API Documentation

API Documentation

December 12, 2013

Contents

1 Package project.easydata

```
Este modulo corresponde a la aplicacion EasyData para la publicacion controlada
de los datos de sus proyectos Django.
```

1.1 Modules

- **decorators**: En este modulo se encuentran definida el decorador encargado de comprobar en las vistas de la aplicacion que el usuario que intenta acceder a la vista, esta logueado y tiene los permisos de superusuario.
  (Section 2, p. 5)
  - **decorators**: En este modulo se encuentran definida el decorador encargado de comprobar en las vistas de la aplicacion que el usuario que intenta acceder a la vista, esta logueado y tiene los permisos de superusuario.
    (Section 3, p. 6)
- **forms**: En este modulo se almacenan tanto los formularios ordinarios como los basados...
  (Section 4, p. 7)
  - **forms**: En este modulo se almacenan tanto los formularios ordinarios de Django que se...
    (Section 5, p. 8)
    - **d2rqform**: Este modulo implementa el formulario para captar ficheros d2rq y modificarlos...
      (Section 6, p. 9)
  - **modelforms**: En este modulo se almacenan tanto los formularios basados en modelos que se...
    (Section 7, p. 10)
    - **editnamespace**: En este modulo se almacena el formulario que se utiliza para realizar la edicion de namespaces.
      (Section 8, p. 11)
    - **fieldpropiedad**: En este modulo se almacena el formulario que se utiliza para realizar el mapeo de los fields con las propiedades existentes.
      (Section 9, p. 12)
    - **modeloentidad**: En este modulo se almacena el formulario que se utiliza para realizar el mapeo de los modelos con las entidades existentes.
      (Section 10, p. 14)
    - **newnamespace**: En este modulo se almacena el formulario que se utiliza para realizar la creacion de nuevos namespaces.
      (Section 11, p. 16)
    - **visibilityfields**: En este modulo se almacena el formulario que se utiliza para realizar la configuracion de visibilidad de los fields.
      (Section 12, p. 17)
    - **visibilitymodels**: En este modulo se almacena el formulario que se utiliza para realizar la configuracion de visibilidad de los modelos.
      (Section 13, p. 18)
- **management**: En este modulo se encuentran aquellas funciones propias de la aplicacion...
  (Section 14, p. 19)
  - **commands**: En este modulo se encuentran aquellas funciones propias de la aplicacion...
    (Section 15, p. 20)
    - **easydata_d2rq**: Este modulo contiene el comando easydata_d2rq que se ejecutara desde el...
      (Section 16, p. 21)
    - **loadmodels**: Este modulo contiene comando loadmodels que se ejecutara desde el manage.py

de...
(Section 17, p. 23)

- **models**: En este modulo se encuentran almacenados todos los modelos que seran usados en...
  (Section 18, p. 25)
  - **entidad**: Este modelo se encarga de almacenar la informacion relativa a las diferentes...
    (Section 19, p. 26)
  - **field**: Este modelo se encarga de almacenar la informacion relativa a los diferentes...
    (Section 20, p. 28)
  - **modelo**: Este modelo se encarga de almacenar la informacion relativa a los diferentes...
    (Section 21, p. 33)
  - **namespace**: Este modelo se encarga de almacenar la informacion relativa a las diferentes...
    (Section 22, p. 36)
  - **propiedad**: Este modelo se encarga de almacenar la informacion relativa a las diferentes...
    (Section 23, p. 38)
- **parsing**: Este paquete llamado parsin, se encarga del parseo de ficheros con la especificacion de los distintos namespaces que se incluiran dentro de la aplicacion.
  (Section 24, p. 40)
  - **parseador**: En este modulo se almacena la clase Parseador, de la cual deberan de heredar todos los parseadores que se deseen implementar.
    (Section 25, p. 41)
  - **parserfactory**: En este modulo se almacena el factory que se encargara en funcion del formato...
    (Section 26, p. 43)
  - **rdf_parser**: En este modulo se almacena la version de Parseador, que se encarga de realizar...
    (Section 27, p. 44)
  - **registers**: En este modulo se definen los distintos tipos de Registros que se van a utilizar, que son registros para las entidades y registros para las propiedades.
    (Section 28, p. 47)
  - **schema_parser**: En este modulo se almacena la version de Parseador, que se encarga de realizar...
    (Section 29, p. 52)
- **templatetags**: En este modulo se encuentran los template tags que permitiran al usuario realizar la apertura de sus datos, incluyendo dicha informacion en las plantillas de Django.
  (Section 30, p. 56)
  - **easydata_links**: En este modulo se define la funcion easydata_include_link la cual permite crear enlaces html a los datos de instancias de modelos concretas.
    (Section 31, p. 57)
  - **easydata_microdata**: Este modulo contiene todos los template tags que permiten la insercion de los datos referentes a una intancia concreta de un modelo haciendo uso de formato microdata.
    (Section 32, p. 58)
  - **easydata_rdfa**: Este modulo contiene todos los template tags que permiten la insercion de los datos referentes a una intancia concreta de un modelo haciendo uso de formato rdfa.
    (Section 33, p. 61)
  - **get_data**: Este modulo contiene todos los template tags que permiten la insercion de los datos referentes a una intancia concreta de un modelo haciendo uso de formato microdata.
    (Section 34, p. 64)
- **tests**: En este fichero se han implementado los distintos test que se haran a la aplicacion easydata, para poder comprobar en cierta medida que esta funciona correctamente.
  (Section 35, p. 65)
- **urls**: En este fichero se encuentran todas las urls definidas en la aplicacion EasyData junto con las vistas que tienen asociadas.
  (Section 36, p. 71)

- **utils**: En este modulo se definen una serie de funciones auxiliares, que ayudaran en las tareas de mapeo y publicacion de datos.
  (Section 37, p. 72)
- **views**: En este modulo se almacenan todas las vistas que se usaran en la aplicacion...
  (Section 38, p. 74)
  - **information**: Este modulo almacena las vistas tanto de bienvenida, como las vistas donde se...
    (Section 39, p. 75)
  - **map**: En este modulo se almacenan las vistas que estan relacionadas con el mapeo de los modelos y de los fields.
    (Section 40, p. 77)
  - **modelo**: En este modulo se almacena las vistas que se encargan de realizar la...
    (Section 41, p. 79)
  - **namespace**: En este modulo se almacenan las vistas que gestionan los namespaces, tanto como...
    (Section 42, p. 80)
  - **publish**: En este modulo se almacenan las vistas que se encargan de la publicacion de...
    (Section 43, p. 81)
  - **sesiones**: Este modulo almacena las vistas tanto de bienvenida, como las vistas donde se...
    (Section 44, p. 82)

2 Package project.easydata.decorators

```
En este modulo se encuentran definida el decorador encargado de comprobar en
las vistas de la aplicacion que el usuario que intenta acceder a la vista, esta
logueado y tiene los permisos de superusuario.
```

Author: llerena

2.1 Modules

- **decorators**: En este modulo se encuentran definida el decorador encargado de comprobar en las vistas de la aplicacion que el usuario que intenta acceder a la vista, esta logueado y tiene los permisos de superusuario.
(Section 3, p. 6)

3 Module project.easydata.decorators.decorators

En este modulo se encuentran definida el decorador encargado de comprobar en las vistas de la aplicacion que el usuario que intenta acceder a la vista, esta logueado y tiene los permisos de superusuario.

Author: llerena

3.1 Functions

easydata_super_member(*view_func*)

Comprueba que un usuario esta logueado y que es super usuario. Si no se cumpliese este requisito, redirige al formulario de login.

