

---

# Django Data Importer Documentation

*Release 2.2.1*

**Valder Gallo**

August 19, 2015



|          |                                   |           |
|----------|-----------------------------------|-----------|
| <b>1</b> | <b>Django Data Importer</b>       | <b>3</b>  |
| 1.1      | Documentation and usage . . . . . | 3         |
| 1.2      | Installation . . . . .            | 3         |
| 1.3      | Settings . . . . .                | 3         |
| 1.4      | Basic example . . . . .           | 4         |
| 1.5      | Django Model . . . . .            | 4         |
| 1.6      | Django XML . . . . .              | 5         |
| 1.7      | Django XLS/XLSX . . . . .         | 5         |
| 1.8      | Descriptor . . . . .              | 6         |
| 1.9      | TEST . . . . .                    | 7         |
| <b>2</b> | <b>Core</b>                       | <b>9</b>  |
| 2.1      | Base . . . . .                    | 9         |
| 2.2      | Default Settings . . . . .        | 9         |
| 2.3      | Exceptions . . . . .              | 9         |
| 2.4      | Descriptors . . . . .             | 10        |
| <b>3</b> | <b>Importers</b>                  | <b>11</b> |
| 3.1      | XLSImporter . . . . .             | 11        |
| 3.2      | XLSImporter . . . . .             | 12        |
| 3.3      | XMLImporter . . . . .             | 13        |
| 3.4      | GenericImporter . . . . .         | 14        |
| <b>4</b> | <b>Data Importer Forms</b>        | <b>17</b> |
| 4.1      | FileUploadForm . . . . .          | 17        |
| <b>5</b> | <b>Data Importer Models</b>       | <b>19</b> |
| 5.1      | FileHistory . . . . .             | 19        |
| <b>6</b> | <b>Views</b>                      | <b>21</b> |
| 6.1      | DataImporterForm . . . . .        | 21        |
| 6.2      | Usage example . . . . .           | 21        |
| <b>7</b> | <b>Indices and tables</b>         | <b>23</b> |
|          | <b>Python Module Index</b>        | <b>25</b> |



Data importer documentation help

Contents:

- [Create Issues](#)
- [Report Bugs](#)
- [GitHub Project](#)



---

## Django Data Importer

---

**Django Data Importer** is a tool which allow you to transform easily a CSV, XML, XLS and XLSX file into a python object or a django model instance. It is based on the django-style declarative model.

### 1.1 Documentation and usage

Read docs online in Read the Docs:

<https://django-data-importer.readthedocs.org/>

You can generate everything at the above site in your local folder by:

```
$ cd doc
$ make html
$ open _build/html/index.html # Or your preferred web browser
```

### 1.2 Installation

Use either `easy_install`:

```
easy_install data-importer
```

or `pip`:

```
pip install data-importer
```

### 1.3 Settings

Customize `data_importer` decoders

**DATA\_IMPORTER\_EXCEL\_DECODER** Default value is `cp1252`

**DATA\_IMPORTER\_DECODER** Default value is `UTF-8`

## 1.4 Basic example

Consider the following:

```
>>> from data_importer.importers import CSVImporter
>>> class MyCSVImporterModel(CSVImporter):
...     fields = ['name', 'age', 'length']
...     class Meta:
...         delimiter = ";"
```

You declare a `MyCSVImporterModel` which will match to a CSV file like this:

```
Anthony;27;1.75
```

To import the file or any iterable object, just do:

```
>>> my_csv_list = MyCSVImporterModel(source="my_csv_file_name.csv")
>>> row, first_line = my_csv_list.cleaned_data[0]
>>> first_line['age']
27
```

Without an explicit declaration, data and columns are matched in the same order:

```
Anthony --> Column 0 --> Field 0 --> name
27        --> Column 1 --> Field 1 --> age
1.75      --> Column 2 --> Field 2 --> length
```

## 1.5 Django Model

If you now want to interact with a django model, you just have to add a **Meta.model** option to the class meta.

```
>>> from django.db import models
>>> class MyModel(models.Model):
...     name = models.CharField(max_length=150)
...     age = models.CharField(max_length=150)
...     length = models.CharField(max_length=150)

>>> from data_importer.importers import CSVImporter
>>> from data_importer.model import MyModel
>>> class MyCSVImporterModel(CSVImporter):
...     class Meta:
...         delimiter = ";"
...         model = MyModel
```

That will automatically match to the following django model.

