

---

# **Highton Documentation**

***Release 2.0.0***

**Michael Bykovski, Julia Giebelhausen, Jakob Löhnertz**

**Jul 30, 2018**



<b>1</b>	<b>Quickstart</b>	<b>1</b>
1.1	Install . . . . .	1
1.2	Setup . . . . .	1
1.3	Retrieve data . . . . .	1
1.4	Get a single object . . . . .	1
1.5	Create data . . . . .	2
1.6	Update data . . . . .	2
1.7	Delete data . . . . .	2
<b>2</b>	<b>Models</b>	<b>3</b>
2.1	Address . . . . .	3
2.2	AssociatedParty . . . . .	4
2.3	Attachment . . . . .	5
2.4	Case . . . . .	6
2.5	Category . . . . .	9
2.6	Comment . . . . .	10
2.7	Company . . . . .	11
2.8	Contact . . . . .	15
2.9	ContactData . . . . .	16
2.10	Deal . . . . .	17
2.11	DealCategory . . . . .	21
2.12	Email . . . . .	22
2.13	EmailAddress . . . . .	24
2.14	Group . . . . .	25
2.15	HightonModel . . . . .	27
2.16	InstantMessenger . . . . .	27
2.17	Note . . . . .	28
2.18	Party . . . . .	30
2.19	Person . . . . .	32
2.20	PhoneNumber . . . . .	35
2.21	SubjectData . . . . .	36
2.22	Tag . . . . .	37
2.23	Task . . . . .	38
2.24	TaskCategory . . . . .	40
2.25	User . . . . .	42
2.26	TwitterAccount . . . . .	44

2.27	WebAddress . . . . .	44
<b>3</b>	<b>Fields</b>	<b>47</b>
3.1	BooleanField . . . . .	47
3.2	DateField . . . . .	48
3.3	DateTimeField . . . . .	48
3.4	Field . . . . .	49
3.5	IntegerField . . . . .	49
3.6	ListField . . . . .	50
3.7	ObjectField . . . . .	50
3.8	StringField . . . . .	51
<b>4</b>	<b>Indices and tables</b>	<b>53</b>
	<b>Python Module Index</b>	<b>55</b>

### 1.1 Install

```
pip install highton
```

### 1.2 Setup

Just setup highton by defining a singleton in your application. You don't have to store this singleton in a variable.

```
from highton.highton_settings import HightonSettings  
  
HightonSettings(username='<your_username>', api_key='<your_API_KEY>')
```

### 1.3 Retrieve data

If you have defined the 'USERNAME' and 'API\_KEY', you can start to retrieve data:

```
from highton.models import Deal  
  
deals = Deal.list()
```

### 1.4 Get a single object

```
from highton.models import Deal  
  
deal_object = Deal.get(12345)
```

## 1.5 Create data

```
from highton.models import Deal

new_deal_object = Deal(name='new deal').create()
```

## 1.6 Update data

```
from highton.models import Deal

deal_object = Deal.get(12345)
deal_object.name = 'new name'
deal_object.update()
```

## 1.7 Delete data

```
from highton.models import Deal

deal_object = Deal.get(12345)
deal_object.delete()
```

## 2.1 Address

```
class highton.models.address.Address (**kwargs)
```

### Variables

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **location** – fields.StringField(name=HightonConstants.LOCATION)
- **city** – fields.StringField(name=HightonConstants.CITY)
- **country** – fields.StringField(name=HightonConstants.COUNTRY)
- **state** – fields.StringField(name=HightonConstants.STATE)
- **street** – fields.StringField(name=HightonConstants.STREET)
- **zip** – fields.StringField(name=HightonConstants.ZIP)

```
ENDPOINT = None
```

```
TAG_NAME = 'address'
```

```
classmethod decode (root_element)
```

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

```
static element_from_string (string)
```

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string (element)**  
Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode ()**  
Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**to\_serializable\_value ()**  
Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

## 2.2 AssociatedParty

```
class highton.models.associated_party.AssociatedParty (**kwargs)
```

### Variables

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **author\_id** – fields.IntegerField(name=HightonConstants.AUTHOR\_ID)
- **background** – fields.StringField(name=HightonConstants.BACKGROUND)
- **company\_id** – fields.IntegerField(name=HightonConstants.COMPANY\_ID)
- **created\_at** – fields.DatetimeField(name=HightonConstants.CREATED\_AT)
- **first\_name** – fields.StringField(name=HightonConstants.FIRST\_NAME)
- **name** – fields.StringField(name=HightonConstants.NAME)
- **group\_id** – fields.IntegerField(name=HightonConstants.GROUP\_ID)
- **last\_name** – fields.StringField(name=HightonConstants.LAST\_NAME)
- **owner\_id** – fields.IntegerField(name=HightonConstants.OWNER\_ID)
- **title** – fields.StringField(name=HightonConstants.TITLE)
- **updated\_at** – fields.DatetimeField(name=HightonConstants.UPDATED\_AT)
- **visible\_to** – fields.StringField(name=HightonConstants.VISIBLE\_TO)
- **company\_name** – fields.StringField(name=HightonConstants.COMPANY\_NAME)
- **linkedin\_url** – fields.StringField(name=HightonConstants.LINKEDIN\_URL)
- **avatar\_url** – fields.StringField(name=HightonConstants.AVATAR\_URL)
- **type** – fields.StringField(name=HightonConstants.TYPE)
- **tags** – fields.ListField(name=HightonConstants.TAGS, init\_class=Tag)



- **contact\_data** – fields.ObjectField(name=HightonConstants.CONTACT\_DATA, init\_class=ContactData)
- **subject\_datas** – fields.ListField(name=HightonConstants.SUBJECT\_DATAS, init\_class=SubjectData)

**ENDPOINT** = 'parties'

**TAG\_NAME** = 'associated-party'

**classmethod decode** (*root\_element*)  
Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static element\_from\_string** (*string*)  
Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)  
Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode** ()  
Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**to\_serializable\_value** ()  
Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

## 2.3 Attachment

**class** highton.models.attachment.**Attachment** (\*\*kwargs)

**Variables**

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **url** – fields.StringField(name=HightonConstants.URL)
- **name** – fields.StringField(name=HightonConstants.NAME)

- **size** – fields.IntegerField(name=HightonConstants.SIZE)

```
COLLECTION_DATETIME = '%Y%m%d%H%M%S'
```

```
ENDPOINT = None
```

```
TAG_NAME = 'attachment'
```

```
classmethod decode (root_element)
```

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

```
static element_from_string (string)
```

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

```
static element_to_string (element)
```

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

```
encode ()
```

Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

```
get ()
```

```
to_serializable_value ()
```

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

## 2.4 Case

```
class highton.models.case.Case (**kwargs)
```

### Variables

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **author\_id** – fields.IntegerField(name=HightonConstants.AUTHOR\_ID)
- **closed\_at** – fields.DatetimeField(name=HightonConstants.CLOSED\_AT)
- **created\_at** – fields.DatetimeField(name=HightonConstants.CREATED\_AT)

- **updated\_at** – fields.DatetimeField(name=HightonConstants.UPDATED\_AT)
- **name** – fields.StringField(name=HightonConstants.NAME)
- **visible\_to** – fields.StringField(name=HightonConstants.VISIBLE\_TO)
- **group\_id** – fields.IntegerField(name=HightonConstants.GROUP\_ID)
- **owner\_id** – fields.IntegerField(name=HightonConstants.OWNER\_ID)
- **background** – fields.StringField(name=HightonConstants.BACKGROUND)
- **parties** – fields.ListField(name=HightonConstants.PARTIES, init\_class=Party)
- **associated\_parties** – fields.ListField(name=HightonConstants.ASSOCIATED\_PARTIES, init\_class=AssociatedParty)