4 Package project.easydata.forms

```
En este modulo se almacenan tanto los formularios ordinarios como los basados
en modelos que se utilizan en la aplicacion EasyData
```

Author: llerena

4.1 Modules

- **forms**: En este modulo se almacenan tanto los formularios ordinarios de Django que se...
  (Section 5, p. 8)
  - **d2rqform**: Este modulo implementa el formulario para captar ficheros d2rq y modificarlos...
    (Section 6, p. 9)
- **modelforms**: En este modulo se almacenan tanto los formularios basados en modelos que se...
  (Section 7, p. 10)
  - **editnamespace**: En este modulo se almacena el formulario que se utiliza para realizar la edicion de namespaces.
    (Section 8, p. 11)
  - **fieldpropiedad**: En este modulo se almacena el formulario que se utiliza para realizar el mapeo de los fields con las propiedades existentes.
    (Section 9, p. 12)
  - **modeloentidad**: En este modulo se almacena el formulario que se utiliza para realizar el mapeo de los modelos con las entidades existentes.
    (Section 10, p. 14)
  - **newnamespace**: En este modulo se almacena el formulario que se utiliza para realizar la creacion de nuevos namespaces.
    (Section 11, p. 16)
  - **visibilityfields**: En este modulo se almacena el formulario que se utiliza para realizar la configuracion de visibilidad de los fields.
    (Section 12, p. 17)
  - **visibilitymodels**: En este modulo se almacena el formulario que se utiliza para realizar la configuracion de visibilidad de los modelos.
    (Section 13, p. 18)

5 Package project.easydata.forms.forms

```
En este modulo se almacenan tanto los formularios ordinarios de Django que se
utilizan en la aplicacion EasyData
```

Author: llerena

5.1 Modules

- **d2rqform**: Este modulo implementa el formulario para captar ficheros d2rq y modificarlos... *(Section 6, p. 9)*

6 Module project.easydata.forms.forms.d2rqform

Este modulo implementa el formulario para captar ficheros d2rq y modificarlos mismos con la configuracion de la aplicacion

Author: llerena

6.1 Class D2RqForm

django.forms.Form
└ **project.easydata.forms.forms.d2rqform.D2RqForm**

Este formulario se utiliza para captar un fichero del tipo D2Rq de tal forma que se modifique esta añadiendosele el etiquetado configurado en la aplicacion

6.1.1 Class Variables

Name	Description
archivo	**Value:** `forms.FileField(required= True)`

7 Package project.easydata.forms.modelforms

```
En este modulo se almacenan tanto los formularios basados en modelos que se
utilizan en la aplicacion EasyData
```

Author: llerena

7.1 Modules

- **editnamespace**: En este modulo se almacena el formulario que se utiliza para realizar la edicion de namespaces.
  (Section 8, p. 11)
- **fieldpropiedad**: En este modulo se almacena el formulario que se utiliza para realizar el mapeo de los fields con las propiedades existentes.
  (Section 9, p. 12)
- **modeloentidad**: En este modulo se almacena el formulario que se utiliza para realizar el mapeo de los modelos con las entidades existentes.
  (Section 10, p. 14)
- **newnamespace**: En este modulo se almacena el formulario que se utiliza para realizar la creacion de nuevos namespaces.
  (Section 11, p. 16)
- **visibilityfields**: En este modulo se almacena el formulario que se utiliza para realizar la configuracion de visibilidad de los fields.
  (Section 12, p. 17)
- **visibilitymodels**: En este modulo se almacena el formulario que se utiliza para realizar la configuracion de visibilidad de los modelos.
  (Section 13, p. 18)

8 Module project.easydata.forms.modelforms.editnamespace

```
En este modulo se almacena el formulario que se utiliza para realizar la
edicion de namespaces.
```

Author: llerena

8.1 Class EditNamespaceForm

django.forms.ModelForm
└ **project.easydata.forms.modelforms.editnamespace.EditNamespaceForm**

```
Este formulario se utiliza para la creacion de nuevos namespaces
```

8.1.1 Class Variables

Name	Description
namespace	**Value:** `forms.CharField(required= True, widget= forms.TextInput(a...`
short_name	**Value:** `forms.CharField(required= True, widget= forms.TextInput(a...`
url	**Value:** `forms.URLField(required= False, widget= forms.TextInput(a...`
archivo	**Value:** `forms.FileField(label= _("File"), required= False)`
formato	**Value:** `forms.ChoiceField(label= _("Format"), choices= NameSpace....`

9 Module project.easydata.forms.modelforms.fieldpropiedad

```
En este modulo se almacena el formulario que se utiliza para realizar el mapeo
de los fields con las propiedades existentes.
```

Author: llerena

9.1 Class SelectProperties

django.forms.Select
└ **project.easydata.forms.modelforms.fieldpropiedad.SelectProperties**

```
Creacion de widget propio para que incluya en cada uno de los options del
select, la etiqueta y la descripcion de la propiedad concreta
```

9.1.1 Methods

render_option(*self*, *selected_choices*, *option_value*, *option_label*)

9.2 Class FieldPropiedadForm

django.forms.ModelForm
└ **project.easydata.forms.modelforms.fieldpropiedad.FieldPropiedadForm**

```
Este formulario se encarga de realizar el mapeo de los fields con las
propiedades de los namespaces
```

9.2.1 Methods

__init__(*self*, **args*, ***kwargs*)

save(*self*, **args*, ***kwargs*)

```
Sobreescribo el metodo save para que al salvar la instancia, se cargue
el field de propiedad con el queryset adecuado
```

9.2.2 Class Variables

Name	Description
namespace	**Value:** `forms.ModelChoiceField(queryset= NameSpace.objects.all()....`
propiedad	**Value:** `forms.ModelChoiceField(queryset= Propiedad.objects.all()....`

10 Module project.easydata.forms.modelforms.modeloentidad

En este modulo se almacena el formulario que se utiliza para realizar el mapeo de los modelos con las entidades existentes.

Author: llerena

10.1 Class SelectEntities

django.forms.Select
└ **project.easydata.forms.modelforms.modeloentidad.SelectEntities**

Creacion de widget propio para que incluya en cada uno de los options del select, la etiqueta y la descripcion de la entidad concreta

10.1.1 Methods

render_option(*self*, *selected_choices*, *option_value*, *option_label*)

10.2 Class ModeloEntidadForm

django.forms.ModelForm
└ **project.easydata.forms.modelforms.modeloentidad.ModeloEntidadForm**

Este formulario se encarga de realizar el mapeo de los modelos con las entidades de los namespaces

10.2.1 Methods

___init___(*self*, **args*, ***kwargs*)

save(*self*, **args*, ***kwargs*)

clean(*self*)

10.2.2 Class Variables

Name	Description
namespace	**Value:** forms.ModelChoiceField(queryset= NameSpace.objects.all()...

continued on next page

Name	Description
entidad	**Value:** `forms.ModelChoiceField(queryset= Entidad.objects.all().or...`

11 Module project.easydata.forms.modelforms.newnamespace

```
En este modulo se almacena el formulario que se utiliza para realizar la
creacion de nuevos namespaces.
```

Author: llerena

11.1 Class NewNamespaceForm

django.forms.ModelForm
└ **project.easydata.forms.modelforms.newnamespace.NewNamespaceForm**

```
Este formulario se utiliza para la creacion de nuevos namespaces
```

11.1.1 Methods

clean(*self*)

11.1.2 Class Variables

Name	Description
formato	**Value:** `forms.ChoiceField(label= _("Format"), choices= NameSpace....`
namespace	**Value:** `forms.CharField(required= True, widget= forms.TextInput(a...`
short_name	**Value:** `forms.CharField(required= True, widget= forms.TextInput(a...`
url	**Value:** `forms.URLField(required= False, widget= forms.TextInput(a...`
archivo	**Value:** `forms.FileField(label= _("File"), required= False)`

12 Module project.easydata.forms.modelforms.visibilityfields

```
En este modulo se almacena el formulario que se utiliza para realizar la
configuracion de visibilidad de los fields.
```

Author: llerena

12.1 Class VisibilityFieldForm

django.forms.ModelForm
└ **project.easydata.forms.modelforms.visibilityfields.VisibilityFieldForm**

```
Este formulario se utiliza para realizar la configuracion de la visibilidad
de los fields de la aplicacion.
```

12.1.1 Methods

___init___(*self*, **args*, ***kwargs*)

13 Module project.easydata.forms.modelforms.visibilitymodels

```
En este modulo se almacena el formulario que se utiliza para realizar la
configuracion de visibilidad de los modelos.
```

Author: llerena

13.1 Class VisibilityModelsForm

django.forms.ModelForm
└ **project.easydata.forms.modelforms.visibilitymodels.VisibilityModelsForm**

```
Este formulario se utiliza para realizar la configuracion de la visibilidad
de los modelos de la aplicacion.
```