*The django model should be imported in the model*

**class Meta**

**delimiter** define the delimiter of the csv file. If you do not set one, the sniffer will try yo find one itself.

**ignore\_first\_line** Skip the first line if True.

**model** If defined, the importer will create an instance of this model.

**raise\_errors** If set to True, an error in a imported line will stop the loading.

**exclude** Exclude fields from list fields to import



**transaction** (*beta*) *not tested* Use transaction to save objects

**ignore\_empty\_lines** Not validate empty lines

## 1.6 Django XML

If you now want to interact with a django model, you just have to add a **Meta.model** option to the class meta.

XML file example:

```
<encspot>
  <file>
    <Name>Rocky Balboa</Name>
    <Age>40</Age>
    <Height>1.77</Height>
  </file>
  <file>
    <Name>Chuck Norris</Name>
    <Age>73</Age>
    <Height>1.78</Height>
  </file>
</encspot>
```

```
>>> from django.db import models
>>> class MyModel(models.Model):
...     name = models.CharField(max_length=150)
...     age = models.CharField(max_length=150)
...     height = models.CharField(max_length=150)
```

```
>>> from data_importer.importers import XMLImporter
>>> from data_importer.model import MyModel
>>> class MyCSVImporterModel(XMLImporter):
...     root = 'file'
...     class Meta:
...         model = MyModel
```

That will automatically match to the following django model.

*The django model should be imported in the model*

**class Meta**

**model** If defined, the importer will create an instance of this model.

**raise\_errors** If set to True, an error in a imported line will stop the loading.

**exclude** Exclude fields from list fields to import

**transaction** (*beta*) *not tested* Use transaction to save objects

## 1.7 Django XLS/XLSX

My XLS/XLSX file can be imported too

| Header1 | Header2 | Header3 | Header4 |
|---------|---------|---------|---------|
| Teste 1 | Teste 2 | Teste 3 | Teste 4 |
| Teste 1 | Teste 2 | Teste 3 | Teste 4 |

This is my model

```
>>> from django.db import models
>>> class MyModel(models.Model):
...     header1 = models.CharField(max_length=150)
...     header2 = models.CharField(max_length=150)
...     header3 = models.CharField(max_length=150)
...     header4 = models.CharField(max_length=150)
```

This is my class

```
>>> from data_importer.importers import XLSImporter
>>> from data_importer.model import MyModel
>>> class MyXLSImporterModel(XLSImporter):
...     class Meta:
...         model = MyModel
```

If you are using XLSX you will need use XLSXImporter to made same importer

```
>>> from data_importer.importers import XLSXImporter
>>> from data_importer.model import MyModel
>>> class MyXLSXImporterModel(XLSXImporter):
...     class Meta:
...         model = MyModel
```

**class Meta**

**ignore\_first\_line** Skip the first line if True.

**model** If defined, the importer will create an instance of this model.

**raise\_errors** If set to True, an error in a imported line will stop the loading.

**exclude** Exclude fields from list fields to import

**transaction (beta) not tested** Use transaction to save objects

## 1.8 Descriptor

Using file descriptor to define fields for large models.

import\_test.json

```
{
  'app_name': 'mytest.Contact',
  {
    // field name / name on import file or key index
    'name': 'My Name',
    'year': 'My Year',
    'last': 3
  }
}
```

model.py

```
class Contact(models.Model):
    name = models.CharField(max_length=50)
    year = models.CharField(max_length=10)
    laster = models.CharField(max_length=5)
    phone = models.CharField(max_length=5)
```

```
address = models.CharField(max_length=5)
state = models.CharField(max_length=5)
```

importer.py

```
class MyImporter(BaseImporter):
    class Meta:
        config_file = 'import_test.json'
        model = Contact
        delimiter = ','
        ignore_first_line = True
```

content\_file.csv

```
name,year,last
Test,12,1
Test2,13,2
Test3,14,3
```

## 1.9 TEST

|                       |                |    |
|-----------------------|----------------|----|
| Acentuation with XLS  | Excel MAC 2011 | OK |
| Acentuation with XLS  | Excel WIN 2010 | OK |
| Acentuation with XLSX | Excel MAC 2011 | OK |
| Acentuation with XLSX | Excel WIN 2010 | OK |
| Acentuation with CSV  | Excel Win 2010 | OK |

