranteed. |
| platform | Windows, Linux, Solaris, MacOs |

### 4.6.2 XDXF bilingual dictionaries

#### 4.6.2.1 German-English

| Feature | Value(s) |
|---|---|
| name | XDF German-English |
| type | translation dictionary |
| abstract | German-English bilingual dictionary |
| license | GPL |
| origin | http://xdxf.revdanica.com/ |
| contact | -- |
| fee | none |
| status | not yet evaluated |
| language | DE-EN |
| size | 96,000 entries |
| comment | The key entries for the source language are base forms, there are no categories, but gender for nouns and additional annotations or examples. The target consists of one or multiple base forms (no category). |
| example | <ar><br><k>America</k><br><pos>NOUN</pos><br><neCategory>LOCORG</neCategory><br><dtrn><br><k>Amérique</k><br><pos>NOUN</pos><br><neCategory>LOCORG</neCategory><br></dtrn><br><dtrn><br><k>Etats Unis</k><br><pos>NOUN</pos><br><neCategory>LOCATION</neCategory><br></dtrn><br><dtrn><br><k>EU</k><br><pos>NOUN</pos><br><neCategory>LOCORG</neCategory><br></dtrn><br></ar> |
| platform | platform independent |

## 4.7 Registered Resources

The repository registers the following resources[9]

- 16 stop word lists (covering all 10 EuropeanaConnect languages)
- 1 language identifier (covering all 10 languages and implementing the Europeana API)
- 14 morphological analyzers (covering all 10 languages; at this writing at least one implementation of the Europeana API is available for all but ES.)
- 3 named entity recognizers (for EN, FR, DE)
- 57 translation dictionaries

[9]The complete list of XML descriptions is available in Appendix C

# 5 Publicly-accessible resource register

The initial motivation for collecting and adapting linguistic resources was to support query translation and vocabulary mapping work within the EuropeanaConnect project, but the resources are potentially of interest to developers outside of Europeana as well. We therefore provide a web interface that outsiders can use to see what resources have been selected for use in Europeana and adapted to the standard Europeana APIs, the licensing conditions under which the resources are available, and contact addresses for the rights holders.

The information displayed in the register is compiled automatically from the descriptor files present in the repository (see Section 2.2). Currently a script generates static HTML pages from these descriptor files, but the XML format of the descriptors was chosen to facilitate a move to a database-backed search application should the size of the repository justify it in the future.

The register can be found at the following address:

http://europeanalabs.eu/wiki/LinguisticResourceRegister

# Appendix A. XML Schema for LR Descriptions

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="repository" type="Repository"/>
       <!-- define values for "type" -->
    <xs:simpleType name="Type">
        <xs:restriction base="xs:string">
            <xs:enumeration value="stop word list"/>
            <xs:enumeration value="language identifier"/>
            <xs:enumeration value="morphological analyzer"/>
            <xs:enumeration value="named entity recognizer"/>
            <xs:enumeration value="translation dictionary"/>
        <!-- add more values as appropriate (if necessary) -->
    </xs:simpleType>
       <!-- define values for "licence" -->
    <xs:simpleType name="Licence">
        <xs:restriction base="xs:string">
            <xs:enumeration value="open source"/>
            <xs:enumeration value="proprietary"/>
        </xs:restriction>
    </xs:simpleType>
       <!-- define values for "status" -->
    <xs:simpleType name="API">
        <xs:restriction base="xs:string">
            <xs:enumeration value="yes"/>
            <xs:enumeration value="no"/>
        </xs:restriction>
    </xs:simpleType>
       <!-- define values for "language" -->
       <!-- these are ISO-639 2-letter codes -->
    <xs:simpleType name="LangVal">
        <xs:restriction base="xs:string">
            <xs:enumeration value="de"/>
            <xs:enumeration value="en"/>
            <xs:enumeration value="es"/>
            <xs:enumeration value="fr"/>
            <xs:enumeration value="hu"/>
            <xs:enumeration value="it"/>
            <xs:enumeration value="nl"/>
            <xs:enumeration value="pl"/>
            <xs:enumeration value="pt"/>
            <xs:enumeration value="sv"/>
       [ … ]
            <xs:enumeration value=""/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="Language">
        <xs:simpleContent>
            <xs:extension base="LangVal">
                <xs:attribute name="dir" type="xs:string"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
       <!-- define values for "function" -->
    <xs:simpleType name="Function">
        <xs:restriction base="xs:string">
            <xs:enumeration value="segmentation"/>
            <xs:enumeration value="stemming"/>
            <xs:enumeration value="lemmatization"/>
            <xs:enumeration value="morphological analysis"/>
            <xs:enumeration value="decompounding"/>
            <xs:enumeration value="POS tagging"/>
            <xs:enumeration value="chunking"/>
            <xs:enumeration value="dependency parsing"/>
            <xs:enumeration value="named entity recognition"/>
            <!-- add more values as appropriate -->
            <xs:enumeration value=""/>
        </xs:restriction>
    </xs:simpleType>
```

```
        <!-- define values for "platform" -->
    <xs:simpleType name="Platform">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Linux"/>
            <xs:enumeration value="SunOS"/>
            <xs:enumeration value="MacOS"/>
            <xs:enumeration value="Windows"/>
            <xs:enumeration value="platform independent"/>
            <xs:enumeration value="Web service"/>
            <xs:enumeration value="online demo"/>
            <xs:enumeration value="other"/>
        </xs:restriction>
    </xs:simpleType>
        <!-- samples with input/output fields -->
    <xs:complexType mixed="true" name="Example">
        <xs:sequence>
            <xs:element name="input" type="xs:string" minOccurs="0"/>
            <xs:element name="output" type="xs:string" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
        <!-- element  language_resource -->
    <xs:complexType name="LanguageResource">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="path" type="xs:string" minOccurs="1" maxOccurs="unbounded"/>
            <xs:element name="type" type="Type" maxOccurs="unbounded"/>
            <xs:element name="abstract" type="xs:string"/>
            <xs:element name="licence" type="Licence"/>
            <xs:element name="licence_note" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="origin" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="contact" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="fee" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="API" type="API"/>
            <xs:element name="status" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="language" type="Language" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="size" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="function" type="Function" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="NE_types" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="ambiguity" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="coverage" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="accuracy" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="platform" type="Platform" maxOccurs="unbounded"/>
            <xs:element name="speed" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="example" type="Example" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="comment" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="internal" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
        <!-- list of resources -->
    <xs:complexType name="Repository">
        <xs:sequence>
            <xs:element name="language_resource" type="LanguageResource"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

# Appendix B. API Documentation

# Package eu.europeana.linguistic.api

The APIs for using resources from the Europeana Linguistic Resource Repository.