13.1.1 Methods

___init___(*self*, **args*, ***kwargs*)

14 Package project.easydata.management

```
En este modulo se encuentran aquellas funciones propias de la aplicacion
EasyData que podran ejecutarse desde el manage.py de Django
```

Author: llerena

14.1 Modules

- **commands**: En este modulo se encuentran aquellas funciones propias de la aplicacion...
  (Section 15, p. 20)
  - **easydata_d2rq**: Este modulo contiene el comando easydata_d2rq que se ejecutara desde el...
    (Section 16, p. 21)
  - **loadmodels**: Este modulo contiene comando loadmodels que se ejecutara desde el manage.py de...
    (Section 17, p. 23)

15 Package project.easydata.management.commands

```
En este modulo se encuentran aquellas funciones propias de la aplicacion
EasyData que podran ejecutarse desde el manage.py de Django
```

Author: llerena

15.1 Modules

- **easydata_d2rq**: Este modulo contiene el comando easydata_d2rq que se ejecutara desde el... *(Section 16, p. 21)*
- **loadmodels**: Este modulo contiene comando loadmodels que se ejecutara desde el manage.py de... *(Section 17, p. 23)*

16 Module project.easydata.management.commands.easydata_d2rq

```
Este modulo contiene el comando easydata_d2rq que se ejecutara desde el
manage.py de Django, el cual se encarga de generar el fichero de configuracion
para d2rq en funcion de la configuracion realizada en la aplicacion
```

16.1 Class Command

object
└
django.core.management.base.BaseCommand
└
project.easydata.management.commands.easydata_d2rq.Comma

```
Esta clase se encarga de implementar el comando d2rq del manage.py
```

16.1.1 Methods

handle(*self*, **args*, ***options*)

```
Este command sirve para generar un fichero d2rq con la configuracion
realizada en la aplicacion
```

Overrides: django.core.management.base.BaseCommand.handle

get_datatype(*self*, *atributo*)

```
Busca el tipo de dato del un determinado field y devuelve su
representacion en el espacio de nombres xsd
```

get_pk_field(*self*, *Modelo*)

```
Devuelve el nombre del campo que es clave primaria del modelo
```

Inherited from django.core.management.base.BaseCommand

___init___(), create_parser(), execute(), get_version(), print_help(), run_from_argv(), usage(), validate()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

16.1.2 Properties

Name	Description
Inherited from object ___class___	

16.1.3 Class Variables

Name	Description
args	**Value:** `'<app_name>'`
help	**Value:** `'Genera la configuracion para d2rq en funcion de la reali...`
Inherited from django.core.management.base.BaseCommand can_import_settings, option_list, output_transaction, requires_model_validation	

17 Module project.easydata.management.commands.loadmodels

Este modulo contiene comando loadmodels que se ejecutara desde el manage.py de Django, el cual realizara la captacion de los distintos modelos y fields del proyecto Django donde se encuetra instalada la aplicacion EasyData

17.1 Class Command

object

django.core.management.base.BaseCommand

project.easydata.management.commands.loadmod

Esta clase se encarga de implementar el comando loadmodels de manage.py

17.1.1 Methods

handle(*self*, **args*, ***options*)

The actual logic of the command. Subclasses must implement this method.

Overrides: django.core.management.base.BaseCommand.handle extit(inherited documentation)

elimina_modelos(*self*)

Iterate over all the models in the data base, and check if exist or not

elimina_fields(*self*)

Iterate over all the models's fields in the data base, and check if the fields already exist or not

get_relation_type(*self*, *clase_relacion*)

```
Return the relatio type. If there are custom relations, it search their
base types relations, util find a ForeignKey, OneToOneField or
ManyToManyField.
```

Inherited from django.core.management.base.BaseCommand

___init___(), create_parser(), execute(), get_version(), print_help(), run_from_argv(), usage(), validate()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

17.1.2 Properties

Name	Description
Inherited from object ___class___	

17.1.3 Class Variables

Name	Description
args	**Value:** `'<app_name>'`
help	**Value:** `'Load the models and fields of the project, into the Easy...`
Inherited from django.core.management.base.BaseCommand can_import_settings, option_list, output_transaction, requires_model_validation	

18 Package project.easydata.models

```
En este modulo se encuentran almacenados todos los modelos que seran usados en
la aplicacion EasyData, para el almacenamiento de namespaces, y de los moelos y
fields que componen a los proyectos
```

18.1 Modules

- **entidad**: Este modelo se encarga de almacenar la informacion relativa a las diferentes...
  (Section 19, p. 26)
- **field**: Este modelo se encarga de almacenar la informacion relativa a los diferentes...
  (Section 20, p. 28)
- **modelo**: Este modelo se encarga de almacenar la informacion relativa a los diferentes...
  (Section 21, p. 33)
- **namespace**: Este modelo se encarga de almacenar la informacion relativa a las diferentes...
  (Section 22, p. 36)
- **propiedad**: Este modelo se encarga de almacenar la informacion relativa a las diferentes...
  (Section 23, p. 38)

19 Module project.easydata.models.entidad

Este modelo se encarga de almacenar la informacion relativa a las diferentes entidades que existen en un determinado namespace

19.1 Class Entidad

django.db.models.Model
└ **project.easydata.models.entidad.Entidad**

Este modelo se encarga de registrar la informacion referente a las distintas entidades que componen a un namespace

19.1.1 Methods

___unicode___(*self*)

Devuelve la representacion unicode de la instancia del modelo

generate_type(*self*)

devuelve la url que representa a la entidad
:return:devuelve un string con la url que representa a la entidad

generate_type_short(*self*)

Devuelve la cadena con el formato namespace:entidad
:return: devuelve la entidad junto con el nombre del namespace

get_publish_url(*self*)

Devuelve una url de ejemplo donde se podria encontrar la informacion de esta entidad
:return: devuelve string con url de ejemplo de la entidad

19.1.2 Class Variables

Name	Description
nombre	es el nombre de la entidad **Value:** models.CharField(max_length= 200)
namespace	es el namespace con el que esta asociado **Value:** models.ForeignKey(NameSpace, related_name= 'entidades', o...
padres	son los padres de los que hereda propiedades **Value:** models.ManyToManyField('Entidad', verbose_name= u"Entidad...
descripcion	es una descripcion que tiene asociada la entidad **Value:** models.CharField(max_length= 500, null= True, blank= True)
etiqueta	es la etiqueta que lo identifica **Value:** models.CharField(max_length= 100, null= True, blank= True)

19.2 Class EntidadAdmin

django.contrib.admin.ModelAdmin

project.easydata.models.entidad.EntidadAdmin

Clase para registar en el admin de Django

19.2.1 Class Variables

Name	Description
list_display	**Value:** ['nombre', 'etiqueta', 'descripcion']
search_fields	**Value:** ['nombre', 'etiqueta', 'descripcion']

20 Module project.easydata.models.field

Este modelo se encarga de almacenar la informacion relativa a los diferentes fields, tanto atributos como relaciones, que existen en un modelo

20.1 Class Field

django.db.models.Model
└ **project.easydata.models.field.Field**

Este modelo se encarga de registrar la informacion referente a los distintos field que componen un modelo

20.1.1 Methods

get_field_class(*self*)

Devuelve la clase django del field

___unicode___(*self*)

20.1.2 Class Variables

Name	Description
nombre	es el nombre del field **Value:** models.CharField(max_length= 200)
TIPOS_VISIBILIDAD	tipos de visibilidad que puede tener un field **Value:** 'V', 'Visible', ('P', 'Privado'),
visibilidad	almacena la visibilidad que posee **Value:** models.CharField(max_length= 1, choices= TIPOS_VISIBILIDA...
modelo	modelo al que pertenece **Value:** models.ForeignKey(Modelo, related_name= 'fields', on_dele...

continued on next page

Name	Description
propiedad	`propiedad con la que esta mapeado` **Value:** `models.ForeignKey("Propiedad", related_name= "fields", nu...`

20.2 Class Atributo

django.db.models.Model
└
project.easydata.models.field.Field
└
project.easydata.models.field.Atributo

`Este modelo se encarga de registrar la informacion referente a los distintos atributos que componen un modelo`