---

**Python** python 2.7

**Django** 1.6; 1.7



## 2.1 Base

`data_importer.core.base.objclass2dict (objclass)`

Meta is a objclass on python 2.7 and no have `__dict__` attribute.

This method convert one objclass to one lazy dict without `AttributeError`

## 2.2 Default Settings

Customize `data_importer` decoders

**DATA\_IMPORTER\_EXCEL\_DECODER** Default value is `cp1252`

**DATA\_IMPORTER\_DECODER** Default value is `UTF-8`

**DATA\_IMPORTER\_TASK**

Need Celery installed to set importers as **Task** *default value is False*

**DATA\_IMPORTER\_QUEUE**

Set Celery Queue in **DataImpoter Tasks** *default value is DataImporter*

**DATA\_IMPORTER\_TASK\_LOCK\_EXPIRE**

Set task expires time *default value is 60 \* 20*

## 2.3 Exceptions

**exception** `data_importer.core.exceptions.InvalidDescriptor`

Invalid Descriptor File Descriptor must be one valid JSON

**exception** `data_importer.core.exceptions.InvalidModel`

Invalid model in descriptor

**exception** `data_importer.core.exceptions.StopImporter`

Stop iterator and raise error message

**exception** `data_importer.core.exceptions.UnsuportedFile`

Unsuported file type

## 2.4 Descriptors

```
class data_importer.core.descriptor.ReadDescriptor (file_name=None,  
                                                    model_name=None)
```

```
    get_fields()
```

```
        Get content
```

```
    get_model()
```

```
        Read model from JSON descriptor
```

```
    read_file()
```

```
        Read json file
```

---

## Importers

---

### 3.1 XLSImporter

```
class data_importer.importers.xls_importer.XLSImporter (source=None, *args,
                                                         **kwargs)

class Meta
    Importer configurations

XLSImporter.clean()
    Custom clean method

XLSImporter.clean_field (field_name, value)
    User default django field validators to clean content and run custom validates

XLSImporter.clean_row (row_values)
    Custom clean method for full row data

XLSImporter.cleaned_data
    Return tupla with data cleaned

XLSImporter.errors
    Show errors catch by clean methods

XLSImporter.exclude_fields()
    Exclude fields from Meta.exclude

XLSImporter.get_error_message (error, row=None, error_type=None)

XLSImporter.is_valid()
    Clear content and return False if have errors

XLSImporter.load_descriptor()
    Set fields from descriptor file

XLSImporter.meta
    Is same to use .Meta

XLSImporter.post_clean()
    Excuted after all clean method

XLSImporter.post_save_all_lines()
    End exection

XLSImporter.pre_clean()
    Executed before all clean methods Important: pre_clean dont have cleaned_data content
```

`XLSImporter.pre_commit()`  
Executed before commit multiple register

`XLSImporter.process_row(row, values)`  
Read clean functions from importer and return tupla with row number, field and value

`XLSImporter.save(instance=None)`  
Save all contents DONT override this method

`XLSImporter.set_reader()`  
[[1,2,3], [2,3,4]]

`XLSImporter.source`  
Return source opened

`XLSImporter.start_fields()`  
Initial function to find fields or headers values This values will be used to process clean and save method  
If this method not have fields and have Meta.model this method will use model fields to populate content without id

`XLSImporter.to_unicode(bytestr)`  
Receive string bytestr and try to return a utf-8 string.