**COLLECTION\_DATETIME** = '%Y%m%d%H%M%S'

**EMAILS\_OFFSET** = 25

**ENDPOINT** = 'kases'

**NOTES\_OFFSET** = 25

**TAG\_NAME** = 'kase'

**add\_note** (*body*)

Create a Note to current object

**Parameters** **body** (*str*) – the body of the note

**Returns** newly created Note

**Return type** *Tag*

**add\_tag** (*name*)

Create a Tag to current object

**Parameters** **name** (*str*) – the name of the tag

**Returns** newly created Tag

**Return type** *Tag*

**create** ()

Creates the object

**Returns** the created object

**Return type** object

**classmethod decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**delete** ()

Deletes the object

**Returns**

**Return type** None

**delete\_tag** (*tag\_id*)

Deletes a Tag to current object

**Parameters** **tag\_id** (*int*) – the id of the tag which should be deleted

**Return type** None

**static element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode** ()

Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**classmethod get** (*object\_id*)

Retrieves a single model

**Parameters** **object\_id** (*integer*) – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**classmethod list** (*params=None*)

Retrieves a list of the model

**Parameters** **params** (*dict*) – params as dictionary

**Returns** the list of the parsed xml objects

**Return type** list

**list\_emails** (*page=0, since=None*)

Get the emails of current object

**Parameters**

- **page** – the page starting at 0
- **since** (*datetime.datetime*) – get all notes since a datetime

**Returns** the emails

**Return type** list

**list\_notes** (*page=0, since=None*)

Get the notes of current object

**Parameters**

- **page** – the page starting at 0

- **since** (*datetime.datetime*) – get all notes since a datetime

**Returns** the notes

**Return type** list

**list\_tags** ()

Get the tags of current object

**Returns** the tags

**Return type** list

**list\_tasks** ()

Get the tasks of current object

**Returns** the tasks

**Return type** list

**to\_serializable\_value** ()

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

**update** ()

Updates the object

**Returns**

**Return type** response

## 2.5 Category

```
class highton.models.category.Category (**kwargs)
```

**Variables**

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **name** – fields.StringField(name=HightonConstants.NAME)

**ENDPOINT** = None

**TAG\_NAME** = 'category'

**classmethod decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** `xml.etree.ElementTree.Element`

**static element\_to\_string** (*element*)  
Parses the `xml.etree.ElementTree.Element` to a string

**Parameters** *element* (`xml.etree.ElementTree.Element`) – a xml element

**Returns** the parsed string

**Return type** `str`

**encode** ()  
Encodes the object to a `xml.etree.ElementTree.Element`

**Returns** the encoded element

**Return type** `xml.etree.ElementTree.Element`

**to\_serializable\_value** ()  
Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** `dict`

## 2.6 Comment

```
class highton.models.comment.Comment (**kwargs)
```

### Variables

- **id** – `fields.IntegerField(name=HightonConstants.ID)`
- **parent\_id** – `fields.IntegerField(name=HightonConstants.PARENT_ID)`
- **author\_id** – `fields.IntegerField(name=HightonConstants.AUTHOR_ID)`
- **created\_at** – `fields.DateTimeField(name=HightonConstants.CREATED_AT)`
- **body** – `fields.StringField(name=HightonConstants.BODY)`

```
COLLECTION_DATETIME = '%Y%m%d%H%M%S'
```

```
ENDPOINT = 'comments'
```

```
TAG_NAME = 'comment'
```

```
create ()
```

Creates the object

**Returns** the created object

**Return type** `object`

```
classmethod decode (root_element)
```

Decode the object to the object

**Parameters** *root\_element* (`xml.etree.ElementTree.Element`) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** `object`

**delete()**  
 Deletes the object

**Returns**

**Return type** None

**static element\_from\_string(string)**  
 Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string(element)**  
 Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode()**  
 Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**classmethod get(object\_id)**  
 Retrieves a single model

**Parameters** **object\_id** (*integer*) – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**to\_serializable\_value()**  
 Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

**update()**  
 Updates the object

**Returns**

**Return type** response

## 2.7 Company

```
class highton.models.company.Company(**kwargs)
```

**Variables**

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **author\_id** – fields.IntegerField(name=HightonConstants.AUTHOR\_ID)

- **background** – fields.StringField(name=HightonConstants.BACKGROUND)
- **created\_at** – fields.DateTimeField(name=HightonConstants.CREATED\_AT)
- **group\_id** – fields.IntegerField(name=HightonConstants.GROUP\_ID)
- **owner\_id** – fields.IntegerField(name=HightonConstants.OWNER\_ID)
- **updated\_at** – fields.DateTimeField(name=HightonConstants.UPDATED\_AT)
- **visible\_to** – fields.StringField(name=HightonConstants.VISIBLE\_TO)
- **linkedin\_url** – fields.StringField(name=HightonConstants.LINKEDIN\_URL)
- **avatar\_url** – fields.StringField(name=HightonConstants.AVATAR\_URL)
- **tags** – fields.ListField(name=HightonConstants.TAGS, init\_class=Tag)
- **contact\_data** – fields.ObjectField(name=HightonConstants.CONTACT\_DATA, init\_class=ContactData)
- **subject\_datas** – fields.ListField(name=HightonConstants.SUBJECT\_DATAS, init\_class=SubjectData)
- **name** – fields.StringField(name=HightonConstants.NAME)

**COLLECTION\_DATETIME** = '%Y%m%d%H%M%S'

**EMAILS\_OFFSET** = 25

**ENDPOINT** = 'companies'

**NOTES\_OFFSET** = 25

**OFFSET** = 500

**SEARCH\_OFFSET** = 25

**TAG\_NAME** = 'company'

**add\_note** (*body*)

Create a Note to current object

**Parameters** **body** (*str*) – the body of the note

**Returns** newly created Note

**Return type** *Tag*

**add\_tag** (*name*)

Create a Tag to current object

**Parameters** **name** (*str*) – the name of the tag

**Returns** newly created Tag

**Return type** *Tag*

**create** ()

Creates the object

**Returns** the created object

**Return type** object

**classmethod decode** (*root\_element*)

Decode the object to the object



**Parameters** `root_element` (`xml.etree.ElementTree.Element`) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**delete()**  
Deletes the object

**Returns**

**Return type** None

**delete\_tag** (`tag_id`)  
Deletes a Tag to current object

**Parameters** `tag_id` (`int`) – the id of the tag which should be deleted

**Return type** None

**static element\_from\_string** (`string`)  
Make a Element from a str

**Parameters** `string` (`str`) – string you want to parse

**Returns** the parsed xml string

**Return type** `xml.etree.ElementTree.Element`

**static element\_to\_string** (`element`)  
Parses the `xml.etree.ElementTree.Element` to a string

**Parameters** `element` (`xml.etree.ElementTree.Element`) – a xml element

**Returns** the parsed string

**Return type** `str`

**encode()**  
Encodes the object to a `xml.etree.ElementTree.Element`

**Returns** the encoded element

**Return type** `xml.etree.ElementTree.Element`

**classmethod get** (`object_id`)  
Retrieves a single model

**Parameters** `object_id` (`integer`) – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**classmethod list** (`page=0, since=None, tag_id=None, title=None`)

**Parameters**

- **page** (`int`) – page starting by 0
- **since** (`datetime.datetime`) –
- **tag\_id** (`int`) – id of a tag
- **title** (`str`) –

**Returns** list of customer objects

**Return type** list

**list\_emails** (*page=0, since=None*)

Get the emails of current object

**Parameters**

- **page** – the page starting at 0
- **since** (*datetime.datetime*) – get all notes since a datetime

**Returns** the emails

**Return type** list

**list\_notes** (*page=0, since=None*)

Get the notes of current object

**Parameters**

- **page** – the page starting at 0
- **since** (*datetime.datetime*) – get all notes since a datetime

**Returns** the notes

**Return type** list

**list\_tags** ()

Get the tags of current object

**Returns** the tags

**Return type** list

**list\_tasks** ()

Get the tasks of current object

**Returns** the tasks

**Return type** list

**people** ()

Retrieve all people of the company

**Returns** list of people objects

**Return type** list

**classmethod search** (*term=None, page=0, \*\*criteria*)

Search a list of the model If you use “term”: - Returns a collection of people that have a name matching the term passed in through the URL.