20.2.1 Methods

___unicode___(*self*) Overrides: project.easydata.models.field.Field.___unicode___

Inherited from project.easydata.models.field.Field(Section 20.1)

get_field_class()

20.2.2 Class Variables

Name	Description
tipo_field	**Value:** `models.CharField(max_length= 200)`
Inherited from project.easydata.models.field.Field (Section 20.1) TIPOS_VISIBILIDAD, modelo, nombre, propiedad, visibilidad	

20.3 Class Relacion

django.db.models.Model

project.easydata.models.field.Field

project.easydata.models.field.Relacion

```
Este modelo se encarga de registrar la informacion referente a las
distintas relaciones que componen un modelo
```

20.3.1 Methods

___unicode___(*self*)

Overrides: project.easydata.models.field.Field.___unicode___

save(*self*, **args*, ***kwargs*)

Inherited from project.easydata.models.field.Field(Section 20.1)

get_field_class()

20.3.2 Class Variables

Name	Description
TIPOS_RELACION	`tipos de relaciones que se pueden dar` **Value:** `'O', 'OneToOneField', ('M', 'ManyToManyField'), ('FO', 'F...`
tipo_relacion	`tipo de relacion del que se trata` **Value:** `models.CharField(max_length= 2, choices= TIPOS_RELACION, ...`
modelo_relacionado	`modelo con el que se relaciona` **Value:** `models.ForeignKey(Modelo, related_name= 'relaciones', on_...`
inversa	`relacion a la inversa` **Value:** `models.OneToOneField('Relacion', null= True)`
Inherited from project.easydata.models.field.Field (Section 20.1)	

continued on next page

Name	Description
TIPOS_VISIBILIDAD, modelo, nombre, propiedad, visibilidad	

20.4 Class FieldAdmin

django.contrib.admin.ModelAdmin

project.easydata.models.field.FieldAdmin

`Clase para registar en el admin de Django`

20.4.1 Class Variables

Name	Description
list_display	**Value:** `['nombre', 'visibilidad', 'modelo']`
search_fields	**Value:** `['nombre', 'visibilidad', 'modelo']`

20.5 Class AtributoAdmin

django.contrib.admin.ModelAdmin

project.easydata.models.field.AtributoAdmin

`Clase para registar en el admin de Django`

20.5.1 Class Variables

Name	Description
list_display	**Value:** `['nombre', 'visibilidad', 'modelo', 'tipo_field']`
search_fields	**Value:** `['nombre', 'visibilidad', 'modelo', 'tipo_field']`

20.6 Class RelacionAdmin

django.contrib.admin.ModelAdmin

project.easydata.models.field.RelacionAdmin

`Clase para registar en el admin de Django`

20.6.1 Class Variables

Name	Description
list_display	**Value:** `['nombre', 'visibilidad', 'modelo', 'tipo_relacion']`
search_fields	**Value:** `['nombre', 'visibilidad', 'modelo', 'tipo_relacion']`

21 Module project.easydata.models.modelo

Este modelo se encarga de almacenar la informacion relativa a los diferentes modelos que pueden existir en la aplicacion Django

21.1 Class Modelo

django.db.models.Model
└ **project.easydata.models.modelo.Modelo**

Este modelo representa a cada uno de los modelos del proyecto Django

21.1.1 Methods

___unicode___(*self*)

devolver_modelo(*self*)

Devuele la clase a la que representa el modelo

get_parent_link(*self*, *modelo*)

en caso de herencias, busca el field que apunta al modelo padre

generate_url(*self*, *instance*, *default*=False)

devuelve la url con la publicacion de datos para una determinada instancia
:param instance: es la instancia de la que se desea obtener la url
:return: devuelve la url de la instancia

generate_url_without_instance(*self*)

devuelve la url para la publicacion de todos los datos del modelo
:return: devuelve la url para el modelo

generate_full_url(*self*)

```
Devuelve una url de ejemplo donde podrian consultar los datos del
modelo
:return: devuelve url de ejemplo
```

get_d2rq_url(*self*)

```
Devuelve url con la informacion de una instancia, en formato d2rq
:return: devuelve cadena con url para d2rq
```

21.1.2 Class Variables

Name	Description
nombre	`es el nombre del modelo` **Value:** `models.CharField(max_length= 200)`
aplicacion	`es el nombre de la aplicacion donde se encuentra el modelo` **Value:** `models.CharField(max_length= 200)`
TIPOS_VISIBILIDAD	`tipos de visibilidad que posee el modelo` **Value:** `'V', _('Visible'), ('P', _('Private')),`
visibilidad	`visibilidad que tiene el modelo` **Value:** `models.CharField(max_length= 1, choices= TIPOS_VISIBILIDA...`
entidad	`entidad con la que esta mapeado el modelo` **Value:** `models.ForeignKey("Entidad", related_name= 'modelos', nul...`

21.2 Class ModeloAdmin

django.contrib.admin.ModelAdmin

project.easydata.models.modelo.ModeloAdmin

`Clase para registar en el admin de Django`

21.2.1 Class Variables

Name	Description
list_display	**Value:** `['nombre', 'aplicacion']`
search_fields	**Value:** `['nombre', 'aplicacion']`

22 Module project.easydata.models.namespace

Este modelo se encarga de almacenar la informacion relativa a las diferentes namespaces que se podran almacenar en la aplicacion

22.1 Class NameSpace

django.db.models.Model
└
project.easydata.models.namespace.NameSpace

Este modelo representa a los namespaces u ontologias que se cargaran en la aplicacion para realizar los mapeos

22.1.1 Methods

___unicode___(*self*)

get_url(*self*)

devuelve la url del namespace

get_type(*self*)

devuelve el nombre corto del namespace

22.1.2 Class Variables

Name	Description
namespace	es el nombre del namespace **Value:** models.CharField(max_length= 40, unique= True)
short_name	Es un nombre corto para el namespace que se usara en la pu datos **Value:** models.CharField(max_length= 15, unique= True, validators...

continued on next page

Name	Description
formatos	`son los formatos disponibles para la carga de namespaces` **Value:** `'R', 'RDF/XML', ('N', 'Ntriples'),`
url	`es la url donde se encuentra el namespace` **Value:** `models.URLField()`

22.2 Class NameSpaceAdmin

django.contrib.admin.ModelAdmin
└ **project.easydata.models.namespace.NameSpaceAdmin**

`Clase para registar en el admin de Django`

22.2.1 Class Variables

Name	Description
list_display	**Value:** `['namespace', 'url', 'short_name']`
search_fields	**Value:** `['namespace', 'url', 'short_name']`

23 Module project.easydata.models.propiedad

Este modelo se encarga de almacenar la informacion relativa a las diferentes propiedades que existen en un determinado namespace

23.1 Class Propiedad

django.db.models.Model
└ **project.easydata.models.propiedad.Propiedad**

Este modelo representa a las distintas propiedades de las que esta compuesto un determinado namespace

23.1.1 Methods

___unicode___(*self*)

get__full__name(*self*)

devuelve el nombre delnamespace junto con el nombre de la propiedad en el formato namespace:propiedad

23.1.2 Class Variables

Name	Description
nombre	es el nombre de la propiedad **Value:** models.CharField(max_length= 200)
simple	indica si la propiedad es simple (tipo de datos simple) **Value:** models.BooleanField(default= True)
namespace	indica el namespace al que pertenece la propiedad **Value:** models.ForeignKey(NameSpace, related_name= 'propiedades',...

continued on next page

Name	Description
tipo	son los posibles tipos a los que puede hacer referencia la **Value:** models.ManyToManyField("Entidad", verbose_name= u"Relacio...
entidades	entidades a las que pertenece dicha propiedad **Value:** models.ManyToManyField("Entidad", verbose_name= u"Entidad...
descripcion	descripcion asociada que tenga la propiedad **Value:** models.CharField(max_length= 500)
etiqueta	etiqueta identificativa que tenga la propiedad **Value:** models.CharField(max_length= 100)

23.2 Class PropiedadAdmin

django.contrib.admin.ModelAdmin

project.easydata.models.propiedad.PropiedadAdmin

Clase para registar en el admin de Django

23.2.1 Class Variables

Name	Description
list_display	**Value:** ['nombre', 'simple', 'descripcion', 'etiqueta']
search_fields	**Value:** ['nombre', 'simple', 'descripcion', 'etiqueta']

24 Package project.easydata.parsing

Este paquete llamado parsin, se encarga del parseo de ficheros con la especificacion de los distintos namespaces que se incluiran dentro de la aplicacion.