## 3.2 XLSImporter

```
class data_importer.importers.xlsx_importer.XLSXImporter (source=None, *args,
                                                         **kwargs)
```

**class Meta**  
Importer configurations

`XLSXImporter.clean()`  
Custom clean method

`XLSXImporter.clean_field(field_name, value)`  
User default django field validators to clean content and run custom validates

`XLSXImporter.clean_row(row_values)`  
Custom clean method for full row data

`XLSXImporter.cleaned_data`  
Return tupla with data cleaned

`XLSXImporter.errors`  
Show errors catch by clean methods

`XLSXImporter.exclude_fields()`  
Exclude fields from Meta.exclude

`XLSXImporter.get_error_message(error, row=None, error_type=None)`

`XLSXImporter.is_valid()`  
Clear content and return False if have errors

`XLSXImporter.load_descriptor()`  
Set fields from descriptor file

`XLSXImporter.meta`  
Is same to use .Meta



`XLSXImporter.post_clean()`  
Excuted after all clean method

`XLSXImporter.post_save_all_lines()`  
End exection

`XLSXImporter.pre_clean()`  
Executed before all clean methods Important: pre\_clean dont have cleaned\_data content

`XLSXImporter.pre_commit()`  
Executed before commit multiple register

`XLSXImporter.process_row(row, values)`  
Read clean functions from importer and return tupla with row number, field and value

`XLSXImporter.save(instance=None)`  
Save all contents DONT override this method

`XLSXImporter.set_reader(use_iterators=True, data_only=True)`  
Read XLSX files

`XLSXImporter.source`  
Return source opened

`XLSXImporter.start_fields()`  
Initial function to find fields or headers values This values will be used to process clean and save method  
If this method not have fields and have Meta.model this method will use model fields to populate content without id

`XLSXImporter.to_unicode(bytestr)`  
Receive string bytestr and try to return a utf-8 string.

### 3.3 XMLImporter

```
class data_importer.importers.xml_importer.XMLImporter (source=None, *args,
                                                         **kwargs)
    Import XML files

    class Meta
        Importer configurations

    XMLImporter.clean()
        Custom clean method

    XMLImporter.clean_field(field_name, value)
        User default django field validators to clean content and run custom validates

    XMLImporter.clean_row(row_values)
        Custom clean method for full row data

    XMLImporter.cleaned_data
        Return tupla with data cleaned

    XMLImporter.errors
        Show errors catch by clean methods

    XMLImporter.exclude_fields()
        Exclude fields from Meta.exclude

    XMLImporter.get_error_message(error, row=None, error_type=None)
```

`XMLImporter.is_valid()`  
Clear content and return False if have errors

`XMLImporter.load_descriptor()`  
Set fields from descriptor file

`XMLImporter.meta`  
Is same to use `.Meta`

`XMLImporter.post_clean()`  
Excuted after all clean method

`XMLImporter.post_save_all_lines()`  
End exection

`XMLImporter.pre_clean()`  
Executed before all clean methods Important: `pre_clean` dont have `cleaned_data` content

`XMLImporter.pre_commit()`  
Executed before commit multiple register

`XMLImporter.process_row(row, values)`  
Read clean functions from importer and return tupla with row number, field and value

`XMLImporter.root = 'root'`

`XMLImporter.save(instance=None)`  
Save all contents DONT override this method

`XMLImporter.set_reader()`

`XMLImporter.source`  
Return source opened

`XMLImporter.start_fields()`  
Initial function to find fields or headers values This values will be used to process clean and save method  
If this method not have fields and have `Meta.model` this method will use model fields to populate content without id

`XMLImporter.to_unicode(bytestr)`  
Receive string bytestr and try to return a utf-8 string.

## 3.4 GenericImporter

`class data_importer.importers.generic.GenericImporter(source=None, *args, **kwargs)`  
An implementation of `BaseImporter` that sets the right reader by file extension. Probably the best choice for almost all implementation cases

`class Meta`  
Importer configurations

`GenericImporter.clean()`  
Custom clean method

`GenericImporter.clean_field(field_name, value)`  
User default django field validators to clean content and run custom validates

`GenericImporter.clean_row(row_values)`  
Custom clean method for full row data

`GenericImporter.cleaned_data`  
Return tupla with data cleaned

`GenericImporter.errors`  
Show errors catch by clean methods

`GenericImporter.exclude_fields()`  
Exclude fields from `Meta.exclude`

`GenericImporter.get_error_message(error, row=None, error_type=None)`

`GenericImporter.get_reader_class()`  
Gets the right file reader class by source file extension