If you use “criteria”: - returns people who match your search criteria. Search by any criteria you can on the Contacts tab, including custom fields. Combine criteria to narrow results

**Parameters**

- **term** (*str*) – params as string
- **criteria** (*dict*) – search for more criteria
- **page** (*int*) – the page

**Returns** the list of the parsed xml objects

**Return type** list

**to\_serializable\_value()**

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

**update()**

Updates the object

**Returns**

**Return type** response

## 2.8 Contact

**class** highton.models.contact.**Contact** (\*\*kwargs)

**Variables**

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **author\_id** – fields.IntegerField(name=HightonConstants.AUTHOR\_ID)
- **background** – fields.StringField(name=HightonConstants.BACKGROUND)
- **created\_at** – fields.DateTimeField(name=HightonConstants.CREATED\_AT)
- **group\_id** – fields.IntegerField(name=HightonConstants.GROUP\_ID)
- **owner\_id** – fields.IntegerField(name=HightonConstants.OWNER\_ID)
- **updated\_at** – fields.DateTimeField(name=HightonConstants.UPDATED\_AT)
- **visible\_to** – fields.StringField(name=HightonConstants.VISIBLE\_TO)
- **linkedin\_url** – fields.StringField(name=HightonConstants.LINKEDIN\_URL)
- **avatar\_url** – fields.StringField(name=HightonConstants.AVATAR\_URL)
- **tags** – fields.ListField(name=HightonConstants.TAGS, init\_class=Tag)
- **contact\_data** – fields.ObjectField(name=HightonConstants.CONTACT\_DATA, init\_class=ContactData)
- **subject\_datas** – fields.ListField(name=HightonConstants.SUBJECT\_DATAS, init\_class=SubjectData)

**ENDPOINT** = None

**TAG\_NAME** = None

**classmethod** **decode** (root\_element)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static** **element\_from\_string** (string)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode** ()

Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**to\_serializable\_value** ()

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

## 2.9 ContactData

```
class highton.models.contact_data.ContactData (**kwargs)
```

### Variables

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **phone\_numbers** – fields.ListField(name=HightonConstants.PHONE\_NUMBERS, init\_class=PhoneNumber)
- **twitter\_accounts** – fields.ListField(name=HightonConstants.TWITTER\_ACCOUNTS, init\_class=TwitterAccount)
- **web\_addresses** – fields.ListField(name=HightonConstants.WEB\_ADDRESSES, init\_class=WebAddress)
- **email\_addresses** – fields.ListField(name=HightonConstants.EMAIL\_ADDRESSES, init\_class=EmailAddress)
- **addresses** – fields.ListField(name=HightonConstants.ADDRESSES, init\_class=Address)
- **instant\_messenger** – fields.ListField(name=HightonConstants.INSTANT\_MESSENGERS, init\_class=InstantMessenger)

**ENDPOINT** = None

**TAG\_NAME** = 'contact-data'

**classmethod decode** (*root\_element*)

Decode the object to the object

**Parameters** `root_element` (`xml.etree.ElementTree.Element`) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static** `element_from_string` (`string`)  
Make a Element from a str

**Parameters** `string` (`str`) – string you want to parse

**Returns** the parsed xml string

**Return type** `xml.etree.ElementTree.Element`

**static** `element_to_string` (`element`)  
Parses the `xml.etree.ElementTree.Element` to a string

**Parameters** `element` (`xml.etree.ElementTree.Element`) – a xml element

**Returns** the parsed string

**Return type** `str`

**encode** ()  
Encodes the object to a `xml.etree.ElementTree.Element`

**Returns** the encoded element

**Return type** `xml.etree.ElementTree.Element`

**to\_serializable\_value** ()  
Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

## 2.10 Deal

**class** `highton.models.deal.Deal` (`**kwargs`)

A deal which represents:

<https://github.com/basecamp/highrise-api/blob/master/sections/deals.md>

### Variables

- `id` – `fields.IntegerField(name=HightonConstants.ID)`
- `author_id` – `fields.IntegerField(name=HightonConstants.AUTHOR_ID)`
- `account_id` – `fields.IntegerField(name=HightonConstants.ACCOUNT_ID)`
- `background` – `fields.StringField(name=HightonConstants.BACKGROUND)`
- `category_id` – `fields.IntegerField(name=HightonConstants.CATEGORY_ID)`
- `created_at` – `fields.DateTimeField(name=HightonConstants.CREATED_AT)`
- `currency` – `fields.StringField(name=HightonConstants.CURRENCY)`
- `duration` – `fields.IntegerField(name=HightonConstants.DURATION)`
- `group_id` – `fields.IntegerField(name=HightonConstants.GROUP_ID)`

- **name** – fields.StringField(name=HightonConstants.NAME)
- **owner\_id** – fields.IntegerField(name=HightonConstants.OWNER\_ID)
- **party\_id** – fields.IntegerField(name=HightonConstants.PARTY\_ID)
- **price** – fields.IntegerField(name=HightonConstants.PRICE)
- **price\_type** – fields.StringField(name=HightonConstants.PRICE\_TYPE)
- **responsible\_party\_id** – fields.IntegerField(name=HightonConstants.RESPONSIBLE\_PARTY\_ID)
- **status** – fields.StringField(name=HightonConstants.STATUS)
- **status\_changed\_on** – fields.DateField(name=HightonConstants.STATUS\_CHANGED\_ON)
- **updated\_at** – fields.DateTimeField(name=HightonConstants.UPDATED\_AT)
- **visible\_to** – fields.StringField(name=HightonConstants.VISIBLE\_TO)
- **party** – fields.ObjectField(name=HightonConstants.PARTY, init\_class=Party)
- **category** – fields.ObjectField(name=HightonConstants.CATEGORY, init\_class=Category)
- **tags** – fields.ListField(name=HightonConstants.TAGS, init\_class=Tag)
- **parties** – fields.ListField(name=HightonConstants.PARTIES, init\_class=Party)
- **contact\_data** – fields.ObjectField(name=HightonConstants.CONTACT\_DATA, init\_class=ContactData)
- **subject\_datas** – fields.ListField(name=HightonConstants.SUBJECT\_DATAS, init\_class=SubjectData)
- **associated\_parties** – fields.ListField(name=HightonConstants.ASSOCIATED\_PARTIES, init\_class=AssociatedParty)

**COLLECTION\_DATETIME** = '%Y%m%d%H%M%S'

**EMAILS\_OFFSET** = 25

**ENDPOINT** = 'deals'

**NOTES\_OFFSET** = 25

**OFFSET** = 500

**TAG\_NAME** = 'deal'

**add\_note** (*body*)

Create a Note to current object

**Parameters** **body** (*str*) – the body of the note

**Returns** newly created Note

**Return type** *Tag*

**add\_tag** (*name*)

Create a Tag to current object

**Parameters** **name** (*str*) – the name of the tag

**Returns** newly created Tag

**Return type** *Tag*

**create()**  
Creates the object

**Returns** the created object

**Return type** object

**classmethod decode(*root\_element*)**  
Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**delete()**  
Deletes the object

**Returns**

**Return type** None

**delete\_tag(*tag\_id*)**  
Deletes a Tag to current object

**Parameters** **tag\_id** (*int*) – the id of the tag which should be deleted

**Return type** None

**static element\_from\_string(*string*)**  
Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string(*element*)**  
Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode()**  
Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**classmethod get(*object\_id*)**  
Retrieves a single model

**Parameters** **object\_id** (*integer*) – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**classmethod list(*page=0, status=None, since=None*)**

**Parameters**

- **page** (*int*) – page starting by 0
- **since** (*datetime.datetime*) –
- **status** (*str*) –

**Returns** list of person objects

**Return type** list

**list\_emails** (*page=0, since=None*)

Get the emails of current object

**Parameters**

- **page** – the page starting at 0
- **since** (*datetime.datetime*) – get all notes since a datetime

**Returns** the emails

**Return type** list

**list\_notes** (*page=0, since=None*)

Get the notes of current object

**Parameters**

- **page** – the page starting at 0
- **since** (*datetime.datetime*) – get all notes since a datetime