**See:**

Description

| Interface Summary | | Page |
|---|---|---|
| *LanguageIdentifier* | LanguageIdentifier interface is the core interface for the language identification in Europeana. | 37 |
| *MorphoAnalyzer* | A service that performs morphological analysis, which may cover any of: segmentation, stemming/lemmatization, and part of speech tagging. | 39 |
| *MorphoAndNamedEntity* | A component that does the jobs of a MorphoAnalyzer and a NamedEntityRecognizer simultaneously. | 40 |
| *NamedEntityRecognizer* | A service that performs named entity extraction from text. | 61 |
| *TranslationDictionary* | Provides lookup in one or more bilingual dictionaries. | 72 |

| Class Summary | | Page |
|---|---|---|
| Annotation | Associates a Tag with the span of text determined by getStart() and getEnd(). | 25 |
| Constants | Provides convenience method for easy access to the Europeana common part of speech tagset. | 27 |
| GuessedLanguage | Represents the output of a LanguageIdentifier and extends the base Language class with weight, a floating point value in interval [0..1] that represents the estimated probability of the guessed language. | 28 |
| Language | Represents a human language. | 32 |
| MorphoAndNamedEntity.Result | | 42 |
| MorphoResult | Represents the results of morphological analysis, which covers segmentation, lemmatization, and part of speech tagging. | 43 |
| MorphoResult.ComposedWordForm | | 45 |
| MorphoResult.Disjunction | Consistency constraints that implementations must satisfy:<br>• All alternatives of a Disjunction should have the same start and end. | 47 |
| MorphoResult.GroundWordForm | | 50 |
| MorphoResult.Sequence | | 52 |
| MorphoResult.SequenceElement | | 54 |
| MorphoResult.WordForm | Represents a word, a multi-word term, or a sub-word unit of a compound word. | 56 |
| NamedEntityRecognizer.Result | | 62 |
| Tagset | A Tagset is a fixed set of labels, such as parts of speech or named entity classes. | 65 |
| Tagset.Tag | | 68 |
| TagsetTypeAdapter | | 70 |
| TranslationWithPosition | This is part of the TranslationDictionary API. | 74 |
| Word | This is part of the TranslationDictionary API. | 79 |

| Enum Summary | | Page |
|---|---|---|
| MorphoResult.WordFormType | | 59 |

| Exception Summary | | Page |
|---|---|---|
| InitializationException | Indicates that something went wrong when initializating a linguistic processing resource, e.g. | 31 |
| Language.BadLanguageException | | 36 |
| LinguisticProcessingException | | 38 |
| NoSuchResourceException | | 64 |
| UnsupportedLanguageException | | 77 |

# Package eu.europeana.linguistic.api Description

The APIs for using resources from the Europeana Linguistic Resource Repository.

There is one interface for each type of resource:

- LanguageIdentifier
- MorphoAnalyzer
- NamedEntityRecognizer
- TranslationDictionary

The interface MorphoAndNamedEntity is a convenience for components that perform both morphological analysis and named entity recognition.

Most of these interfaces are parameterized by a Tagset. A Tagset is a predefined list of tags or categories, for example a set of grammatical categories (noun, verb, ...) or a set of named entity types (person, country,...). To be able to use a resource, it is not sufficient to know that it implements a particular interface; one must also know which Tagset it uses, understand the meaning of the Tags in that Tagset, and typically determine how to map those Tags to categories meaningful for one's application.

# Class Annotation

eu.europeana.linguistic.api

java.lang.Object

└ **eu.europeana.linguistic.api.Annotation**

---

public class **Annotation**

extends Object

Associates a Tag with the span of text determined by getStart() and getEnd().

---

| Constructor Summary | Page |
|---|---|
| **Annotation**(int start, int end, Tagset.Tag tag) | 25 |

| Method Summary | | Page |
|---|---|---|
| boolean | **equals**(Object o) | 26 |
| int | **getEnd**() | 25 |
| int | **getStart**() | 25 |
| Tagset.Tag | **getTag**() | 25 |
| int | **hashCode**() | 26 |

## Constructor Detail

### Annotation

public **Annotation**(int start,
int end,
Tagset.Tag tag)

## Method Detail

### getStart

public int **getStart**()

---

### getEnd

public int **getEnd**()

---

### getTag

public Tagset.Tag **getTag**()

## equals

```
public boolean equals(Object o)
```

**Overrides:**

`equals` in class `Object`

## hashCode

```
public int hashCode()
```

**Overrides:**

`hashCode` in class `Object`

# Class Constants

eu.europeana.linguistic.api

java.lang.Object

└ **eu.europeana.linguistic.api.Constants**

---

public class **Constants**

extends Object

Provides convenience method for easy access to the Europeana common part of speech tagset.

---

| Field Summary | | Page |
|---|---|---|
| static Tagset | **EUROPEANA_POS_TAGSET** | 27 |

| Constructor Summary | Page |
|---|---|
| **Constants**() | 27 |

| Method Summary | | Page |
|---|---|---|
| static Tagset.Tag | **getEuropeanaTag**(String vendorString)<br>Returns the Tag from the Europeana PoS tagset with the given vendorString | 27 |

## Field Detail

### EUROPEANA_POS_TAGSET

public static final Tagset **EUROPEANA_POS_TAGSET**

## Constructor Detail

### Constants

public **Constants**()

## Method Detail

### getEuropeanaTag

public static Tagset.Tag **getEuropeanaTag**(String vendorString)

Returns the Tag from the Europeana PoS tagset with the given vendorString

# Class GuessedLanguage

eu.europeana.linguistic.api

java.lang.Object

└ eu.europeana.linguistic.api.Language

└ **eu.europeana.linguistic.api.GuessedLanguage**

**All Implemented Interfaces:**

Comparable<GuessedLanguage>

---

public class **GuessedLanguage**

extends Language

implements Comparable<GuessedLanguage>

Represents the output of a LanguageIdentifier and extends the base Language class with weight, a floating point value in interval [0..1] that represents the estimated probability of the guessed language.

---

| Nested classes/interfaces inherited from class eu.europeana.linguistic.api.Language |
|---|
| Language.BadLanguageException |

| Field Summary | | Page |
|---|---|---|
| protected float | **guessConfidence**<br>represents the confidence of the guess | 29 |

| Fields inherited from class eu.europeana.linguistic.api.Language |
|---|
| DE, EN, ES, FR, HU, IT, NL, PL, PT, SV, ZZ |

| Constructor Summary | Page |
|---|---|
| **GuessedLanguage**() | 29 |
| **GuessedLanguage**(Language langCode, float weight) | 29 |

| Method Summary | | Page |
|---|---|---|
| int | **compareTo**(GuessedLanguage l) | 29 |
| float | **getGuessConfidence**() | 29 |
| String | **getGuessedLanguageCodeAsString**()<br>Returns a textual representation of the Language | 29 |
| void | **setGuessConfidence**(float f) | 29 |
| String | **toString**()<br>Returns a string representation of the language, e.g. | 30 |

| **Methods inherited from class eu.europeana.linguistic.api.Language** |
| --- |
| equals, getId, hashCode, setId, valid |

# Field Detail

## guessConfidence

```
protected float guessConfidence
```

represents the confidence of the guess

# Constructor Detail

## GuessedLanguage

```
public GuessedLanguage()
```

---

## GuessedLanguage

```
public GuessedLanguage(Language langCode,
                       float weight)
                throws Language.BadLanguageException
```

# Method Detail

## getGuessConfidence

```
public float getGuessConfidence()
```

---

## setGuessConfidence

```
public void setGuessConfidence(float f)
```

---

## getGuessedLanguageCodeAsString

```
public String getGuessedLanguageCodeAsString()
```

Returns a textual representation of the Language

---

## compareTo

```
public int compareTo(GuessedLanguage l)
```

**Specified by:**

`compareTo` in interface `Comparable`

## toString

```
public String toString()
```

Returns a string representation of the language, e.g. "en" for English. Use this only for informational purposes, e.g. in log messages. To compare two Languages, use equals().