La clase principal es Parseador, de tal forma que cuando se desee generar un nuevo parseador, se debera crear una nueva clase que herede de esta. Posterioemente se debera de incluir esta nueva clase en el Factory, para que contemple el nuevo formato.

Author: llerena

24.1 Modules

- **parseador**: En este modulo se almacena la clase Parseador, de la cual deberan de heredar todos los parseadores que se deseen implementar.
  (Section 25, p. 41)
- **parserfactory**: En este modulo se almacena el factory que se encargara en funcion del formato...
  (Section 26, p. 43)
- **rdf_parser**: En este modulo se almacena la version de Parseador, que se encarga de realizar...
  (Section 27, p. 44)
- **registers**: En este modulo se definen los distintos tipos de Registros que se van a utilizar, que son registros para las entidades y registros para las propiedades.
  (Section 28, p. 47)
- **schema_parser**: En este modulo se almacena la version de Parseador, que se encarga de realizar...
  (Section 29, p. 52)

25 Module project.easydata.parsing.parseador

En este modulo se almacena la clase Parseador, de la cual deberan de heredar todos los parseadores que se deseen implementar. Proporciona la interfaz que deberan de implementar todos los parseadores

Author: llerena

25.1 Class Parser

object
└ **project.easydata.parsing.parseador.Parser**

Esta es una clase abstracta de la que heredaran los pareseadores que se deseen implementar para captar las entidades y propiedades de un determinado namespace.

25.1.1 Methods

___init___(*self*)

Constructor

Overrides: object.___init___

parse(*self*)

Metodo abstracto: este metodo se encarga de parsear los datos y completar las listas de propiedades y relaciones

register(*self*)

Save the instances of entities and properties in the DB

update(*self*)

```
Update the current registers
```

validate(*self*)

```
Metodo abstracto que se encarga de validar los datos captados en caso
de que existiese algun tipo de validacion
```

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

25.1.2 Properties

Name	Description
Inherited from object ___class___	

25.1.3 Class Variables

Name	Description
grafo	`almacena el grafo con los datos al leer el fichero` **Value:** `None`
list_entities	`almacena el listado de propiedades que se han captado` **Value:** `None`
list_properties	`almacena las propiedades que se han captado` **Value:** `None`
list_relations	**Value:** `None`
namespace	`almacena el namespace al que se va a asociar las entidades captadas` **Value:** `None`

26 Module project.easydata.parsing.parserfactory

```
En este modulo se almacena el factory que se encargara en funcion del formato
del fichero que se desea parsear para cargar un nuevo namespace, de devolver la
instancia del parseador adecuado inicializado
```

Author: llerena

26.1 Functions

parser_factory(*archivo*, *formato*, *namespace*)

```
Esta funcion hace de Factory, de tal forma que a partir del formato del
fichero, devolvera el parseador asociado
:param archivo: archivo que se va a parsear
:param formato: formato en el que se encuentra el archivo
:param namespace: namespace donde se van a incluir las entidades y
    propiedades parseadas
:return: devuelve un objecto de la clase Parseador inicializado.
```

27 Module project.easydata.parsing.rdf_parser

En este modulo se almacena la version de Parseador, que se encarga de realizar el parseo de los datos a partir de ficheros RDF, tanto en formato XML como Ntriples

Author: llerena

27.1 Class ParserXML

object
project.easydata.parsing.parseador.Parser
project.easydata.parsing.rdf_parser.ParserXML

This class parse a schema ontology from a XML file.

27.1.1 Methods

___init___(*self*, *file*, *namespace*)

Constructor

Overrides: object.___init___ extit(inherited documentation)

parse(*self*)

Este metodo se encarga de realizar el parseo de un namespace en el formato RDF

Overrides: project.easydata.parsing.parseador.Parser.parse

Inherited from project.easydata.parsing.parseador.Parser(Section 25.1)

register(), update(), validate()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

27.1.2 Properties

Name	Description
Inherited from object	
___class___	

27.1.3 Class Variables

Name	Description
Inherited from project.easydata.parsing.parseador.Parser (Section 25.1)	
grafo, list_entities, list_properties, list_relations, namespace	

27.2 Class ParserN3

object

project.easydata.parsing.parseador.Parser

project.easydata.parsing.rdf_parser.ParserXML

project.easydata.parsing.rdf_parser.ParserN3

```
This class parse a schema ontology from a NTriples file.
```

27.2.1 Methods

___init___(*self*, *file*, *namespace*)

```
Constructor
```

Overrides: object.___init___ extit(inherited documentation)

Inherited from project.easydata.parsing.rdf_parser.ParserXML(Section 27.1)

parse()

Inherited from project.easydata.parsing.parseador.Parser(Section 25.1)

register(), update(), validate()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(),

___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

27.2.2 Properties

Name	Description
Inherited from object ___class___	

27.2.3 Class Variables

Name	Description
Inherited from project.easydata.parsing.parseador.Parser (Section 25.1) grafo, list_entities, list_properties, list_relations, namespace	

28 Module project.easydata.parsing.registers

En este modulo se definen los distintos tipos de Registros que se van a utilizar, que son registros para las entidades y registros para las propiedades. Estos se encargaran de almacenar los datos de cada uno, validar los mismos y almacenarlos en la base de datos.

Author: llerena

28.1 Class RegisterEntity

object
└ **project.easydata.parsing.registers.RegisterEntity**

Este registro se usa para almacenar las entidades de las que estara compuesto un determinado namespace. Este registro se encargara de administrar la correcta creacion de la entidad, asi como de crear sus relaciones entre ellos.

28.1.1 Methods

___init___(*self*, *name*, *description*, *label*, *namespace*, *fathers*, *url*="")

Constructor de la clase RegisterEntity que almacena los datos de una determinada entidad
:param name: Nombre de la entidad
:param description: Descripcion asociada a la entidad
:param label: etiqueta asociada a la entidad
:param namespace: namespace al que pertenece
:param fathers: padres del que hereda sus propiedades
:param url: url donde se encuentra su especificacion

Overrides: object.___init___

register(*self*)

Salva en la base de datos la entidad que almacena el registro

register_fathers(*self*)

```
Agrega las relaciones de la entidad con su padre, si posee
```

update_registers(*self*)

```
This method is used when the namespace exists, and you only want to
update the fields, or create the fields that do not exist before
```

update_fathers(*self*)

```
This methos is only used when the namespace exists, and you only want
to update the relations between the entities
```

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

28.1.2 Properties

Name	Description
Inherited from object ___class___	

28.1.3 Class Variables

Name	Description
name	`almacena el nombre de la entidad` **Value:** `None`
description	`almacena la descripcion asociada a la entidad` **Value:** `None`
label	`almacena la etiqueta asociada a la entidad` **Value:** `None`

continued on next page

Name	Description
namespace	almacena el namespace al que pertenece la entidad **Value:** None
fathers	almacena las entidades padre de las que hereda la entidad **Value:** None
entity	almacena la instancia de entidad una vez creada **Value:** None
url	almacena la url asociada a la entidad **Value:** None

28.2 Class RegisterProperty

object
└
project.easydata.parsing.registers.RegisterProperty

Este es un registro para almacenar las propiedades de las entidades. Este registro se encarga de almacenar una determinada propiedad, sus relaciones, tipo y demas caracteristicas, para posteriormente salvar las mismas en la base de datos.