`GenericImporter.get_source_file_extension()`  
Gets the source file extension. Used to choose the right reader

`GenericImporter.is_valid()`  
Clear content and return False if have errors

`GenericImporter.load_descriptor()`  
Set fields from descriptor file

`GenericImporter.meta`  
Is same to use `.Meta`

`GenericImporter.post_clean()`  
Excuted after all clean method

`GenericImporter.post_save_all_lines()`  
End exection

`GenericImporter.pre_clean()`  
Executed before all clean methods Important: `pre_clean` dont have `cleaned_data` content

`GenericImporter.pre_commit()`  
Executed before commit multiple register

`GenericImporter.process_row(row, values)`  
Read clean functions from importer and return tupla with row number, field and value

`GenericImporter.save(instance=None)`  
Save all contents DONT override this method

`GenericImporter.set_reader()`

`GenericImporter.source`  
Return source opened

`GenericImporter.start_fields()`  
Initial function to find fields or headers values This values will be used to process clean and save method  
If this method not have fields and have `Meta.model` this method will use model fields to populate content without id

`GenericImporter.to_unicode(bytestr)`  
Receive string bytestr and try to return a utf-8 string.



---

## Data Importer Forms

---

Is one simple `django.ModelForm` with content to upload content

**Parameters**

- content (*FileField*) File uploaded

### 4.1 FileUploadForm



---

## Data Importer Models

---

### 5.1 FileHistory





## 6.1 DataImporterForm

Is a mixin of `django.views.generic.edit.FormView` with default template and form to upload files and import content.

### Parameters

**model** Model where the file will be save

*By default this values is FileHistory*

**template\_name** Template name to be used with FormView

*By default is data\_importer/data\_importer.html*

**form\_class** Form that will be used to upload file

*By default this value is FileUploadForm*

**task** Task that will be used to parse file imported

*By default this value is DataImpoterTask*

**importer** Must be one `data_importer.importers` class that will be used to validate data.

**is\_task** Use importer in async mode.

**success\_url** Redirect to success page after importer

**extra\_context** Set extra context values in template

## 6.2 Usage example

```
class DataImporterCreateView(DataImporterForm):
    extra_context = {'title': 'Create Form Data Importer',
                    'template_file': 'myfile.csv'}
    importer = MyImporterModel
```



---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



## d

`data_importer.core.base`, [9](#)  
`data_importer.core.descriptor`, [10](#)  
`data_importer.core.exceptions`, [9](#)  
`data_importer.importers.generic`, [14](#)  
`data_importer.importers.xls_importer`,  
    [11](#)  
`data_importer.importers.xlsx_importer`,  
    [12](#)  
`data_importer.importers.xml_importer`,  
    [13](#)



**C**

`clean()` (`data_importer.importers.generic.GenericImporter` method), 14

`clean()` (`data_importer.importers.xls_importer.XLSImporter` method), 11

`clean()` (`data_importer.importers.xlsx_importer.XLSXImporter` method), 12

`clean()` (`data_importer.importers.xml_importer.XMLImporter` method), 13

`clean_field()` (`data_importer.importers.generic.GenericImporter` method), 14

`clean_field()` (`data_importer.importers.xls_importer.XLSImporter` method), 11

`clean_field()` (`data_importer.importers.xlsx_importer.XLSXImporter` method), 12

`clean_field()` (`data_importer.importers.xml_importer.XMLImporter` method), 13

`clean_row()` (`data_importer.importers.generic.GenericImporter` method), 14

`clean_row()` (`data_importer.importers.xls_importer.XLSImporter` method), 11

`clean_row()` (`data_importer.importers.xlsx_importer.XLSXImporter` method), 12

`clean_row()` (`data_importer.importers.xml_importer.XMLImporter` method), 13

`cleaned_data` (`data_importer.importers.generic.GenericImporter` attribute), 14

`cleaned_data` (`data_importer.importers.xls_importer.XLSImporter` attribute), 11

`cleaned_data` (`data_importer.importers.xlsx_importer.XLSXImporter` attribute), 12