**Overrides:**

`toString` in class `Language`

# Class InitializationException

eu.europeana.linguistic.api

```
java.lang.Object
  └ java.lang.Throwable
      └ java.lang.Exception
          └ eu.europeana.linguistic.api.InitializationException
```

**All Implemented Interfaces:**

Serializable

---

```
public class InitializationException
extends Exception
```

Indicates that something went wrong when initializating a linguistic processing resource, e.g. a configuration file was invalid.

---

| Constructor Summary | Page |
|---|---|
| InitializationException() | 31 |
| InitializationException(String message) | 31 |
| InitializationException(String message, Throwable cause) | 31 |
| InitializationException(Throwable cause) | 31 |

## Constructor Detail

### InitializationException

```
public InitializationException()
```

---

### InitializationException

```
public InitializationException(String message)
```

---

### InitializationException

```
public InitializationException(Throwable cause)
```

---

### InitializationException

```
public InitializationException(String message,
                               Throwable cause)
```

# Class Language

eu.europeana.linguistic.api

java.lang.Object

└ **eu.europeana.linguistic.api.Language**

**Direct Known Subclasses:**

GuessedLanguage

---

public class **Language**

extends Object

Represents a human language. This class provides convenience constants for the ten Europeana languages, but other languages can be represented using the Language(String) constructor.

---

| Nested Class Summary | | Page |
|---|---|---|
| class | **Language.BadLanguageException** | 36 |

| Field Summary | | Page |
|---|---|---|
| static Language | **DE** | 33 |
| static Language | **EN** | 33 |
| static Language | **ES** | 33 |
| static Language | **FR** | 33 |
| static Language | **HU** | 33 |
| static Language | **IT** | 33 |
| static Language | **NL** | 33 |
| static Language | **PL** | 34 |
| static Language | **PT** | 34 |
| static Language | **SV** | 34 |
| static Language | **ZZ** | 34 |

| Constructor Summary | | Page |
|---|---|---|
| | **Language**() | 34 |
| protected | **Language**(Language l) | 34 |
| | **Language**(String id)<br>Create a new Language from a language code. | 34 |

| Method Summary | | Page |
|---|---|---|
| boolean | **equals**(Object o) | 35 |
| String | **getId**() | 34 |
| int | **hashCode**() | 35 |
| void | **setId**(String id) | 35 |
| String | **toString**()<br>Returns a string representation of the language, e.g. | 35 |
| static boolean | **valid**(String s)<br>Returns true iff the given string is a valid language code. | 35 |

## Field Detail

### DE

public static Language **DE**

---

### EN

public static Language **EN**

---

### ES

public static Language **ES**

---

### FR

public static Language **FR**

---

### HU

public static Language **HU**

---

### IT

public static Language **IT**

---

### NL

public static Language **NL**

---

### PL

```
public static Language PL
```

---

### PT

```
public static Language PT
```

---

### SV

```
public static Language SV
```

---

### ZZ

```
public static Language ZZ
```

## Constructor Detail

### Language

```
protected Language(Language l)
```

---

### Language

```
public Language(String id)
         throws Language.BadLanguageException
```

Create a new Language from a language code.

**Throws:**

Language.BadLanguageException - if the given string is not a valid language code as defined by {valid(String).

---

### Language

```
public Language()
```

## Method Detail

### getId

```
public String getId()
```

---

### setId

```
public void setId(String id)
```

---

### valid

```
public static boolean valid(String s)
```

Returns true iff the given string is a valid language code. Valid language codes are two lowercase ASCII letters.

---

### toString

```
public String toString()
```

Returns a string representation of the language, e.g. "en" for English. Use this only for informational purposes, e.g. in log messages. To compare two Languages, use equals().

**Overrides:**

`toString` in class `Object`

---

### equals

```
public boolean equals(Object o)
```

**Overrides:**

`equals` in class `Object`

---

### hashCode

```
public int hashCode()
```

**Overrides:**

`hashCode` in class `Object`

# Class Language.BadLanguageException

eu.europeana.linguistic.api

```
java.lang.Object
  └ java.lang.Throwable
      └ java.lang.Exception
          └ eu.europeana.linguistic.api.Language.BadLanguageException
```

**All Implemented Interfaces:**

Serializable

**Enclosing class:**

Language

---

```
public class Language.BadLanguageException
extends Exception
```

---

| Constructor Summary | Page |
|---|---|
| Language.BadLanguageException() | 36 |

## Constructor Detail

### Language.BadLanguageException

```
public Language.BadLanguageException()
```

# Interface LanguageIdentifier

**eu.europeana.linguistic.api**

---

`public interface` **`LanguageIdentifier`**

LanguageIdentifier interface is the core interface for the language identification in Europeana.

---

| Method Summary | | Page |
|---|---|---|
| GuessedLanguage | **guessLanguage**(String text)<br>Guesses the language of a text fragment | *37* |

## Method Detail

### guessLanguage

GuessedLanguage **guessLanguage**(String text)

Guesses the language of a text fragment

**Parameters:**

`text` - The input text, the language of should be guessed

**Returns:**

A language (a language and a float in [0...1] and representing the identification confidence)

# Class LinguisticProcessingException

eu.europeana.linguistic.api

```
java.lang.Object
  └ java.lang.Throwable
      └ java.lang.Exception
          └ eu.europeana.linguistic.api.LinguisticProcessingException
```

**All Implemented Interfaces:**

Serializable

---

```
public class LinguisticProcessingException
extends Exception
```

---

| Constructor Summary | Page |
|---|---|
| LinguisticProcessingException() | 38 |
| LinguisticProcessingException(String message) | 38 |
| LinguisticProcessingException(String message, Throwable cause) | 38 |
| LinguisticProcessingException(Throwable cause) | 38 |

## Constructor Detail

### LinguisticProcessingException

```
public LinguisticProcessingException()
```

---

### LinguisticProcessingException

```
public LinguisticProcessingException(String message)
```

---

### LinguisticProcessingException

```
public LinguisticProcessingException(Throwable cause)
```

---

### LinguisticProcessingException

```
public LinguisticProcessingException(String message,
                                     Throwable cause)
```

# Interface MorphoAnalyzer

eu.europeana.linguistic.api

public interface **MorphoAnalyzer**

A service that performs morphological analysis, which may cover any of: segmentation, stemming/lemmatization, and part of speech tagging. Segmentation may include tokenization, decompounding, and multi-word detection. See MorphoResult for detailed description of the result representation.

| Method Summary | | Page |
|---|---|---|
| List<Language> | **getSupportedLanguages**() | 39 |
| Tagset | **getTagset**(Language lang)<br>Returns the tagset used for language lang, or null if that language is not supported. | 39 |
| MorphoResult | **process**(String string, Language lang)<br>Performs linguistic processing | 39 |

## Method Detail

### getSupportedLanguages

List<Language> **getSupportedLanguages**()

### getTagset

Tagset **getTagset**(Language lang)

Returns the tagset used for language lang, or null if that language is not supported.