**Returns** the notes

**Return type** list

**list\_tags** ()

Get the tags of current object

**Returns** the tags

**Return type** list

**list\_tasks** ()

Get the tasks of current object

**Returns** the tasks

**Return type** list

**to\_serializable\_value** ()

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

**update** ()

Updates the object

**Returns**

**Return type** response

**update\_status** (*status*)

Updates the status of the deal

**Parameters** **status** – status have to be ('won', 'pending', 'lost')



**Returns** successfull response or raise Exception

**Return type**

## 2.11 DealCategory

```
class highton.models.deal_category.DealCategory (**kwargs)
```

### Variables

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **name** – fields.StringField(name=HightonConstants.NAME)
- **color** – fields.StringField(name=HightonConstants.COLOR)
- **account\_id** – fields.IntegerField(name=HightonConstants.ACCOUNT\_ID)
- **created\_at** – fields.DatetimeField(name=HightonConstants.CREATED\_AT)
- **updated\_at** – fields.DatetimeField(name=HightonConstants.UPDATED\_AT)
- **elements\_count** – fields.IntegerField(name=HightonConstants.ELEMENTS\_COUNT)

```
COLLECTION_DATETIME = '%Y%m%d%H%M%S'
```

```
ENDPOINT = 'deal_categories'
```

```
TAG_NAME = 'deal-category'
```

```
create()
```

Creates the object

**Returns** the created object

**Return type** object

```
classmethod decode(root_element)
```

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

```
delete()
```

Deletes the object

**Returns**

**Return type** None

```
static element_from_string(string)
```

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

```
static element_to_string(element)
```

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode()**

Encodes the object to a *xml.etree.ElementTree.Element*

**Returns** the encoded element

**Return type** *xml.etree.ElementTree.Element*

**classmethod** **get** (*object\_id*)

Retrieves a single model

**Parameters** **object\_id** (*integer*) – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**classmethod** **list** (*params=None*)

Retrieves a list of the model

**Parameters** **params** (*dict*) – params as dictionary

**Returns** the list of the parsed xml objects

**Return type** list

**to\_serializable\_value()**

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

**update()**

Updates the object

**Returns**

**Return type** response

## 2.12 Email

**class** *highton.models.email.Email* (*\*\*kwargs*)

### Variables

- **id** – *fields.IntegerField(name=HightonConstants.ID)*
- **title** – *fields.StringField(name=HightonConstants.TITLE, required=True)*
- **body** – *fields.StringField(name=HightonConstants.BODY, required=True)*
- **subject\_id** – *fields.IntegerField(name=HightonConstants.SUBJECT\_ID, required=True)*
- **subject\_type** – *fields.StringField(name=HightonConstants.SUBJECT\_TYPE, required=True)*
- **subject\_name** – *fields.StringField(name=HightonConstants.SUBJECT\_NAME)*

- **author\_id** – fields.IntegerField(name=HightonConstants.AUTHOR\_ID)
- **collection\_id** – fields.IntegerField(name=HightonConstants.COLLECTION\_ID)
- **collection\_type** – fields.StringField(name=HightonConstants.COLLECTION\_TYPE)
- **visible\_to** – fields.StringField(name=HightonConstants.VISIBLE\_TO)
- **owner\_id** – fields.IntegerField(name=HightonConstants.OWNER\_ID)
- **group\_id** – fields.IntegerField(name=HightonConstants.GROUP\_ID)
- **updated\_at** – fields.DateTimeField(name=HightonConstants.UPDATED\_AT)
- **created\_at** – fields.DateTimeField(name=HightonConstants.CREATED\_AT)
- **attachments** – fields.ListField(name=HightonConstants.ATTACHMENTS, init\_class=Attachment)

**COLLECTION\_DATETIME** = '%Y%m%d%H%M%S'

**COMMENT\_OFFSET** = 25

**ENDPOINT** = 'emails'

**TAG\_NAME** = 'email'

**create()**

Creates the object

**Returns** the created object

**Return type** object

**classmethod decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**delete()**

Deletes the object

**Returns**

**Return type** None

**static element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode()**  
Encodes the object to a `xml.etree.ElementTree.Element`

**Returns** the encoded element

**Return type** `xml.etree.ElementTree.Element`

**classmethod get(object\_id)**  
Retrieves a single model

**Parameters** **object\_id** (*integer*) – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**list\_comments(page=0)**  
Get the comments of current object

**Parameters** **page** – the page starting at 0

**Returns** the emails

**Return type** list

**to\_serializable\_value()**  
Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

**update()**  
Updates the object

**Returns**

**Return type** response

## 2.13 EmailAddress

```
class highton.models.email_address.EmailAddress(**kwargs)
```

### Variables

- **id** – `fields.IntegerField(name=HightonConstants.ID)`
- **location** – `fields.StringField(name=HightonConstants.LOCATION)`
- **address** – `fields.StringField(name=HightonConstants.ADDRESS)`

**ENDPOINT** = `None`

**TAG\_NAME** = `'email-address'`

**classmethod decode(root\_element)**

Decode the object to the object

**Parameters** **root\_element** (`xml.etree.ElementTree.Element`) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode** ()

Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**to\_serializable\_value** ()

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

## 2.14 Group

**class** highton.models.group.**Group** (*\*\*kwargs*)

The authenticated user needs to be an administrator to perform these actions.

**Variables**

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **name** – fields.StringField(name=HightonConstants.NAME)
- **users** – fields.ListField(name=HightonConstants.USERS, init\_class=models.User)

**COLLECTION\_DATETIME** = '%Y%m%d%H%M%S'

**ENDPOINT** = 'groups'

**TAG\_NAME** = 'group'

**create** ()

Creates the object

**Returns** the created object

**Return type** object

**classmethod decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**delete** ()

Deletes the object

**Returns**

**Return type** None

**static element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode** ()

Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**classmethod get** (*object\_id*)

Retrieves a single model

**Parameters** **object\_id** (*integer*) – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**classmethod list** (*params=None*)

Retrieves a list of the model

**Parameters** **params** (*dict*) – params as dictionary

**Returns** the list of the parsed xml objects

**Return type** list

**to\_serializable\_value** ()

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

**update** ()

Updates the object

**Returns**

**Return type** response

## 2.15 HightonModel

**class** `highton.models.highton_model.HightonModel` (*\*\*kwargs*)

This class inherit from XMLDecoder and XMLEncoder which allows every HightonModel to be encoded and decoded in xml.

**Variables** `id` – `fields.IntegerField(name=HightonConstants.ID)`

**ENDPOINT** = `None`

**TAG\_NAME** = `None`

**classmethod** `decode` (*root\_element*)

Decode the object to the object

**Parameters** `root_element` (`xml.etree.ElementTree.Element`) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static** `element_from_string` (*string*)

Make a Element from a str

**Parameters** `string` (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** `xml.etree.ElementTree.Element`

**static** `element_to_string` (*element*)

Parses the `xml.etree.ElementTree.Element` to a string

**Parameters** `element` (`xml.etree.ElementTree.Element`) – a xml element

**Returns** the parsed string

**Return type** `str`

**encode** ()

Encodes the object to a `xml.etree.ElementTree.Element`

**Returns** the encoded element

**Return type** `xml.etree.ElementTree.Element`

**to\_serializable\_value** ()

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** `dict`

## 2.16 InstantMessenger

**class** `highton.models.instant_messenger.InstantMessenger` (*\*\*kwargs*)

**Variables**

- `id` – `fields.IntegerField(name=HightonConstants.ID)`
- `location` – `fields.StringField(name=HightonConstants.LOCATION)`

- **address** – fields.StringField(name=HightonConstants.ADDRESS)
- **protocol** – fields.StringField(name=HightonConstants.PROTOCOL)

**ENDPOINT** = None

**TAG\_NAME** = 'instant-messenger'

**classmethod decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode** ()

Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**to\_serializable\_value** ()

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

## 2.17 Note

**class** highton.models.note.**Note** (*\*\*kwargs*)

**Variables**

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **body** – fields.StringField(name=HightonConstants.BODY, required=True)
- **author\_id** – fields.IntegerField(name=HightonConstants.AUTHOR\_ID)
- **subject\_id** – fields.IntegerField(name=HightonConstants.SUBJECT\_ID, required=True)