`cleaned_data` (`data_importer.importers.xml_importer.XMLImporter` attribute), 13

**D**

`data_importer.core.base` (module), 9

`data_importer.core.descriptor` (module), 10

`data_importer.core.exceptions` (module), 9

`data_importer.importers.generic` (module), 14

`data_importer.importers.xls_importer` (module), 11

`data_importer.importers.xlsx_importer` (module), 12

`data_importer.importers.xml_importer` (module), 13

**E**

`errors` (`data_importer.importers.generic.GenericImporter` attribute), 15

`errors` (`data_importer.importers.xls_importer.XLSImporter` attribute), 11

`errors` (`data_importer.importers.xlsx_importer.XLSXImporter` attribute), 12

`errors` (`data_importer.importers.xml_importer.XMLImporter` attribute), 13

`exclude_fields()` (`data_importer.importers.generic.GenericImporter` method), 15

`exclude_fields()` (`data_importer.importers.xls_importer.XLSImporter` method), 11

`exclude_fields()` (`data_importer.importers.xlsx_importer.XLSXImporter` method), 12

`exclude_fields()` (`data_importer.importers.xml_importer.XMLImporter` method), 13

**G**

`GenericImporter` (class in `data_importer.importers.generic`), 14

`GenericImporter.Meta` (class in `data_importer.importers.generic`), 14

`get_error_message()` (`data_importer.importers.generic.GenericImporter` method), 15

`get_error_message()` (`data_importer.importers.xls_importer.XLSImporter` method), 11

`get_error_message()` (`data_importer.importers.xlsx_importer.XLSXImporter` method), 12

`get_error_message()` (`data_importer.importers.xml_importer.XMLImporter` method), 13

`get_fields()` (`data_importer.core.descriptor.ReadDescriptor` method), 10

`get_model()` (`data_importer.core.descriptor.ReadDescriptor` method), 10

`get_reader_class()` (`data_importer.importers.generic.GenericImporter` method), 15

`get_source_file_extension()`  
(`data_importer.importers.generic.GenericImporter`  
method), 15

**I**

`InvalidDescriptor`, 9

`InvalidModel`, 9

`is_valid()` (`data_importer.importers.generic.GenericImporter`  
method), 15

`is_valid()` (`data_importer.importers.xls_importer.XLSImporter`  
method), 11

`is_valid()` (`data_importer.importers.xlsx_importer.XLSXImporter`  
method), 12

`is_valid()` (`data_importer.importers.xml_importer.XMLImporter`  
method), 13

**L**

`load_descriptor()` (`data_importer.importers.generic.GenericImporter`  
method), 15

`load_descriptor()` (`data_importer.importers.xls_importer.XLSImporter`  
method), 11

`load_descriptor()` (`data_importer.importers.xlsx_importer.XLSXImporter`  
method), 12

`load_descriptor()` (`data_importer.importers.xml_importer.XMLImporter`  
method), 14

**M**

`Meta` (built-in class), 4–6

`meta` (`data_importer.importers.generic.GenericImporter`  
attribute), 15

`meta` (`data_importer.importers.xls_importer.XLSImporter`  
attribute), 11

`meta` (`data_importer.importers.xlsx_importer.XLSXImporter`  
attribute), 12

`meta` (`data_importer.importers.xml_importer.XMLImporter`  
attribute), 14

**O**

`objclass2dict()` (in module `data_importer.core.base`), 9

**P**

`post_clean()` (`data_importer.importers.generic.GenericImporter`  
method), 15

`post_clean()` (`data_importer.importers.xls_importer.XLSImporter`  
method), 11

`post_clean()` (`data_importer.importers.xlsx_importer.XLSXImporter`  
method), 12

`post_clean()` (`data_importer.importers.xml_importer.XMLImporter`  
method), 14

`post_save_all_lines()` (`data_importer.importers.generic.GenericImporter`  
method), 15

`post_save_all_lines()` (`data_importer.importers.xls_importer.XLSImporter`  
method), 11

`post_save_all_lines()` (`data_importer.importers.xlsx_importer.XLSXImporter`  
method), 13