### process

MorphoResult **process**(String string,
Language lang)
throws LinguisticProcessingException

Performs linguistic processing

**Throws:**

LinguisticProcessingException

# Interface MorphoAndNamedEntity

eu.europeana.linguistic.api

public interface **MorphoAndNamedEntity**

A component that does the jobs of a MorphoAnalyzer and a NamedEntityRecognizer simultaneously.

| Nested Class Summary | | Page |
|---|---|---|
| static class | MorphoAndNamedEntity.Result | 42 |

| Method Summary | | Page |
|---|---|---|
| Tagset | **getEntityTagset**(Language lang)<br>See NamedEntityRecognizer.getTagset(Language) | 40 |
| List<Language> | **getSupportedLanguages**() | 40 |
| Tagset | **getSyntacticTagset**(Language lang)<br>See MorphoAnalyzer.getTagset(Language) | 40 |
| MorphoAndNamedEntity.Result | **process**(String text, Language lang)<br>see MorphoAnalyzer.process(String, Language) | 41 |

## Method Detail

### getSupportedLanguages

List<Language> **getSupportedLanguages**()

### getSyntacticTagset

Tagset **getSyntacticTagset**(Language lang)

See MorphoAnalyzer.getTagset(Language)

### getEntityTagset

Tagset **getEntityTagset**(Language lang)

See NamedEntityRecognizer.getTagset(Language)

## process

```
MorphoAndNamedEntity.Result process(String text,
                                    Language lang)
                             throws LinguisticProcessingException
```

see `MorphoAnalyzer.process(String, Language)`

**Throws:**

`LinguisticProcessingException`

# Class MorphoAndNamedEntity.Result

eu.europeana.linguistic.api

java.lang.Object

└ **eu.europeana.linguistic.api.MorphoAndNamedEntity.Result**

**Enclosing class:**

MorphoAndNamedEntity

---

public static class **MorphoAndNamedEntity.Result**

extends Object

---

| Constructor Summary | Page |
|---|---|
| **MorphoAndNamedEntity.Result**(MorphoResult morphoResult, NamedEntityRecognizer.Result entityResult) | 42 |

| Method Summary | | Page |
|---|---|---|
| NamedEntityRecognizer.Result | **getEntityResult**() | 42 |
| MorphoResult | **getMorphoResult**() | 42 |

## Constructor Detail

### MorphoAndNamedEntity.Result

public **MorphoAndNamedEntity.Result**(MorphoResult morphoResult,
NamedEntityRecognizer.Result entityResult)

## Method Detail

### getEntityResult

public NamedEntityRecognizer.Result **getEntityResult**()

---

### getMorphoResult

public MorphoResult **getMorphoResult**()

# Class MorphoResult

eu.europeana.linguistic.api

```
java.lang.Object
  └eu.europeana.linguistic.api.MorphoResult
```

---

```
public class MorphoResult
extends Object
```

Represents the results of morphological analysis, which covers segmentation, lemmatization, and part of speech tagging. Segmentation may include tokenization, decompounding, and multi-word detection. The representation allows for ambiguous results.

This representation is inspired by ISO/DIS 24611, the Morphosyntactic Annotation Framework (MAF), but is simpler. Our representation can be translated straightforwardly into MAF. MAF can also be directly translated into our representation, but potentially with a loss of information, with the consequence that a round trip may yield a result that is not semantically equivalent to the original. We differ from MAF in the following ways:

1. No vAlt construction; if there is ambiguity about the value of a feature, express it by creating two different WordForms, one with each possible value.
2. The unit called "token" in MAF is not represented explicitly. To translate into MAF one can create one token per ground wordForm. To translate a MAF wordForm with multiple tokens into our representation, create one MorphUnit per token; this is a loss of information in the sense that translating back to MAF will result in a representation semantically different from the original.
3. MAF allows start and end offsets to be in arbitrary formats (e.g. timecodes for video), whereas we assume the input is a string and therefore allow only character offsets.
4. MAF allows null wordForms, i.e. wordForms that contain no tokens, to cover phenomena like PRO-drop. We do not represent null word forms.
5. In MAF all tokens (and thus all non-null wordForms) are required to have start and end offsets. In our representation, sub-word wordForms (typically the morphemes of a compound word) may have null start and/or end, as long as the first one in a word has a start, and the last one has an end.
6. MAF allows a wordForm to cover multiple non-contiguous spans of text, while our representation does not.
7. MAF allows both in-line and standoff (offset-based) annotation, whereas we allow only standoff.

If the list returned by getExceptions() is non-empty, then one or more errors happened during processing. Partial results may still be available despite errors, but they should be consumed with caution.

---

| Nested Class Summary | | Page |
|---|---|---|
| static class | MorphoResult.ComposedWordForm | 45 |
| static class | MorphoResult.Disjunction<br>Consistency constraints that implementations must satisfy:<br>•All alternatives of a Disjunction should have the same start and end. | 47 |
| static class | MorphoResult.GroundWordForm | 50 |
| static class | MorphoResult.Sequence | 52 |
| abstract static class | MorphoResult.SequenceElement | 54 |

| | | |
|---|---|---|
| abstract static class | **MorphoResult.WordForm**<br>Represents a word, a multi-word term, or a sub-word unit of a compound word. | 56 |
| static enum | **MorphoResult.WordFormType** | 59 |

| Constructor Summary | Page |
|---|---|
| **MorphoResult**(MorphoResult.Sequence sequence, List<? extends Exception> exceptions) | 44 |

| Method Summary | | Page |
|---|---|---|
| List<? extends Exception> | **getExceptions**() | 44 |
| MorphoResult.Sequence | **getSequence**() | 44 |

# Constructor Detail

## MorphoResult

```
public MorphoResult(MorphoResult.Sequence sequence,
                    List<? extends Exception> exceptions)
```

# Method Detail

## getSequence

```
public MorphoResult.Sequence getSequence()
```

---

## getExceptions

```
public List<? extends Exception> getExceptions()
```

# Class MorphoResult.ComposedWordForm

eu.europeana.linguistic.api

java.lang.Object
  └ eu.europeana.linguistic.api.MorphoResult.SequenceElement
    └ eu.europeana.linguistic.api.MorphoResult.WordForm
      └ **eu.europeana.linguistic.api.MorphoResult.ComposedWordForm**

**Enclosing class:**

MorphoResult

---

public static class **MorphoResult.ComposedWordForm**
extends MorphoResult.WordForm

---

| Fields inherited from class eu.europeana.linguistic.api.MorphoResult.WordForm |
|---|
| lemma, pos, surface, type |

| Constructor Summary | Page |
|---|---|
| **MorphoResult.ComposedWordForm**(List<? extends MorphoResult.WordForm> components, String surface, String lemma, Tagset.Tag pos, MorphoResult.WordFormType type) | 45 |

| Method Summary | | Page |
|---|---|---|
| boolean | **equals**(Object o) | 46 |
| List<? extends MorphoResult.WordForm> | **getComponents**()<br>Must not return null, nor an empty list. | 46 |
| int | **getEnd**() | 46 |
| int | **getStart**() | 46 |
| int | **hashCode**() | 46 |

| Methods inherited from class eu.europeana.linguistic.api.MorphoResult.WordForm |
|---|
| getLemma, getPos, getSurface, getType, prettyPrint |

## Constructor Detail

### MorphoResult.ComposedWordForm

```
public MorphoResult.ComposedWordForm(List<? extends MorphoResult.WordForm> components,
                                     String surface,
                                     String lemma,
                                     Tagset.Tag pos,
                                     MorphoResult.WordFormType type)
```

# Method Detail

## getComponents

```
public List<? extends MorphoResult.WordForm> getComponents()
```

Must not return null, nor an empty list.