28.2.1 Methods

___init___(*self*, *name*, *description*, *label*, *namespace*, *relations*, *types*, *url*="", *tipos_base*=[])

Constructor de la clase RegisterProperty que almacena los datos de una determinada propiedad
:param name: Nombre de la propiedad
:param description: Descripcion asociada a la propiedad
:param label: etiqueta asociada a la propiedad
:param namespace: namespace al que pertenece
:param relations: entidades con la que esta relacionadas
:param types: tipos de datos a los que hace referencia

Overrides: object.___init___

register(*self*)

Salva en la base de datos la propiedad que almacena el registro

register_relations(*self*)

Agrega las relaciones de la propiedad con las distintas entidades

update_registers(*self*)

This method is used when the namespace exists, and you only want to update the fields, or create the fields that do not exist before

update_relations(*self*)

Agrega las relaciones de la propiedad con las distintas entidades

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

28.2.2 Properties

Name	Description
Inherited from object ___class___	

28.2.3 Class Variables

Name	Description
name	nombre de la propidad **Value:** None

continued on next page

Name	Description
description	descripcion asociada a la propiedad **Value:** None
label	etiqueta asociada a la propiedad **Value:** None
relations	almacena las entidades con las que esta relacionada **Value:** None
namespace	almacena el namespace al que pertenece **Value:** None
property	almacena la instancia de la propiedad una vez creada **Value:** None
types	tipos a los que hace referencia la propiedad **Value:** None
simple	almacena si es una propiedad de tipo simple **Value:** None
url	almacena la url asociada a la propiedad **Value:** None

29 Module project.easydata.parsing.schema_parser

```
En este modulo se almacena la version de Parseador, que se encarga de realizar
el parseo de los datos a partir de ficheros json (experimental, solo funciona
para Schema)
```

Author: llerena

29.1 Class ParserJSON

object
project.easydata.parsing.parseador.Parser
project.easydata.parsing.schema_parser.ParserJSON

```
This class parse a schema ontology from a JSON file.
```

29.1.1 Methods

___init___(*self*, *file*, *namespace*)

```
Constructor
```

Overrides: object.___init___ extit(inherited documentation)

parse(*self*)

```
Catch the entities and properties from the JSON, and store on the
object
```

Overrides: project.easydata.parsing.parseador.Parser.parse

Inherited from project.easydata.parsing.parseador.Parser(Section 25.1)

register(), update(), validate()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

29.1.2 Properties

Name	Description
Inherited from object ___class___	

29.1.3 Class Variables

Name	Description
Inherited from project.easydata.parsing.parseador.Parser (Section 25.1) grafo, list_entities, list_properties, list_relations, namespace	

29.2 Class SchemaParserXML

object

project.easydata.parsing.parseador.Parser

project.easydata.parsing.schema_parser.SchemaPars

```
This class parse a schema ontology from a N-Triples file.
```

29.2.1 Methods

___init___(*self*, *file*, *namespace*)

```
Constructor
```

Overrides: object.___init___ extit(inherited documentation)

parse(*self*)

```
Metodo abstracto: este metodo se encarga de parsear los datos y
completar las listas de propiedades y relaciones
```

Overrides: project.easydata.parsing.parseador.Parser.parse extit(inherited documentation)

Inherited from project.easydata.parsing.parseador.Parser(Section 25.1)

register(), update(), validate()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

29.2.2 Properties

Name	Description
Inherited from object ___class___	

29.2.3 Class Variables

Name	Description
Inherited from project.easydata.parsing.parseador.Parser (Section 25.1) grafo, list_entities, list_properties, list_relations, namespace	

29.3 Class SchemaParserNT

object

project.easydata.parsing.parseador.Parser

project.easydata.parsing.schema_parser.SchemaParserXML

project.easydata.parsing.schema_par

```
This class parse a schema ontology from a N-Triples file.
```

29.3.1 Methods

___init___(*self*, *file*, *namespace*)

```
Constructor
```

Overrides: object.___init___ extit(inherited documentation)

Inherited from project.easydata.parsing.schema_parser.SchemaParserXML(Section 29.2)

parse()

Inherited from project.easydata.parsing.parseador.Parser(Section 25.1)

register(), update(), validate()

Inherited from object

___delattr___(), ___format___(), ___getattribute___(), ___hash___(), ___new___(), ___reduce___(), ___reduce_ex___(), ___repr___(), ___setattr___(), ___sizeof___(), ___str___(), ___subclasshook___()

29.3.2 Properties

Name	Description
Inherited from object	
___class___	

29.3.3 Class Variables

Name	Description
Inherited from project.easydata.parsing.parseador.Parser (Section 25.1)	
grafo, list_entities, list_properties, list_relations, namespace	

30 Package project.easydata.templatetags

```
En este modulo se encuentran los template tags que permitiran al usuario
realizar la apertura de sus datos, incluyendo dicha informacion en las
plantillas de Django.

Proporciona herramientas, tanto para la publicacion de los datos en formato
rdfa como en formato microdata.

Tambien permite crear links directos a las url con la informacion en RDF/XML de
instancias concretas de modelos.
```

Author: llerena

30.1 Modules

- **easydata_links**: En este modulo se define la funcion easydata_include_link la cual permite crear enlaces html a los datos de instancias de modelos concretas.
  (Section 31, p. 57)
- **easydata_microdata**: Este modulo contiene todos los template tags que permiten la insercion de los datos referentes a una intancia concreta de un modelo haciendo uso de formato microdata.
  (Section 32, p. 58)
- **easydata_rdfa**: Este modulo contiene todos los template tags que permiten la insercion de los datos referentes a una intancia concreta de un modelo haciendo uso de formato rdfa.
  (Section 33, p. 61)
- **get_data**: Este modulo contiene todos los template tags que permiten la insercion de los datos referentes a una intancia concreta de un modelo haciendo uso de formato microdata.
  (Section 34, p. 64)

31 Module project.easydata.templatetags.easydata_links

En este modulo se define la funcion easydata_include_link la cual permite crear enlaces html a los datos de instancias de modelos concretas.

31.1 Functions

easydata_include_link(*elemento*)

This tamplate tag get an instance element, and return a html a tag with the url where is published the information of the instance un RDF format.
:param elemento: is the instance that you want to generate the link
:return: HTML with the link to the RDF information of the elem

31.2 Variables

Name	Description
register	**Value:** `template.Library()`

32 Module project.easydata.templatetags.easydata_microdata

Este modulo contiene todos los template tags que permiten la insercion de los datos referentes a una intancia concreta de un modelo haciendo uso de formato microdata.

32.1 Functions

microdata_div_meta(*token*)

Generate the html with microdata using div and meta blocks
:param token: is the instance to generate the microdata
:return: HTML with the microdata tags and the instance's info

microdata_div_span(*token*)

Generate the html with microdata using div and span blocks
:param token: is the instance to generate the microdata
:return: HTML with the microdata tags and the instance's info

microdata_ul(*token*)

Generate the html with microdata using ul and li blocks
:param token: is the instance to generate the microdata
:return: HTML with the microdata tags and the instance's info

microdata_meta_field(*token*, *field*)

Generate the html of a concrete field with microdata and using div and meta blocks
:param token: is the instance to generate the microdata
:param field: is the concrete field to generate the info
:return: HTML with the microdata tags and the instance's info

microdata_span_field(*token*, *field*)

```
Generate the html of a concrete field with microdata and using div and span
blocks
:param token: is the instance to generate the microdata
:param field: is the concrete field to generate the info
:return: HTML with the microdata tags and the instance's info
```

microdata_li_field(*token*, *field*)

```
Generate the html of a concrete field with microdata and using ul and li
blocks
:param token: is the instance to generate the microdata
:param field: is the concrete field to generate the info
:return: HTML with the microdata tags and the instance's info
```

generate_html_microdata(*instance*, *tag1*, *tag2*, *content*, *field*=None)

```
Generate the html using the tags given
:param instance: is the instance to generate the microdata
:param tag1: the first tag to use
:param tag2: the second tag to use
:param field: is the concrete field to generate the info, if exists.
:return: HTML with the microdata tags
```

microdata_open_tag(*parser*, *token*)

```
Generate the open html tag with namespaces declaration and the instace
typeof in microdata format
```

microdata_open_tag_interno(*parser*, *token*)

```
Generate the open html tag with namespaces declaration and the instace
typeof in microdata format
```

32.2 Variables

Name	Description
register	**Value:** `template.Library()`

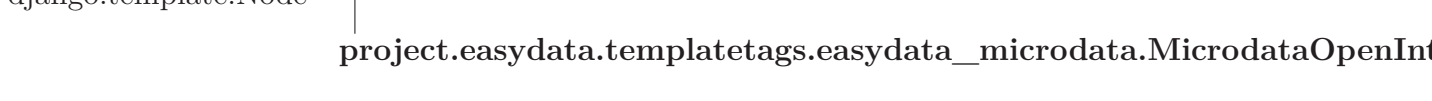
32.3 Class MicrodataOpenNode

32.3.1 Methods

___init___(*self*, *instancia*, *etiqueta*, *nodelist*)

render(*self*, *context*)

32.4 Class MicrodataOpenInternoNode



32.4.1 Methods

___init___(*self*, *padre*, *instancia*, *field*, *etiqueta*, *nodelist*)

render(*self*, *context*)

33 Module project.easydata.templatetags.easydata_rdfa

Este modulo contiene todos los template tags que permiten la insercion de los datos referentes a una intancia concreta de un modelo haciendo uso de formato rdfa.