- **subject\_type** – fields.StringField(name=HightonConstants.SUBJECT\_TYPE, required=True)
- **subject\_name** – fields.StringField(name=HightonConstants.SUBJECT\_NAME)
- **collection\_id** – fields.IntegerField(name=HightonConstants.COLLECTION\_ID)
- **collection\_type** – fields.StringField(name=HightonConstants.COLLECTION\_TYPE)
- **visible\_to** – fields.StringField(name=HightonConstants.VISIBLE\_TO)
- **owner\_id** – fields.IntegerField(name=HightonConstants.OWNER\_ID)
- **group\_id** – fields.IntegerField(name=HightonConstants.GROUP\_ID)
- **updated\_at** – fields.DateTimeField(name=HightonConstants.UPDATED\_AT)
- **created\_at** – fields.DateTimeField(name=HightonConstants.CREATED\_AT)
- **attachments** – fields.ListField(name=HightonConstants.ATTACHMENTS, init\_class=Attachment)

**COLLECTION\_DATETIME** = '%Y%m%d%H%M%S'

**COMMENT\_OFFSET** = 25

**ENDPOINT** = 'notes'

**TAG\_NAME** = 'note'

**create()**

Creates the object

**Returns** the created object

**Return type** object

**classmethod decode(*root\_element*)**

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**delete()**

Deletes the object

**Returns**

**Return type** None

**static element\_from\_string(*string*)**

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string(*element*)**

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode()**  
Encodes the object to a `xml.etree.ElementTree.Element`

**Returns** the encoded element

**Return type** `xml.etree.ElementTree.Element`

**classmethod get(object\_id)**  
Retrieves a single model

**Parameters** **object\_id** (*integer*) – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**list\_comments(page=0)**  
Get the comments of current object

**Parameters** **page** – the page starting at 0

**Returns** the emails

**Return type** list

**to\_serializable\_value()**  
Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

**update()**  
Updates the object

**Returns**

**Return type** response

## 2.18 Party

```
class highton.models.party.Party(**kwargs)
```

### Variables

- **id** – `fields.IntegerField(name=HightonConstants.ID)`
- **author\_id** – `fields.IntegerField(name=HightonConstants.AUTHOR_ID)`
- **background** – `fields.StringField(name=HightonConstants.BACKGROUND)`
- **company\_id** – `fields.IntegerField(name=HightonConstants.COMPANY_ID)`
- **created\_at** – `fields.DatetimeField(name=HightonConstants.CREATED_AT)`
- **first\_name** – `fields.StringField(name=HightonConstants.FIRST_NAME)`
- **name** – `fields.StringField(name=HightonConstants.NAME)`
- **group\_id** – `fields.IntegerField(name=HightonConstants.GROUP_ID)`
- **last\_name** – `fields.StringField(name=HightonConstants.LAST_NAME)`

- **owner\_id** – fields.IntegerField(name=HightonConstants.OWNER\_ID)
- **title** – fields.StringField(name=HightonConstants.TITLE)
- **updated\_at** – fields.DateTimeField(name=HightonConstants.UPDATED\_AT)
- **visible\_to** – fields.StringField(name=HightonConstants.VISIBLE\_TO)
- **company\_name** – fields.StringField(name=HightonConstants.COMPANY\_NAME)
- **linkedin\_url** – fields.StringField(name=HightonConstants.LINKEDIN\_URL)
- **avatar\_url** – fields.StringField(name=HightonConstants.AVATAR\_URL)
- **type** – fields.StringField(name=HightonConstants.TYPE)
- **tags** – fields.ListField(name=HightonConstants.TAGS, init\_class=Tag)
- **contact\_data** – fields.ObjectField(name=HightonConstants.CONTACT\_DATA, init\_class=ContactData)
- **subject\_datas** – fields.ListField(name=HightonConstants.SUBJECT\_DATAS, init\_class=SubjectData)

**ENDPOINT** = 'parties'

**TAG\_NAME** = 'party'

**classmethod decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode** ()

Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**to\_serializable\_value** ()

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

Return type dict

## 2.19 Person

```
class highton.models.person.Person (**kwargs)
```

### Variables

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **company\_id** – fields.IntegerField(name=HightonConstants.COMPANY\_ID)
- **company\_name** – fields.StringField(name=HightonConstants.COMPANY\_NAME)
- **first\_name** – fields.StringField(name=HightonConstants.FIRST\_NAME)
- **last\_name** – fields.StringField(name=HightonConstants.LAST\_NAME)
- **title** – fields.StringField(name=HightonConstants.TITLE)
- **author\_id** – fields.IntegerField(name=HightonConstants.AUTHOR\_ID)
- **background** – fields.StringField(name=HightonConstants.BACKGROUND)
- **created\_at** – fields.DateTimeField(name=HightonConstants.CREATED\_AT)
- **group\_id** – fields.IntegerField(name=HightonConstants.GROUP\_ID)
- **owner\_id** – fields.IntegerField(name=HightonConstants.OWNER\_ID)
- **updated\_at** – fields.DateTimeField(name=HightonConstants.UPDATED\_AT)
- **visible\_to** – fields.StringField(name=HightonConstants.VISIBLE\_TO)
- **linkedin\_url** – fields.StringField(name=HightonConstants.LINKEDIN\_URL)
- **avatar\_url** – fields.StringField(name=HightonConstants.AVATAR\_URL)
- **tags** – fields.ListField(name=HightonConstants.TAGS, init\_class=Tag)
- **contact\_data** – fields.ObjectField(name=HightonConstants.CONTACT\_DATA, init\_class=ContactData)
- **subject\_datas** – fields.ListField(name=HightonConstants.SUBJECT\_DATAS, init\_class=SubjectData)

```
COLLECTION_DATETIME = '%Y%m%d%H%M%S'
```

```
EMAILS_OFFSET = 25
```

```
ENDPOINT = 'people'
```

```
NOTES_OFFSET = 25
```

```
OFFSET = 500
```

```
SEARCH_OFFSET = 25
```

```
TAG_NAME = 'person'
```

```
add_note(body)
```

Create a Note to current object

**Parameters** **body** (*str*) – the body of the note

**Returns** newly created Note

**Return type** *Tag*

**add\_tag** (*name*)

Create a Tag to current object

**Parameters** **name** (*str*) – the name of the tag

**Returns** newly created Tag

**Return type** *Tag*

**create** ()

Creates the object

**Returns** the created object

**Return type** object

**classmethod decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**delete** ()

Deletes the object

**Returns**

**Return type** None

**delete\_tag** (*tag\_id*)

Deletes a Tag to current object

**Parameters** **tag\_id** (*int*) – the id of the tag which should be deleted

**Return type** None

**static element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode** ()

Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**classmethod** `get (object_id)`

Retrieves a single model

**Parameters** `object_id (integer)` – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**classmethod** `list (page=0, since=None, tag_id=None, title=None)`

**Parameters**

- **page** (*int*) – page starting by 0
- **since** (*datetime.datetime*) –
- **tag\_id** (*int*) – id of a tag
- **title** (*str*) –

**Returns** list of person objects

**Return type** list

**list\_emails** (*page=0, since=None*)

Get the emails of current object

**Parameters**

- **page** – the page starting at 0
- **since** (*datetime.datetime*) – get all notes since a datetime

**Returns** the emails

**Return type** list

**list\_notes** (*page=0, since=None*)

Get the notes of current object

**Parameters**

- **page** – the page starting at 0
- **since** (*datetime.datetime*) – get all notes since a datetime

**Returns** the notes

**Return type** list

**list\_tags** ()

Get the tags of current object

**Returns** the tags

**Return type** list

**list\_tasks** ()

Get the tasks of current object

**Returns** the tasks

**Return type** list

**classmethod** `search (term=None, page=0, **criteria)`

Search a list of the model If you use “term”: - Returns a collection of people that have a name matching the term passed in through the URL.