**Overrides:**

getComponents in class MorphoResult.WordForm

---

## getStart

```
public int getStart()
```

**Overrides:**

getStart in class MorphoResult.SequenceElement

---

## getEnd

```
public int getEnd()
```

**Overrides:**

getEnd in class MorphoResult.SequenceElement

---

## hashCode

```
public int hashCode()
```

**Overrides:**

hashCode in class MorphoResult.SequenceElement

---

## equals

```
public boolean equals(Object o)
```

**Overrides:**

equals in class MorphoResult.SequenceElement

# Class MorphoResult.Disjunction

eu.europeana.linguistic.api

java.lang.Object
  └ eu.europeana.linguistic.api.MorphoResult.SequenceElement
    └ **eu.europeana.linguistic.api.MorphoResult.Disjunction**

**Enclosing class:**

MorphoResult

---

public static class **MorphoResult.Disjunction**
extends MorphoResult.SequenceElement

Consistency constraints that implementations must satisfy:

1. All alternatives of a Disjunction should have the same start and end.
2. A Disjunction should have at least two alternatives (if not, it can be eliminated).

---

| Constructor Summary | Page |
|---|---|
| **MorphoResult.Disjunction**(List<MorphoResult.WordForm> words)<br>A convenience method for the common case where each Sequence in the Disjunction consists of just a single WordForm. | 47 |
| **MorphoResult.Disjunction**(Set<? extends MorphoResult.Sequence> alternatives) | 47 |

| Method Summary | | Page |
|---|---|---|
| boolean | **equals**(Object o) | 48 |
| Set<? extends MorphoResult.Sequence> | **getAlternatives**()<br>Never returns null | 48 |
| int | **getEnd**() | 48 |
| int | **getStart**() | 48 |
| int | **hashCode**() | 48 |
| String | **prettyPrint**(int level) | 48 |

## Constructor Detail

### MorphoResult.Disjunction

public **MorphoResult.Disjunction**(Set<? extends MorphoResult.Sequence> alternatives)

---

### MorphoResult.Disjunction

public **MorphoResult.Disjunction**(List<MorphoResult.WordForm> words)

A convenience method for the common case where each Sequence in the Disjunction consists of just a single WordForm.

## Method Detail

### getAlternatives

```
public Set<? extends MorphoResult.Sequence> getAlternatives()
```

Never returns null

---

### prettyPrint

```
public String prettyPrint(int level)
```

**Overrides:**

prettyPrint in class MorphoResult.SequenceElement

---

### hashCode

```
public int hashCode()
```

**Overrides:**

hashCode in class MorphoResult.SequenceElement

---

### equals

```
public boolean equals(Object o)
```

**Overrides:**

equals in class MorphoResult.SequenceElement

---

### getStart

```
public int getStart()
```

**Overrides:**

getStart in class MorphoResult.SequenceElement

---

### getEnd

```
public int getEnd()
```

**Overrides:**

`getEnd` in class `MorphoResult.SequenceElement`

# Class MorphoResult.GroundWordForm

eu.europeana.linguistic.api

java.lang.Object
  └ eu.europeana.linguistic.api.MorphoResult.SequenceElement
    └ eu.europeana.linguistic.api.MorphoResult.WordForm
      └ **eu.europeana.linguistic.api.MorphoResult.GroundWordForm**

**Enclosing class:**

MorphoResult

---

public static class **MorphoResult.GroundWordForm**
extends MorphoResult.WordForm

---

| Fields inherited from class eu.europeana.linguistic.api.MorphoResult.WordForm |
|---|
| lemma, pos, surface, type |

| Constructor Summary | Page |
|---|---|
| **MorphoResult.GroundWordForm**(int start, int end, String surface, String lemma, Tagset.Tag pos, MorphoResult.WordFormType type) | 50 |

| Method Summary | | Page |
|---|---|---|
| boolean | **equals**(Object o) | 51 |
| List<? extends MorphoResult.WordForm> | **getComponents**() | 51 |
| int | **getEnd**() | 51 |
| int | **getStart**() | 51 |
| int | **hashCode**() | 51 |

| Methods inherited from class eu.europeana.linguistic.api.MorphoResult.WordForm |
|---|
| getLemma, getPos, getSurface, getType, prettyPrint |

## Constructor Detail

### MorphoResult.GroundWordForm

```
public MorphoResult.GroundWordForm(int start,
                                   int end,
                                   String surface,
                                   String lemma,
                                   Tagset.Tag pos,
                                   MorphoResult.WordFormType type)
```

# Method Detail

## getStart

```
public int getStart()
```

**Overrides:**

getStart in class MorphoResult.SequenceElement

---

## getEnd

```
public int getEnd()
```

**Overrides:**

getEnd in class MorphoResult.SequenceElement

---

## hashCode

```
public int hashCode()
```

**Overrides:**

hashCode in class MorphoResult.SequenceElement

---

## equals

```
public boolean equals(Object o)
```

**Overrides:**

equals in class MorphoResult.SequenceElement

---

## getComponents

```
public List<? extends MorphoResult.WordForm> getComponents()
```

**Overrides:**

getComponents in class MorphoResult.WordForm

# Class MorphoResult.Sequence

eu.europeana.linguistic.api

java.lang.Object

└ **eu.europeana.linguistic.api.MorphoResult.Sequence**

**Enclosing class:**

MorphoResult

---

public static class **MorphoResult.Sequence**

extends Object

---

| Constructor Summary | Page |
|---|---|
| **MorphoResult.Sequence**(List<? extends MorphoResult.SequenceElement> elements) | 52 |

| Method Summary | | Page |
|---|---|---|
| boolean | **equals**(Object o) | 53 |
| List<? extends MorphoResult.SequenceElement> | **getElements**()<br>Never returns null | 53 |
| int | **getEnd**() | 52 |
| int | **getStart**() | 52 |
| int | **hashCode**() | 53 |
| String | **prettyPrint**(int level) | 53 |

## Constructor Detail

### MorphoResult.Sequence

public **MorphoResult.Sequence**(List<? extends MorphoResult.SequenceElement> elements)

## Method Detail

### getStart

public int **getStart**()

---

### getEnd

public int **getEnd**()

---

## getElements

```
public List<? extends MorphoResult.SequenceElement> getElements()
```

Never returns null

---

## prettyPrint

```
public String prettyPrint(int level)
```

---

## hashCode

```
public int hashCode()
```

**Overrides:**

`hashCode` in class `Object`

---

## equals

```
public boolean equals(Object o)
```

**Overrides:**

`equals` in class `Object`

# Class MorphoResult.SequenceElement

eu.europeana.linguistic.api

```
java.lang.Object
  └ eu.europeana.linguistic.api.MorphoResult.SequenceElement
```

**Direct Known Subclasses:**

MorphoResult.Disjunction, MorphoResult.WordForm

**Enclosing class:**

MorphoResult

---

```
abstract public static class MorphoResult.SequenceElement
extends Object
```

---

| Constructor Summary | Page |
|---|---|
| MorphoResult.SequenceElement() | 54 |

| Method Summary | | Page |
|---|---|---|
| abstract boolean | equals(Object o) | 55 |
| abstract int | getEnd() | 55 |
| abstract int | getStart() | 54 |
| abstract int | hashCode() | 55 |
| abstract String | prettyPrint(int level) | 54 |