33.1 Functions

rdfa_div(*token*)

```
Generate the html with rdfa format using div blocks
:param token: is the instance to generate the rdfa
:return: HTML with the rdfa tags and the instance's info
```

rdfa_div_span(*token*)

```
Generate the html with rdfa format using div and span blocks
:param token: is the instance to generate the rdfa
:return: HTML with the rdfa tags and the instance's info
```

rdfa_ul(*token*)

```
Generate the html with rdfa format using ul and li blocks
:param token: is the instance to generate the rdfa
:return: HTML with the rdfa tags and the instance's info
```

rdfa_div_field(*token*, *field*)

```
Generate the html with rdfa format for a concrete field, using div blocks
:param token: is the instance to generate the rdfa
:param field: is the concrete field to generate the info
:return: HTML with the rdfa tags and the instance's info
```

rdfa_span_field(*token*, *field*)

```
Generate the html with rdfa format for a concrete field, using div and span
blocks
:param token: is the instance to generate the rdfa
:param field: is the concrete field to generate the info
:return: HTML with the rdfa tags and the instance's info
```

rdfa_li_field(*token*, *field*)

```
Generate the html with rdfa format for a concrete field, using ul and li
blocks
:param token: is the instance to generate the rdfa
:param field: is the concrete field to generate the info
:return: HTML with the rdfa tags and the instance's info
```

generate_html_rdfa(*instance*, *tag1*, *tag2*, *content*, *field*=None)

```
Generate the html with the rdfa properties using the tags given
:param instance: is the instance to generate the rdfa
:param tag1: the first tag to use
:param tag2: the second tag to use
:return: HTML with the rdfa tags
```

rdfa_open_tag(*parser*, *token*)

```
Generate the open html tag with namespaces declaration and the instace
typeof in rdfa format
```

rdfa_open_tag_interno(*parser*, *token*)

```
Generate the open html tag with namespaces declaration and the instace
typeof in rdfa format
```

33.2 Variables

Name	Description
register	**Value:** `template.Library()`

33.3 Class RdfaOpenNode

django.template.Node
└ **project.easydata.templatetags.easydata_rdfa.RdfaOpenNode**

33.3.1 Methods

___init___(*self*, *instancia*, *etiqueta*, *nodelist*)

render(*self*, *context*)

33.4 Class RdfaOpenInternoNode

django.template.Node
└ **project.easydata.templatetags.easydata_rdfa.RdfaOpenInternoNode**

33.4.1 Methods

___init___(*self*, *padre*, *instancia*, *field*, *etiqueta*, *nodelist*)

render(*self*, *context*)

34 Module project.easydata.templatetags.get_data

Este modulo contiene todos los template tags que permiten la insercion de los datos referentes a una intancia concreta de un modelo haciendo uso de formato microdata.

34.1 Functions

get_instance_data(*elemento*, *field*=None)

```
Return the info of a concrete model instance
:param elemento: is the model instance
:param field: concrete field of the instance to generate the data.
:return: dict with the instance data.
```

35 Module project.easydata.tests

En este fichero se han implementado los distintos test que se haran a la aplicacion easydata, para poder comprobar en cierta medida que esta funciona correctamente.

35.1 Class NameSpaceTestCase

django.test.TestCase
└ **project.easydata.tests.NameSpaceTestCase**

Esta clase de tests se encarga de probar el modelo NameSpace

35.1.1 Methods

setUp(*self*)

Este metodo se encarga de preparar los datos para las pruebas

test_namespace_save(*self*)

Se encarga de probar que los namespaces se crear correctamente

test_namespaces_name(*self*)

Se encarga de probar que se corresponde el atributo name con el indicado

test_namespace_url(*self*)

Se encarga de probar que se corresponde el atributo url con el indicado

test_namespace_shortname(*self*)

```
Se encarga de probar que se corresponde el atributo short_name con el
indicado
```

35.2 Class EntidadTestCase

django.test.TestCase
└ **project.easydata.tests.EntidadTestCase**

```
Esta clase de tests se encarga de probar el modelo Entidad
```

35.2.1 Methods

setUp(*self*)

```
Este metodo se encarga de preparar los datos para las pruebas
```

test_entidad_get_tag(*self*)

```
Se encarga de probar que se corresponde el atributo etiqueta con el
indicado
```

test_entidad_get_description(*self*)

```
Se encarga de probar que se corresponde el atributo descripcion con el
indicado
```

test_entidades_namespaces(*self*)

```
Se encarga de probar que se le ha asignado correctamente el namespace
```

test_entidad_get_type(*self*)

Se encarga de probar que funciona correctamente el metodo get_type

test_entidad_get_type_short(*self*)

Se encarga de probar que funciona correctamente el metodo get_type_short

test_entidad_get_publish_url(*self*)

Se encarga de probar que funciona correctamente el metodo get_publish_url

35.3 Class PropiedadTestCase

django.test.TestCase
└ **project.easydata.tests.PropiedadTestCase**

Esta clase de tests se encarga de probar el modelo Propiedad

35.3.1 Methods

setUp(*self*)

Este metodo se encarga de preparar los datos para las pruebas

test_propiedad_get_tag(*self*)

Este caso de prueba se encarga de probar la asignacion de etiquetas

test_propiedad_get_description(*self*)

```
Este caso de prueba se encarga de probar la asignacion de una
descripcion
```

test_get_propiedad_entidad(*self*)

```
Este caso de prueba se encarga de probar que las entidades relacionadas
se asignan correctamente
```

test_get_propiedad_fullname(*self*)

```
Este caso de prueba se encarga de probar que el metodo get_full_name
funciona correctamente
```

35.4 Class ModeloTestCase

django.test.TestCase
└ **project.easydata.tests.ModeloTestCase**

```
Esta clase de tests se encarga de probar el modelo Modelo
```

35.4.1 Methods

setUp(*self*)

```
Este metodo se encarga de preparar los datos para las pruebas
```

test_devolver_modelo(*self*)

test_generate_url(*self*)

test_generate_url_without_instance(*self*)

test_generate_full_url(*self*)

test_get_d2rq_url(*self*)

35.5 Class UtilsTestCase

django.test.TestCase
└ **project.easydata.tests.UtilsTestCase**

```
Esta clase de test se encarga de realizar las pruebas pertinentes de los
diferentes utils implementados para la aplicacion easydata.
```

35.5.1 Methods

setUp(*self*)

```
Este metodo se encarga de preparar los datos para las pruebas
```

test_descubre_propiedades(*self*)

```
Caso de prueba para el metodo descubre_propiedades
```

test_descubre_padres_hijos(*self*)

```
Este caso de prueba se encarga de que los utils descubre_padres y
descubre_hijos detectan perfectamente las entidades padre y entidades
hijas de una determinada entidad
```

test_get_model_assigned(*self*)

```
Este caso de prueba se encarga de el util get_model_assigned devuelve
correctamente el modelo con el que se corresponde una determinada
instancia
```

test_inverse_relation(*self*)

```
Este caso de prueba se encarga de que util inverse_relation devuelva
correctamente la relacion inversa a una relacion dada
```

35.6 Class TemplateTagsTestCase

django.test.TestCase
└ **project.easydata.tests.TemplateTagsTestCase**

```
Esta clase de tests, se encarga de agrupar todos los tests referentes a los
template tags implementados para la aplicacion easydata.
```

35.6.1 Methods

setUp(*self*)

```
Este metodo se encarga de preparar los datos para las pruebas
```

test_microdata(*self*)

```
Este caso de prueba se encarga de comprobar que el codigo microdata
generado por un template tag es correcto
```

test_rdfa(*self*)

```
Este caso de prueba se encarga de comprobar que el codigo rdfa generado
por un template tag es correcto
```

test_links(*self*)

```
Este caso de prueba se encarga de comprobar que el enlace generado para
una determinada instancia es correcto
```

36 Module project.easydata.urls

En este fichero se encuentran todas las urls definidas en la aplicacion EasyData junto con las vistas que tienen asociadas.