If you use “criteria”: - returns people who match your search criteria. Search by any criteria you can on the Contacts tab, including custom fields. Combine criteria to narrow results

#### Parameters

- **term** (*str*) – params as string
- **criteria** (*dict*) – search for more criteria
- **page** (*int*) – the page

**Returns** the list of the parsed xml objects

**Return type** list

#### **to\_serializable\_value()**

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

#### **update()**

Updates the object

**Returns**

**Return type** response

## 2.20 PhoneNumber

```
class highton.models.phone_number.PhoneNumber (**kwargs)
```

#### Variables

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **location** – fields.StringField(name=HightonConstants.LOCATION)
- **number** – fields.StringField(name=HightonConstants.NUMBER)

**ENDPOINT** = None

**TAG\_NAME** = 'phone-number'

**classmethod decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)

Parses the `xml.etree.ElementTree.Element` to a string

**Parameters** **element** (`xml.etree.ElementTree.Element`) – a xml element

**Returns** the parsed string

**Return type** `str`

**encode** ()

Encodes the object to a `xml.etree.ElementTree.Element`

**Returns** the encoded element

**Return type** `xml.etree.ElementTree.Element`

**to\_serializable\_value** ()

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** `dict`

## 2.21 SubjectData

**class** `highton.models.subject_data.SubjectData` (*\*\*kwargs*)

**Variables**

- **id** – `fields.IntegerField(name=HightonConstants.ID)`
- **value** – `fields.StringField(name=HightonConstants.VALUE)`
- **type** – `fields.StringField(name=HightonConstants.TYPE)`
- **subject\_field\_id** – `fields.StringField(name=HightonConstants.SUBJECT_FIELD_ID)`
- **subject\_field\_label** – `fields.StringField(name=HightonConstants.SUBJECT_FIELD_LABEL)`

**ENDPOINT** = `None`

**TAG\_NAME** = `'subject_data'`

**classmethod** **decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (`xml.etree.ElementTree.Element`) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** `object`

**static** **element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** `xml.etree.ElementTree.Element`

**static** **element\_to\_string** (*element*)

Parses the `xml.etree.ElementTree.Element` to a string



**Parameters** `element` (`xml.etree.ElementTree.Element`) – a xml element

**Returns** the parsed string

**Return type** `str`

**encode()**

Encodes the object to a `xml.etree.ElementTree.Element`

**Returns** the encoded element

**Return type** `xml.etree.ElementTree.Element`

**to\_serializable\_value()**

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** `dict`

## 2.22 Tag

```
class highton.models.tag.Tag(**kwargs)
```

**Variables**

- **id** – `fields.IntegerField(name=HightonConstants.ID)`
- **name** – `fields.StringField(name=HightonConstants.NAME)`

```
COLLECTION_DATETIME = '%Y%m%d%H%M%S'
```

```
ENDPOINT = 'tags'
```

```
TAG_NAME = 'tag'
```

```
classmethod decode(root_element)
```

Decode the object to the object

**Parameters** `root_element` (`xml.etree.ElementTree.Element`) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** `object`

```
delete()
```

Deletes the object

**Returns**

**Return type** `None`

```
static element_from_string(string)
```

Make a Element from a str

**Parameters** `string` (`str`) – string you want to parse

**Returns** the parsed xml string

**Return type** `xml.etree.ElementTree.Element`

```
static element_to_string(element)
```

Parses the `xml.etree.ElementTree.Element` to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode()**

Encodes the object to a *xml.etree.ElementTree.Element*

**Returns** the encoded element

**Return type** *xml.etree.ElementTree.Element*

**classmethod get** (*object\_id*)

Get all parties (people and companies) associated with a given tag. :param object\_id: the primary id of the model :type object\_id: integer :return: the parties :rtype: list

**classmethod list** (*params=None*)

Retrieves a list of the model

**Parameters** **params** (*dict*) – params as dictionary

**Returns** the list of the parsed xml objects

**Return type** list

**to\_serializable\_value()**

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

## 2.23 Task

**class** *highton.models.task.Task* (*\*\*kwargs*)

**Variables**

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **recording\_id** – fields.IntegerField(name=HightonConstants.RECORDING\_ID)
- **subject\_id** – fields.IntegerField(name=HightonConstants.SUBJECT\_ID)
- **subject\_type** – fields.StringField(name=HightonConstants.SUBJECT\_TYPE)
- **subject\_name** – fields.StringField(name=HightonConstants.SUBJECT\_NAME)
- **category\_id** – fields.IntegerField(name=HightonConstants.CATEGORY\_ID, required=True)
- **body** – fields.StringField(name=HightonConstants.BODY, required=True)
- **frame** – fields.StringField(name=HightonConstants.FRAME, required=True)
- **due\_at** – fields.DatetimeField(name=HightonConstants.DUE\_AT, required=True)
- **alert\_at** – fields.DatetimeField(name=HightonConstants.ALERT\_AT)
- **created\_at** – fields.DatetimeField(name=HightonConstants.CREATED\_AT)
- **author\_id** – fields.IntegerField(name=HightonConstants.AUTHOR\_ID)
- **updated\_at** – fields.DatetimeField(name=HightonConstants.UPDATED\_AT)

- **public** – fields.BooleanField(name=HightonConstants.PUBLIC)
- **recurring\_period** – fields.StringField(name=HightonConstants.RECURRING\_PERIOD)
- **anchor\_type** – fields.IntegerField(name=HightonConstants.ANCHOR\_TYPE)
- **done\_at** – fields.DateTimeField(name=HightonConstants.DONE\_AT)
- **owner\_id** – fields.IntegerField(name=HightonConstants.OWNER\_ID)

**COLLECTION\_DATETIME** = '%Y%m%d%H%M%S'

**ENDPOINT** = 'tasks'

**TAG\_NAME** = 'task'

**complete()**  
Complete current task :return: :rtype: requests.models.Response

**create()**  
Creates the object

**Returns** the created object

**Return type** object

**classmethod decode(*root\_element*)**  
Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**delete\_tag(*tag\_id*)**  
Deletes a Tag to current object

**Parameters** **tag\_id** (*int*) – the id of the tag which should be deleted

**Return type** None

**static element\_from\_string(*string*)**  
Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string(*element*)**  
Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode()**  
Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**classmethod** `get(object_id)`

Retrieves a single model

**Parameters** `object_id(integer)` – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**classmethod** `list_all()`

Returns a collection of all tasks visible to the current user.

**Returns**

**Return type** list

**classmethod** `list_assigned()`

Returns a collection of upcoming tasks (tasks that have not yet been completed, regardless of whether they're overdue) that were created by the authenticated user, but assigned to somebody else.

**Returns**

**Return type** list

**classmethod** `list_completed()`

Returns a collection of completed tasks.

**Returns**

**Return type** list

**classmethod** `list_today()`

Returns a collection of uncompleted tasks due for the rest of today for the authenticated user.

**Returns**

**Return type** list

**classmethod** `list_upcoming()`

Returns a collection of upcoming tasks (tasks that have not yet been completed, regardless of whether they're overdue) for the authenticated user

**Returns**

**Return type** list

**to\_serializable\_value()**

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

**update()**

Updates the object

**Returns**

**Return type** response

## 2.24 TaskCategory

```
class highton.models.task_category.TaskCategory(**kwargs)
```

**Variables**

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **name** – fields.StringField(name=HightonConstants.NAME)
- **color** – fields.StringField(name=HightonConstants.COLOR)
- **account\_id** – fields.IntegerField(name=HightonConstants.ACCOUNT\_ID)
- **created\_at** – fields.DatetimeField(name=HightonConstants.CREATED\_AT)
- **updated\_at** – fields.DatetimeField(name=HightonConstants.UPDATED\_AT)
- **elements\_count** – fields.IntegerField(name=HightonConstants.ELEMENTS\_COUNT)

**COLLECTION\_DATETIME** = '%Y%m%d%H%M%S'

**ENDPOINT** = 'task\_categories'

**TAG\_NAME** = 'task-category'

**create()**

Creates the object

**Returns** the created object

**Return type** object

**classmethod decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**delete()**

Deletes the object

**Returns**

**Return type** None

**static element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode()**

Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**classmethod** `get(object_id)`

Retrieves a single model

**Parameters** `object_id` (*integer*) – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**classmethod** `list(params=None)`

Retrieves a list of the model

**Parameters** `params` (*dict*) – params as dictionary

**Returns** the list of the parsed xml objects

**Return type** list

**to\_serializable\_value()**

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

**update()**

Updates the object

**Returns**

**Return type** response

## 2.25 User

**class** `highton.models.user.User(**kwargs)`

**Variables**

- `id` – fields.IntegerField(name=HightonConstants.ID)
- `name` – fields.StringField(name=HightonConstants.NAME)
- `email_address` – fields.StringField(name=HightonConstants.EMAIL\_ADDRESS)
- `admin` – fields.BooleanField(name=HightonConstants.ADMIN)
- `token` – fields.StringField(name=HightonConstants.TOKEN)
- `dropbox` – fields.StringField(name=HightonConstants.DROPBOX)
- `created_at` – fields.DatetimeField(name=HightonConstants.CREATED\_AT)
- `updated_at` – fields.DatetimeField(name=HightonConstants.UPDATED\_AT)