## Constructor Detail

### MorphoResult.SequenceElement

```
public MorphoResult.SequenceElement()
```

## Method Detail

### prettyPrint

```
public abstract String prettyPrint(int level)
```

---

### getStart

```
public abstract int getStart()
```

---

## getEnd

```
public abstract int getEnd()
```

---

## hashCode

```
public abstract int hashCode()
```

**Overrides:**

`hashCode` in class `Object`

---

## equals

```
public abstract boolean equals(Object o)
```

**Overrides:**

`equals` in class `Object`

# Class MorphoResult.WordForm

eu.europeana.linguistic.api

java.lang.Object

└ eu.europeana.linguistic.api.MorphoResult.SequenceElement

└ **eu.europeana.linguistic.api.MorphoResult.WordForm**

**Direct Known Subclasses:**

MorphoResult.ComposedWordForm, MorphoResult.GroundWordForm

**Enclosing class:**

MorphoResult

---

abstract public static class **MorphoResult.WordForm**

extends MorphoResult.SequenceElement

Represents a word, a multi-word term, or a sub-word unit of a compound word. If getComponents() is empty, then this is a ground WordForm, which corresponds in MAF to a wordForm made up of one or more tokens. The components of a WordForm are not required to be contiguous.

TODO: the presence of getStart() and getEnd() methods on WordForm is incompatible with the possibility that a WordForm could contain non-contiguous components. Is there a fundamental inconsistency in MAF, or did I just map it to Java in an inconsistent way?

---

| Field Summary | | Page |
|---|---|---|
| protected String | **lemma** | 57 |
| protected Tagset.Tag | **pos** | 57 |
| protected String | **surface** | 57 |
| protected MorphoResult.WordFormType | **type** | 57 |

| Constructor Summary | | Page |
|---|---|---|
| protected | **MorphoResult.WordForm**(String surface, String lemma, Tagset.Tag pos, MorphoResult.WordFormType type) | 57 |

| Method Summary | | Page |
|---|---|---|
| abstract List<? extends MorphoResult.WordForm> | **getComponents**() | 58 |
| String | **getLemma**()<br>May return null | 57 |
| Tagset.Tag | **getPos**()<br>May return null | 57 |
| String | **getSurface**()<br>May return null | 58 |

| | | |
|---|---|---|
| MorphoResult.WordFormType | **getType**()<br>TODO: do we need this? | *58* |
| String | **prettyPrint**(int level) | *58* |

| Methods inherited from class eu.europeana.linguistic.api.MorphoResult.SequenceElement |
|---|
| equals, getEnd, getStart, hashCode |

# Field Detail

## surface

protected String **surface**

---

## lemma

protected String **lemma**

---

## pos

protected Tagset.Tag **pos**

---

## type

protected MorphoResult.WordFormType **type**

# Constructor Detail

## MorphoResult.WordForm

protected **MorphoResult.WordForm**(String surface,
String lemma,
Tagset.Tag pos,
MorphoResult.WordFormType type)

# Method Detail

## getLemma

public String **getLemma**()

May return null

---

## getPos

public Tagset.Tag **getPos**()

May return null

---

### getSurface

```
public String getSurface()
```

May return null

---

### getType

```
public MorphoResult.WordFormType getType()
```

TODO: do we need this?

---

### prettyPrint

```
public String prettyPrint(int level)
```

**Overrides:**

prettyPrint in class MorphoResult.SequenceElement

---

### getComponents

```
public abstract List<? extends MorphoResult.WordForm> getComponents()
```

# Enum MorphoResult.WordFormType

eu.europeana.linguistic.api

java.lang.Object
  └ java.lang.Enum<MorphoResult.WordFormType>
    └ **eu.europeana.linguistic.api.MorphoResult.WordFormType**

**All Implemented Interfaces:**

Comparable<MorphoResult.WordFormType>, Serializable

**Enclosing class:**

MorphoResult

---

public static enum **MorphoResult.WordFormType**
extends Enum<MorphoResult.WordFormType>

---

| Enum Constant Summary | Page |
|---|---|
| **MULTIWORD** | 59 |
| **SUBWORD** | 59 |
| **WORD** | 59 |

| Method Summary | | Page |
|---|---|---|
| static MorphoResult.WordFormType | **valueOf**(String name) | 60 |
| static MorphoResult.WordFormType[] | **values**() | 60 |

## Enum Constant Detail

### MULTIWORD

public static final MorphoResult.WordFormType **MULTIWORD**

---

### WORD

public static final MorphoResult.WordFormType **WORD**

---

### SUBWORD

public static final MorphoResult.WordFormType **SUBWORD**

# Method Detail

## values

```
public static MorphoResult.WordFormType[] values()
```

---

## valueOf

```
public static MorphoResult.WordFormType valueOf(String name)
```

# Interface NamedEntityRecognizer

eu.europeana.linguistic.api

public interface **NamedEntityRecognizer**

A service that performs named entity extraction from text.

| Nested Class Summary | | Page |
|---|---|---|
| static class | NamedEntityRecognizer.Result | 62 |

| Method Summary | | Page |
|---|---|---|
| List<Language> | getSupportedLanguages() | 61 |
| Tagset | getTagset(Language lang)<br>Returns the tagset used for language lang, or null if that language is not supported. | 61 |
| NamedEntityRecognizer.Result | process(String text, Language lang)<br>Identifies named entity expressions in a text. | 61 |

## Method Detail

### getSupportedLanguages

List<Language> **getSupportedLanguages**()

### getTagset

Tagset **getTagset**(Language lang)

Returns the tagset used for language lang, or null if that language is not supported.

### process

NamedEntityRecognizer.Result **process**(String text,
Language lang)
throws LinguisticProcessingException

Identifies named entity expressions in a text. The return value contains a list of Annotation and a list of Exception. If the list of Exception is non-empty, then one or more errors happened during processing. Partial results might still be available, but they should be consumed with caution.

**Throws:**

LinguisticProcessingException

# Class NamedEntityRecognizer.Result

eu.europeana.linguistic.api

java.lang.Object

└ **eu.europeana.linguistic.api.NamedEntityRecognizer.Result**

**Enclosing class:**

NamedEntityRecognizer

---

public static class **NamedEntityRecognizer.Result**

extends Object

---

| Field Summary | | Page |
|---|---|---|
| protected List<? extends Exception> | **exceptions** | 62 |

| Constructor Summary | Page |
|---|---|
| **NamedEntityRecognizer.Result**(List<? extends Annotation> annotations, List<? extends Exception> exceptions) | 62 |

| Method Summary | | Page |
|---|---|---|
| List<? extends Annotation> | **getAnnotations**() | 62 |
| List<? extends Exception> | **getExceptions**() | 63 |

## Field Detail

### exceptions

protected List<? extends Exception> **exceptions**

## Constructor Detail

### NamedEntityRecognizer.Result

public **NamedEntityRecognizer.Result**(List<? extends Annotation> annotations,
List<? extends Exception> exceptions)

## Method Detail

### getAnnotations

public List<? extends Annotation> **getAnnotations**()

---

## getExceptions

```
public List<? extends Exception> getExceptions()
```

# Class NoSuchResourceException

eu.europeana.linguistic.api

java.lang.Object
  └ java.lang.Throwable
    └ java.lang.Exception
      └ **eu.europeana.linguistic.api.NoSuchResourceException**