`post_save_all_lines()` (`data_importer.importers.xml_importer.XMLImporter`  
method), 14

`pre_clean()` (`data_importer.importers.generic.GenericImporter`  
method), 15

`pre_clean()` (`data_importer.importers.xls_importer.XLSImporter`  
method), 11

`pre_clean()` (`data_importer.importers.xlsx_importer.XLSXImporter`  
method), 13

`pre_clean()` (`data_importer.importers.xml_importer.XMLImporter`  
method), 14

`pre_commit()` (`data_importer.importers.generic.GenericImporter`  
method), 15

`pre_commit()` (`data_importer.importers.xls_importer.XLSImporter`  
method), 11

`pre_commit()` (`data_importer.importers.xlsx_importer.XLSXImporter`  
method), 13

`pre_commit()` (`data_importer.importers.xml_importer.XMLImporter`  
method), 14

`process_row()` (`data_importer.importers.generic.GenericImporter`  
method), 15

`process_row()` (`data_importer.importers.xls_importer.XLSImporter`  
method), 12

`process_row()` (`data_importer.importers.xlsx_importer.XLSXImporter`  
method), 13

`process_row()` (`data_importer.importers.xml_importer.XMLImporter`  
method), 14

**R**

`read_file()` (`data_importer.core.descriptor.ReadDescriptor`  
method), 10

`ReadDescriptor` (class in `data_importer.core.descriptor`), 10

`root` (`data_importer.importers.xml_importer.XMLImporter`  
attribute), 14

**S**

`save()` (`data_importer.importers.generic.GenericImporter`  
method), 15

`save()` (`data_importer.importers.xls_importer.XLSImporter`  
method), 12

`save()` (`data_importer.importers.xlsx_importer.XLSXImporter`  
method), 13

`save()` (`data_importer.importers.xml_importer.XMLImporter`  
method), 14

`set_reader()` (`data_importer.importers.generic.GenericImporter`  
method), 15

`set_reader()` (`data_importer.importers.xls_importer.XLSImporter`  
method), 12

`set_reader()` (`data_importer.importers.xlsx_importer.XLSXImporter`  
method), 13

`set_reader()` (`data_importer.importers.xml_importer.XMLImporter`  
method), 14



`source` (`data_importer.importers.generic.GenericImporter`  
attribute), [15](#)  
`source` (`data_importer.importers.xls_importer.XLSImporter`  
attribute), [12](#)  
`source` (`data_importer.importers.xlsx_importer.XLSXImporter`  
attribute), [13](#)  
`source` (`data_importer.importers.xml_importer.XMLImporter`  
attribute), [14](#)  
`start_fields()` (`data_importer.importers.generic.GenericImporter`  
method), [15](#)  
`start_fields()` (`data_importer.importers.xls_importer.XLSImporter`  
method), [12](#)  
`start_fields()` (`data_importer.importers.xlsx_importer.XLSXImporter`  
method), [13](#)  
`start_fields()` (`data_importer.importers.xml_importer.XMLImporter`  
method), [14](#)  
`StopImporter`, [9](#)

## T

`to_unicode()` (`data_importer.importers.generic.GenericImporter`  
method), [15](#)  
`to_unicode()` (`data_importer.importers.xls_importer.XLSImporter`  
method), [12](#)  
`to_unicode()` (`data_importer.importers.xlsx_importer.XLSXImporter`  
method), [13](#)  
`to_unicode()` (`data_importer.importers.xml_importer.XMLImporter`  
method), [14](#)

## U

`UnsupportedFile`, [9](#)

## X

`XLSImporter` (class in  
`data_importer.importers.xls_importer`), [11](#)  
`XLSImporter.Meta` (class in  
`data_importer.importers.xls_importer`), [11](#)  
`XLSXImporter` (class in  
`data_importer.importers.xlsx_importer`),  
[12](#)  
`XLSXImporter.Meta` (class in  
`data_importer.importers.xlsx_importer`),  
[12](#)  
`XMLImporter` (class in  
`data_importer.importers.xml_importer`), [13](#)  
`XMLImporter.Meta` (class in  
`data_importer.importers.xml_importer`), [13](#)