`COLLECTION_DATETIME = '%Y%m%d%H%M%S'`

`ENDPOINT = 'users'`

`TAG_NAME = 'user'`

**classmethod** `decode(root_element)`

Decode the object to the object

**Parameters** `root_element` (`xml.etree.ElementTree.Element`) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static element\_from\_string** (`string`)  
Make a Element from a str

**Parameters** `string` (`str`) – string you want to parse

**Returns** the parsed xml string

**Return type** `xml.etree.ElementTree.Element`

**static element\_to\_string** (`element`)  
Parses the `xml.etree.ElementTree.Element` to a string

**Parameters** `element` (`xml.etree.ElementTree.Element`) – a xml element

**Returns** the parsed string

**Return type** str

**encode** ()  
Encodes the object to a `xml.etree.ElementTree.Element`

**Returns** the encoded element

**Return type** `xml.etree.ElementTree.Element`

**classmethod get** (`object_id`)  
Retrieves a single model

**Parameters** `object_id` (`integer`) – the primary id of the model

**Returns** the object of the parsed xml object

**Return type** object

**classmethod list** (`params=None`)  
Retrieves a list of the model

**Parameters** `params` (`dict`) – params as dictionary

**Returns** the list of the parsed xml objects

**Return type** list

**classmethod me** ()  
Returns information about the currently authenticated user.

**Returns**

**Return type** *User*

**to\_serializable\_value** ()  
Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

## 2.26 TwitterAccount

```
class highton.models.twitter_account.TwitterAccount (**kwargs)
```

### Variables

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **location** – fields.StringField(name=HightonConstants.LOCATION)
- **username** – fields.StringField(name=HightonConstants.USERNAME)
- **url** – fields.StringField(name=HightonConstants.URL)

**ENDPOINT** = None

**TAG\_NAME** = 'twitter-account'

**classmethod** **decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static** **element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static** **element\_to\_string** (*element*)

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode** ()

Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**to\_serializable\_value** ()

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict

## 2.27 WebAddress

```
class highton.models.web_address.WebAddress (**kwargs)
```



**Variables**

- **id** – fields.IntegerField(name=HightonConstants.ID)
- **location** – fields.StringField(name=HightonConstants.LOCATION)
- **url** – fields.StringField(name=HightonConstants.URL)

**ENDPOINT** = None

**TAG\_NAME** = 'web-address'

**classmethod decode** (*root\_element*)

Decode the object to the object

**Parameters** **root\_element** (*xml.etree.ElementTree.Element*) – the parsed xml Element

**Returns** the decoded Element as object

**Return type** object

**static element\_from\_string** (*string*)

Make a Element from a str

**Parameters** **string** (*str*) – string you want to parse

**Returns** the parsed xml string

**Return type** xml.etree.ElementTree.Element

**static element\_to\_string** (*element*)

Parses the xml.etree.ElementTree.Element to a string

**Parameters** **element** (*xml.etree.ElementTree.Element*) – a xml element

**Returns** the parsed string

**Return type** str

**encode** ()

Encodes the object to a xml.etree.ElementTree.Element

**Returns** the encoded element

**Return type** xml.etree.ElementTree.Element

**to\_serializable\_value** ()

Parses the Hightonmodel to a serializable value such dicts, lists, strings This can be used to save the model in a NoSQL database

**Returns** the serialized HightonModel

**Return type** dict



### 3.1 BooleanField

**class** `highton.fields.boolean_field.BooleanField` (*name*, *required=False*)

The BooleanField represents a field which value will be parsed to Highrise specific boolean format

**FALSE** = 'false'

**MAPPING** = {False: 'false', True: 'true'}

**TRUE** = 'true'

**decode** (*element*)

**Parameters** *element* (`xml.etree.ElementTree.Element`) – the element which needs to be parsed

**Returns** `datetime.date`

**Return type** the parsed date object

**encode** ()

**Returns**

**Return type** `xml.etree.ElementTree.Element`

**to\_serializable\_value** ()

Parse the value to a serializable pythonic value Default: Just return the value

**Returns**

**Return type** Any

## 3.2 DateField

```
class highton.fields.date_field.DateField(name, required=False)
    The DateField represents a field which value will be parsed to Highrise specific date format

    DATE_FORMAT = '%Y-%m-%d'

    decode(element)

        Parameters element (xml.etree.ElementTree.Element) – the element which needs
            to be parsed

        Returns datetime.date

        Return type the parsed date object

    encode()

        Returns

        Return type xml.etree.ElementTree.Element

    to_serializable_value()
        Parse the value to a serializable pythonic value Default: Just return the value

        Returns

        Return type Any
```

## 3.3 DateTimeField

```
class highton.fields.datetime_field.DatetimeField(name, required=False)

    DATETIME_FORMAT = '%Y-%m-%dT%H:%M:%SZ'

    decode(element)

        Parameters element (xml.etree.ElementTree.Element) – the element which needs
            to be parsed

        Returns the parsed datetime object

        Return type datetime.datetime

    encode()

        Returns the encoded element

        Return type xml.etree.ElementTree.Element

    to_serializable_value()
        Parse the value to a serializable pythonic value Default: Just return the value

        Returns

        Return type Any
```

## 3.4 Field

**class** `highton.fields.field.Field` (*name*, *required=False*)

This is an abstract class for a field, which makes the base for an encode and decode method

### Variables

- **name** – name of the field which needed to serialize this field to a xml object
- **required** – if the field is required when you create or update an object
- **value** – the value of the field

**decode** (*element*)

Decodes the value of the element

**Parameters** **element** –

**Returns**

**Return type**

**encode** ()

Encodes the value of the field and put it in the element also make the check for nil=true if there is one

**Returns** returns the encoded element

**Return type** `xml.etree.ElementTree.Element`

**to\_serializable\_value** ()

Parse the value to a serializable pythonic value Default: Just return the value

**Returns**

**Return type** Any

## 3.5 IntegerField

**class** `highton.fields.integer_field.IntegerField` (*name*, *required=False*)

**decode** (*element*)

**Parameters** **element** (`xml.etree.ElementTree.Element`) –

**Returns** the parsed int object

**Return type** int

**encode** ()

**Returns**

**Return type** `xml.etree.ElementTree.Element`

**to\_serializable\_value** ()

Parse the value to a serializable pythonic value Default: Just return the value

**Returns**

**Return type** Any

## 3.6 ListField

**class** `highton.fields.list_field.ListField`(*name*, *init\_class*)

The ListField parses the `init_class` objects in a list

**Variables** `init_class` – a Highton model

**decode** (*element*)

**Parameters** `element` –

**Returns** the parsed list with `init_class` objects

**Return type** list

**encode** ()

Just iterate over the child elements and append them to the current element

**Returns** the encoded element

**Return type** `xml.etree.ElementTree.Element`

**to\_serializable\_value** ()

Run through the values and parse them to a serializable value

**Returns**

**Return type** list

## 3.7 ObjectField

**class** `highton.fields.object_field.ObjectField`(*name*, *init\_class*)

The ObjectField parses the `init_class` object

**Variables** `init_class` – a Highton model

**decode** (*element*)

Decodes the value of the element

**Parameters** `element` –

**Returns**

**Return type**

**encode** ()

Just encode the object you have as value

**Returns** the parsed element

**Return type** `xml.etree.ElementTree.Element`

**to\_serializable\_value** ()

Run through all fields of the object and parse the values

**Returns**

**Return type** dict

## 3.8 StringField

**class** highton.fields.string\_field.**StringField**(*name*, *required=False*)

**decode** (*element*)

Decodes the value of the element

**Parameters** *element* –

**Returns**

**Return type**

**encode** ()

Encodes the value of the field and put it in the element also make the check for nil=true if there is one