**All Implemented Interfaces:**

Serializable

---

public class **NoSuchResourceException**
extends Exception

---

| Constructor Summary | Page |
|---|---|
| **NoSuchResourceException**() | 64 |
| **NoSuchResourceException**(String message) | 64 |
| **NoSuchResourceException**(String message, Throwable cause) | 64 |
| **NoSuchResourceException**(Throwable cause) | 64 |

## Constructor Detail

### NoSuchResourceException

public **NoSuchResourceException**()

---

### NoSuchResourceException

public **NoSuchResourceException**(String message)

---

### NoSuchResourceException

public **NoSuchResourceException**(Throwable cause)

---

### NoSuchResourceException

public **NoSuchResourceException**(String message,
Throwable cause)

# Class Tagset

eu.europeana.linguistic.api

```
java.lang.Object
  └eu.europeana.linguistic.api.Tagset
```

---

```
public class Tagset
extends Object
```

A Tagset is a fixed set of labels, such as parts of speech or named entity classes. Every Tagset and Tag has a URI, and the URI of a Tag consists of the URI of its Tagset plus a fragment part.

When modifying a tagset, be sure to give the new version a new URI.

---

| Nested Class Summary | | Page |
|---|---|---|
| static class | Tagset.Tag | 68 |

| Constructor Summary | Page |
|---|---|
| Tagset(eu.europeana.linguistic.api.TagsetBean tagsetBean) | 66 |
| Tagset(URI uri, String[] vendorStrings) | 65 |

| Method Summary | | Page |
|---|---|---|
| boolean | equals(Object o) | 67 |
| static Tagset | fromResource(String resource, Class<?> clazz)<br>Creates a Tagset from an XML serialization. | 66 |
| Tagset.Tag | getTag(String vendorString) | 67 |
| URI | getUri() | 66 |
| Map<String,Tagset.Tag> | tagMap() | 66 |
| List<Tagset.Tag> | tags() | 66 |
| static void | validate(URI uri)<br>Checks that the URI is a legal tagset URI. | 66 |

## Constructor Detail

### Tagset

```
public Tagset(URI uri,
              String[] vendorStrings)
       throws URISyntaxException
```

---

### Tagset

```
public Tagset(eu.europeana.linguistic.api.TagsetBean tagsetBean)
       throws URISyntaxException
```

## Method Detail

### getUri

```
public URI getUri()
```

---

### tags

```
public List<Tagset.Tag> tags()
```

---

### tagMap

```
public Map<String,Tagset.Tag> tagMap()
```

---

### fromResource

```
public static Tagset fromResource(String resource,
                                  Class<?> clazz)
                           throws JAXBException,
                                  URISyntaxException
```

Creates a Tagset from an XML serialization. The schema is generated automatically by JAXB. clazz is the class whose classloader will be used to find the resource (relevant when the resource path is relative). TODO: instructions for finding schema

**Throws:**

JAXBException

URISyntaxException

---

### validate

```
public static void validate(URI uri)
                     throws URISyntaxException
```

Checks that the URI is a legal tagset URI. Does nothing if valid, throws InitializationException with an explanatory message if invalid.

**Throws:**

URISyntaxException

---

## equals

```
public boolean equals(Object o)
```

**Overrides:**

`equals` in class `Object`

---

## getTag

```
public Tagset.Tag getTag(String vendorString)
```

# Class Tagset.Tag

eu.europeana.linguistic.api

java.lang.Object

└ **eu.europeana.linguistic.api.Tagset.Tag**

**Enclosing class:**

Tagset

---

public static class **Tagset.Tag**

extends Object

---

| Constructor Summary | Page |
|---|---|
| **Tagset.Tag**(URI tagsetUri, String vendorString) | 68 |

| Method Summary | | Page |
|---|---|---|
| boolean | **equals**(Object o) | 69 |
| URI | **getUri**() | 68 |
| int | **hashCode**() | 68 |

## Constructor Detail

### Tagset.Tag

```
public Tagset.Tag(URI tagsetUri,
                  String vendorString)
           throws URISyntaxException
```

## Method Detail

### getUri

```
public URI getUri()
```

---

### hashCode

```
public int hashCode()
```

**Overrides:**

hashCode in class Object

---

## equals

```
public boolean equals(Object o)
```

**Overrides:**

`equals` in class `Object`

# Class TagsetTypeAdapter

eu.europeana.linguistic.api

java.lang.Object

└ javax.xml.bind.annotation.adapters.XmlAdapter<eu.europeana.linguistic.api.TagsetBean,Tagset>

└ **eu.europeana.linguistic.api.TagsetTypeAdapter**

---

public class **TagsetTypeAdapter**

extends XmlAdapter<eu.europeana.linguistic.api.TagsetBean,Tagset>

---

| Constructor Summary | Page |
|---|---|
| **TagsetTypeAdapter**() | 70 |

| Method Summary | | Page |
|---|---|---|
| eu.europeana.linguistic.api.TagsetBean | **marshal**(Tagset tagset) | 70 |
| Tagset | **unmarshal**(eu.europeana.linguistic.api.TagsetBean bean) | 70 |

## Constructor Detail

### TagsetTypeAdapter

public **TagsetTypeAdapter**()

## Method Detail

### marshal

public eu.europeana.linguistic.api.TagsetBean **marshal**(Tagset tagset)
throws Exception

**Overrides:**

marshal in class XmlAdapter

**Throws:**

Exception

---

### unmarshal

public Tagset **unmarshal**(eu.europeana.linguistic.api.TagsetBean bean)
throws Exception

**Overrides:**

unmarshal in class XmlAdapter

**Throws:**

`Exception`

# Interface TranslationDictionary

eu.europeana.linguistic.api

public interface **TranslationDictionary**

Provides lookup in one or more bilingual dictionaries. Lookup methods accept a sequence of words rather than a single word. This way, the dictionary can return translations for multi-word terms as well as for individual words.

This interface is modeled after the Language Grid service AbstractBilingualDictionaryWithLongestMatchSearch, but returns all matches rather than only the longest match, in the interest of a cleaner separation between dictionary lookup and the translation algorithm.

TODO: Javadoc comments for methods don't adequately explain behavior when one of the input lemmas or Words is a multi-word term, i.e. a string containing spaces.

| Method Summary | | Page |
|---|---|---|
| Tagset | **getTagset**()<br>Returns the tagset used by the service | 73 |
| boolean | **supportsLanguagePair**(Language sourceLang, Language targetLang)<br>Returns true iff this dictionary is able to translate from sourceLang to targetLang. | 72 |
| List<TranslationWithPosition> | **translateLemmaSequence**(List<String> lemmas, Language sourceLang, Language targetLang)<br>Returns translations into targetLang for each subsequence of lemmas that matches a term of sourceLang. | 73 |
| List<TranslationWithPosition> | **translatePrefixSequence**(List<String> prefixes, Language sourceLang, Language targetLang)<br>Returns translations into targetLang for each subsequence of prefixes that matches a term of sourceLang. | 73 |
| List<TranslationWithPosition> | **translateWordSequence**(List<Word> words, Language sourceLang, Language targetLang)<br>Returns translations into targetLang for each subsequence of words that matches a term of sourceLang. | 73 |

## Method Detail

### supportsLanguagePair

boolean **supportsLanguagePair**(Language sourceLang,
Language targetLang)

Returns true iff this dictionary is able to translate from sourceLang to targetLang.