36.1 Variables

Name	Description
urlpatterns	**Value:** `patterns('easydata.views', (r'^$', 'information.welcome')...`

37 Module project.easydata.utils

En este modulo se definen una serie de funciones auxiliares, que ayudaran en las tareas de mapeo y publicacion de datos.

Author: llerena

37.1 Functions

descubre_propiedades(*entidad*, *simple*=True)

Descubre todas las propiedades de la entidad
:param entidad: es la entidad de la que se quiere conocer sus propiedades
:param simple: indica si se quieren conocer propiedades que sean simples
:return: devuelve un queryset con las propiedades

descubre_padres(*entidad*)

Descubre todas las entidades padre de la entidad indicada
:param entidad: es la entidad de la que se quiere conocer todas sus entidades padre
:return: devuelve un conjunto con todas las entidades padre

descubre_hijos(*entidad*)

Discover the entities that inherits from the provided entity

inverse_relation(*relacion*)

Casi todas las relaciones pueden recorrese en los dos sentidos, por lo que se crea una instancia de Relacion para cada uno de estos. Esta funcion dada una instancia de Relacion en un sentido, resuelve la instancia de la relacion en el sentido inverso.

get_model_assigned(*instancia*)

```
Devuelve el modelo que tiene asignado una determinada instancia
:param instancia: es la instancia de la que se desea conocer su modelo
asignado
:return: devuelve la instancia de modelo en caso de que tenga, sino None
```

get_namespaces(*model*, *visitados=[]*)

```
Devuelve los namespaces que estan relacionados con el modelo o los modelos
relacionados con este modelo, cuyas relaciones son visibles.
:param model: modelo del que se desea conocer los namespaces
:param visitados: almacena los modelos ya comprobados
:return: devuelve una lista de namespaces implicados
```

generate_unique(*model*, *clave*, *graph*, *names*)

```
Rellena el grafo graph con los datos del modelo.
:param model: es el modelo del que se va a incluir su informacion en el
grafo
:param clave: es la clave primaria de la instancia del modelo
:param graph: es el grafo donde se va a incluir la informacion
:param names: son los namespaces utilizados
:return: devuelve el grafo con la informacion incluida
```

get_header()

```
Return the url header from a request:
    protocol (http|https) + host
```

get_default_header()

```
Default method to obtain the header of the url
```

38 Package project.easydata.views

```
En este modulo se almacenan todas las vistas que se usaran en la aplicacion
EasyData, tanto para la configuracion de visibilidad, carga de namespaces,
mapeo de datos y publicacion de los mismos
```

Author: llerena

38.1 Modules

- **information**: Este modulo almacena las vistas tanto de bienvenida, como las vistas donde se...
  (Section 39, p. 75)
- **map**: En este modulo se almacenan las vistas que estan relacionadas con el mapeo de los modelos y de los fields.
  (Section 40, p. 77)
- **modelo**: En este modulo se almacena las vistas que se encargan de realizar la...
  (Section 41, p. 79)
- **namespace**: En este modulo se almacenan las vistas que gestionan los namespaces, tanto como...
  (Section 42, p. 80)
- **publish**: En este modulo se almacenan las vistas que se encargan de la publicacion de...
  (Section 43, p. 81)
- **sesiones**: Este modulo almacena las vistas tanto de bienvenida, como las vistas donde se...
  (Section 44, p. 82)

39 Module project.easydata.views.information

Este modulo almacena las vistas tanto de bienvenida, como las vistas donde se ofrece ayuda a los usuarios, indicandoles ejemplos de uso de la aplicacion tanto para urls y plantillas

Author: llerena

39.1 Functions

welcome(*request*)

This view render the welcome view

info_models(*request*)

This view render a template with information about the models of the project

info_entities(*request*)

This view render a template with information about the entities loaded in the application

info_templatetags(*request*)

This view render a template with information about the template tags availables in the application

disconnect(*request*)

Esta vista se encarga de cerrar la sesion

login(*request*)

```
Esta vista se encarga de iniciar la sesion
```

40 Module project.easydata.views.map

En este modulo se almacenan las vistas que estan relacionadas con el mapeo de los modelos y de los fields. Ademas tambien almacena aquellas vistas que estan relacionadas con el mapeo, como son las de generacion de graficos y de fichero de configuracion D2Rq

Author: llerena

40.1 Functions

mapea_modelo(*request*)

This views is used to map the existing models, with the entities of the namespaces

devuelve_entidades(*request*)

Used with ajax, return a list of entities of a given namespace

devuelve_propiedades_namespace(*request*)

Used with ajax, return a list of properties of a given namespace

devuelve_propiedades_default(*request*)

Used with ajax, return the list of properties for a concrete field

mapea_fields(*request*, *id*)

This view is used to map the fields of a concrete model, with the properties of the namespaces loaded into the application

create_configuration_graph(*request*)

```
This view is used to create a graphviz graph with the models's
configuration
```

41 Module project.easydata.views.modelo

En este modulo se almacena las vistas que se encargan de realizar la configuracion de la visibilidad de los modelos y fields

Author: llerena

41.1 Functions

configure_visibility_models(*request*, *aplicacion*)

Esta vista se encarga de realizar la configuración de visibilidad de los modelos y fields de los mismos

select_visibility_app(*request*)

Esta vista se encarga de realizar la configuración de visibilidad de los modelos y fields de los mismos

42 Module project.easydata.views.namespace

En este modulo se almacenan las vistas que gestionan los namespaces, tanto como el listado, alta de nuevo namespaces, editar un namespace existente y la eliminacion de los namespaces

Author: llerena

42.1 Functions

listado_namespaces(*request*)

This view show all the namespaces loaded into the application
:param request: the request of the view
:return: render the view

vista_carga_namespace(*request*)

This view shows a form to load a new namespace into the application.
:param request: the request of the view
:return: render the form

editar_namespace(*request*, *id*)

This view shows a form to edit the info of a namespace, and reload the entities and properties
:param request: the request of the view
:param id: the primary key of the namespace
:return: render the form

eliminar_namespace(*request*, *id*)

This view delete a concreta namespace, and all the entities and properties, and the relations between the models and fields
:param request: the request of the view
:param id: the primary key of the namespace
:return: None

43 Module project.easydata.views.publish

En este modulo se almacenan las vistas que se encargan de la publicacion de datos, ya sea para la publicacion de los datos de una instancia en concreto, de todas las instancias de un modelo y de todas las instancias de los modelos mapeados con una determinada entidad

Author: llerena

43.1 Functions

publish_model(*request*, *aplicacion*, *tipo*, *modelo*, *clave*, *format*)

Publish the information of all instances of a concrete model in the format given
:param request:the request of the view
:param format:the format to publish the data
:param aplicacion: the name of the application where is located the model
:param tipo: is the name of the entitie mapped
:param modelo: the name of the model
:return: return a file in xml, ttl or nt format, with the information

publish_type(*request*, *format*, *names*, *tipo*)

Publish the information of all instances which their models are mapped with the entity given
:param request: the request of the view
:param format: the format to publish the data (xml, nt or ttl)
:param names: is the namespace short name
:param tipo: the name of the entity
:return: return a file in xml, ttl or nt format, with the information

44 Module project.easydata.views.sesiones

Este modulo almacena las vistas tanto de bienvenida, como las vistas donde se ofrece ayuda a los usuarios, indicandoles ejemplos de uso de la aplicacion tanto para urls y plantillas

Author: llerena

44.1 Functions

disconnect(*request*)

Esta vista se encarga de cerrar la sesion

login(*request*)

Esta vista se encarga de iniciar la sesion

Index