## translateLemmaSequence

```
List<TranslationWithPosition> translateLemmaSequence(
            List<String> lemmas,

            Language sourceLang,

            Language targetLang)
```

Returns translations into `targetLang` for each subsequence of `lemmas` that matches a term of `sourceLang`. A sequence of lemmas $l_1$, $l_2$, ..., $l_n$ matches a source language term $s_1$, $s_2$, ..., $s_n$ iff $l_i$.equals($s_i$) for each i from 1 to n. For example, if translating from French to English the list {"pomme", "de", "terre"} might return the translations "apple", "of", "earth", "soil", "potato".

---

## translatePrefixSequence

```
List<TranslationWithPosition> translatePrefixSequence(
              List<String> prefixes,

              Language sourceLang,

              Language targetLang)
```

Returns translations into `targetLang` for each subsequence of `prefixes` that matches a term of `sourceLang`. A sequence of prefixes $p_1$, $p_2$, ...,$p_n$ matches a source language term $s_1$, $s_2$, ..., $s_n$ iff $s_i$.startsWith($p_i$) for each i from 1 to n. For example, if translating from French to English the list {"pom", "de", "ter"} might return the translations "apple", "of", "earth", "soil", "potato".

---

## translateWordSequence

```
List<TranslationWithPosition> translateWordSequence(
          List<Word> words,

          Language sourceLang,

          Language targetLang)
```

Returns translations into `targetLang` for each subsequence of `words` that matches a term of `sourceLang`.

A sequence of `Word` objects $w_1$, $w_2$, ..., $w_n$ matches a source language term $s_1$, $s_2$, ..., $s_n$ iff for each i from 1 to n, either $w_i$.getSurface().equals($s_i$) or $w_i$.getLemma().equals(si).

---

## getTagset

```
Tagset getTagset()
```

Returns the tagset used by the service

# Class TranslationWithPosition

eu.europeana.linguistic.api

java.lang.Object

└ **eu.europeana.linguistic.api.TranslationWithPosition**

---

public class **TranslationWithPosition**

extends Object

This is part of the TranslationDictionary API. It was originally an inner class of TranslationDictionary, but was moved to a top- level class to work around a bug in JAX-WS.

---

| Constructor Summary | Page |
|---|---|
| **TranslationWithPosition**() | 75 |
| **TranslationWithPosition**(String sourceString, String targetString, Tagset.Tag pos, int startIndex, int numberOfWords, boolean isNamedEntity) | 75 |

| Method Summary | | Page |
|---|---|---|
| int | **getNumberOfWords**()<br>The number of words of the input sequence covered by the translation. | 75 |
| Tagset.Tag | **getPos**() | 75 |
| String | **getSourceString**()<br>The source language (headword) string. | 75 |
| int | **getStartIndex**()<br>The index of the first word in the input sequence covered by the translation. | 75 |
| String | **getTargetString**()<br>The target language string. | 75 |
| boolean | **isNamedEntity**() | 76 |
| void | **setNamedEntity**(boolean isNamedEntity) | 76 |
| void | **setNumberOfWords**(int numberOfWords) | 76 |
| void | **setPos**(Tagset.Tag pos) | 76 |
| void | **setSourceString**(String sourceString) | 76 |
| void | **setStartIndex**(int startIndex) | 76 |
| void | **setTargetString**(String targetString) | 76 |
| String | **toString**() | 76 |

# Constructor Detail

## TranslationWithPosition

```
public TranslationWithPosition(String sourceString,
                               String targetString,
                               Tagset.Tag pos,
                               int startIndex,
                               int numberOfWords,
                               boolean isNamedEntity)
```

---

## TranslationWithPosition

```
public TranslationWithPosition()
```

# Method Detail

## getSourceString

```
public String getSourceString()
```

The source language (headword) string. May contain multiple words separated by spaces.

---

## getTargetString

```
public String getTargetString()
```

The target language string. May contain multiple words separated by spaces.

---

## getPos

```
public Tagset.Tag getPos()
```

---

## getStartIndex

```
public int getStartIndex()
```

The index of the first word in the input sequence covered by the translation. (Counts from zero.)

---

## getNumberOfWords

```
public int getNumberOfWords()
```

The number of words of the input sequence covered by the translation.

---

### isNamedEntity

```
public boolean isNamedEntity()
```

---

### toString

```
public String toString()
```

**Overrides:**

`toString` in class `Object`

---

### setSourceString

```
public void setSourceString(String sourceString)
```

---

### setTargetString

```
public void setTargetString(String targetString)
```

---

### setPos

```
public void setPos(Tagset.Tag pos)
```

---

### setStartIndex

```
public void setStartIndex(int startIndex)
```

---

### setNumberOfWords

```
public void setNumberOfWords(int numberOfWords)
```

---

### setNamedEntity

```
public void setNamedEntity(boolean isNamedEntity)
```

# Class UnsupportedLanguageException

eu.europeana.linguistic.api

```
java.lang.Object
  └ java.lang.Throwable
      └ java.lang.Exception
          └ java.lang.RuntimeException
              └ eu.europeana.linguistic.api.UnsupportedLanguageException
```

**All Implemented Interfaces:**

Serializable

---

```
public class UnsupportedLanguageException
extends RuntimeException
```

---

| Constructor Summary | Page |
|---|---|
| UnsupportedLanguageException() | 77 |
| UnsupportedLanguageException(Language lang) | 78 |
| UnsupportedLanguageException(Language l1, Language l2) | 78 |
| UnsupportedLanguageException(String message) | 77 |
| UnsupportedLanguageException(String message, Throwable cause) | 77 |
| UnsupportedLanguageException(Throwable cause) | 77 |

## Constructor Detail

### UnsupportedLanguageException

```
public UnsupportedLanguageException()
```

---

### UnsupportedLanguageException

```
public UnsupportedLanguageException(String message)
```

---

### UnsupportedLanguageException

```
public UnsupportedLanguageException(Throwable cause)
```

---

### UnsupportedLanguageException

```
public UnsupportedLanguageException(String message,
                                    Throwable cause)
```

## UnsupportedLanguageException

```
public UnsupportedLanguageException(Language lang)
```

## UnsupportedLanguageException

```
public UnsupportedLanguageException(Language l1,
                                    Language l2)
```

# Class Word

eu.europeana.linguistic.api

java.lang.Object

└ **eu.europeana.linguistic.api.Word**

---

public class **Word**

extends Object

This is part of the TranslationDictionary API. It was originally an inner class of TranslationDictionary, but was moved to a top- level class to work around a bug in JAX-WS.

---

| Constructor Summary | Page |
|---|---|
| **Word**() | 79 |
| **Word**(String surface, String lemma) | 79 |

| Method Summary | | Page |
|---|---|---|
| String | **getLemma**() | 80 |
| String | **getSurface**() | 79 |
| void | **setLemma**(String lemma) | 80 |
| void | **setSurface**(String surface) | 79 |

## Constructor Detail

### Word

public **Word**(String surface,
String lemma)

---

### Word

public **Word**()

## Method Detail

### getSurface

public String **getSurface**()

---

### setSurface

public void **setSurface**(String surface)

## getLemma

```
public String getLemma()
```

## setLemma

```
public void setLemma(String lemma)
```