**Returns** returns the encoded element

**Return type** xml.etree.ElementTree.Element

**to\_serializable\_value** ()

Parse the value to a serializable pythonic value Default: Just return the value

**Returns**

**Return type** Any





## CHAPTER 4

---

### Indices and tables

---

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



### h

- `highton.fields.boolean_field`, 47
- `highton.fields.date_field`, 48
- `highton.fields.datetime_field`, 48
- `highton.fields.field`, 49
- `highton.fields.integer_field`, 49
- `highton.fields.list_field`, 50
- `highton.fields.object_field`, 50
- `highton.fields.string_field`, 51
- `highton.models.address`, 3
- `highton.models.associated_party`, 4
- `highton.models.attachment`, 5
- `highton.models.case`, 6
- `highton.models.category`, 9
- `highton.models.comment`, 10
- `highton.models.company`, 11
- `highton.models.contact`, 15
- `highton.models.contact_data`, 16
- `highton.models.deal`, 17
- `highton.models.deal_category`, 21
- `highton.models.email`, 22
- `highton.models.email_address`, 24
- `highton.models.group`, 25
- `highton.models.highton_model`, 27
- `highton.models.instant_messenger`, 27
- `highton.models.note`, 28
- `highton.models.party`, 30
- `highton.models.person`, 32
- `highton.models.phone_number`, 35
- `highton.models.subject_data`, 36
- `highton.models.tag`, 37
- `highton.models.task`, 38
- `highton.models.task_category`, 40
- `highton.models.twitter_account`, 44
- `highton.models.user`, 42
- `highton.models.web_address`, 44



## A

add\_note() (highton.models.case.Case method), 7  
 add\_note() (highton.models.company.Company method), 12  
 add\_note() (highton.models.deal.Deal method), 18  
 add\_note() (highton.models.person.Person method), 32  
 add\_tag() (highton.models.case.Case method), 7  
 add\_tag() (highton.models.company.Company method), 12  
 add\_tag() (highton.models.deal.Deal method), 18  
 add\_tag() (highton.models.person.Person method), 33  
 Address (class in highton.models.address), 3  
 AssociatedParty (class in highton.models.associated\_party), 4  
 Attachment (class in highton.models.attachment), 5

## B

BooleanField (class in highton.fields.boolean\_field), 47

## C

Case (class in highton.models.case), 6  
 Category (class in highton.models.category), 9  
 COLLECTION\_DATETIME (highton.models.attachment.Attachment attribute), 6  
 COLLECTION\_DATETIME (highton.models.case.Case attribute), 7  
 COLLECTION\_DATETIME (highton.models.comment.Comment attribute), 10  
 COLLECTION\_DATETIME (highton.models.company.Company attribute), 12  
 COLLECTION\_DATETIME (highton.models.deal.Deal attribute), 18  
 COLLECTION\_DATETIME (highton.models.deal\_category.DealCategory attribute), 21

COLLECTION\_DATETIME (highton.models.email.Email attribute), 23  
 COLLECTION\_DATETIME (highton.models.group.Group attribute), 25  
 COLLECTION\_DATETIME (highton.models.note.Note attribute), 29  
 COLLECTION\_DATETIME (highton.models.person.Person attribute), 32  
 COLLECTION\_DATETIME (highton.models.tag.Tag attribute), 37  
 COLLECTION\_DATETIME (highton.models.task.Task attribute), 39  
 COLLECTION\_DATETIME (highton.models.task\_category.TaskCategory attribute), 41  
 COLLECTION\_DATETIME (highton.models.user.User attribute), 42  
 Comment (class in highton.models.comment), 10  
 COMMENT\_OFFSET (highton.models.email.Email attribute), 23  
 COMMENT\_OFFSET (highton.models.note.Note attribute), 29  
 Company (class in highton.models.company), 11  
 complete() (highton.models.task.Task method), 39  
 Contact (class in highton.models.contact), 15  
 ContactData (class in highton.models.contact\_data), 16  
 create() (highton.models.case.Case method), 7  
 create() (highton.models.comment.Comment method), 10  
 create() (highton.models.company.Company method), 12  
 create() (highton.models.deal.Deal method), 18  
 create() (highton.models.deal\_category.DealCategory method), 21  
 create() (highton.models.email.Email method), 23  
 create() (highton.models.group.Group method), 25  
 create() (highton.models.note.Note method), 29  
 create() (highton.models.person.Person method), 33  
 create() (highton.models.task.Task method), 39  
 create() (highton.models.task\_category.TaskCategory method), 41

## D

DATE\_FORMAT (highton.fields.date\_field.DateField attribute), 48

DateField (class in highton.fields.date\_field), 48

DATETIME\_FORMAT (highton.fields.datetime\_field.DatetimeField attribute), 48

DatetimeField (class in highton.fields.datetime\_field), 48

Deal (class in highton.models.deal), 17

DealCategory (class in highton.models.deal\_category), 21

decode() (highton.fields.boolean\_field.BooleanField method), 47

decode() (highton.fields.date\_field.DateField method), 48

decode() (highton.fields.datetime\_field.DatetimeField method), 48

decode() (highton.fields.field.Field method), 49

decode() (highton.fields.integer\_field.IntegerField method), 49

decode() (highton.fields.list\_field.ListField method), 50

decode() (highton.fields.object\_field.ObjectField method), 50

decode() (highton.fields.string\_field.StringField method), 51

decode() (highton.models.address.Address class method), 3

decode() (highton.models.associated\_party.AssociatedParty class method), 5

decode() (highton.models.attachment.Attachment class method), 6

decode() (highton.models.case.Case class method), 7

decode() (highton.models.category.Category class method), 9

decode() (highton.models.comment.Comment class method), 10

decode() (highton.models.company.Company class method), 12

decode() (highton.models.contact.Contact class method), 15

decode() (highton.models.contact\_data.ContactData class method), 16

decode() (highton.models.deal.Deal class method), 19

decode() (highton.models.deal\_category.DealCategory class method), 21

decode() (highton.models.email.Email class method), 23

decode() (highton.models.email\_address.EmailAddress class method), 24

decode() (highton.models.group.Group class method), 25

decode() (highton.models.highton\_model.HightonModel class method), 27

decode() (highton.models.instant\_messenger.InstantMessenger class method), 28

decode() (highton.models.note.Note class method), 29

decode() (highton.models.party.Party class method), 31

decode() (highton.models.person.Person class method), 33

decode() (highton.models.phone\_number.PhoneNumber class method), 35

decode() (highton.models.subject\_data.SubjectData class method), 36

decode() (highton.models.tag.Tag class method), 37

decode() (highton.models.task.Task class method), 39

decode() (highton.models.task\_category.TaskCategory class method), 41

decode() (highton.models.twitter\_account.TwitterAccount class method), 44

decode() (highton.models.user.User class method), 42

decode() (highton.models.web\_address.WebAddress class method), 45

delete() (highton.models.case.Case method), 7

delete() (highton.models.comment.Comment method), 10

delete() (highton.models.company.Company method), 13

delete() (highton.models.deal.Deal method), 19

delete() (highton.models.deal\_category.DealCategory method), 21

delete() (highton.models.email.Email method), 23

delete() (highton.models.group.Group method), 26

delete() (highton.models.note.Note method), 29

delete() (highton.models.person.Person method), 33

delete() (highton.models.tag.Tag method), 37

delete() (highton.models.task\_category.TaskCategory method), 41

delete\_tag() (highton.models.case.Case method), 7

delete\_tag() (highton.models.company.Company method), 13

delete\_tag() (highton.models.deal.Deal method), 19

delete\_tag() (highton.models.person.Person method), 33

delete\_tag() (highton.models.task.Task method), 39

## E

element\_from\_string() (highton.models.address.Address static method), 3

element\_from\_string() (highton.models.associated\_party.AssociatedParty static method), 5

element\_from\_string() (highton.models.attachment.Attachment static method), 6

element\_from\_string() (highton.models.case.Case static method), 8

element\_from\_string() (highton.models.category.Category static method), 9

element\_from\_string() (highton.models.comment.Comment static method), 11

element\_from\_string() (highton.models.company.Company static method),

Index 59

method), 41

element\_to\_string() (highton.models.twitter\_account.TwitterAccount static method), 44

element\_to\_string() (highton.models.user.User static method), 43

element\_to\_string() (highton.models.web\_address.WebAddress static method), 45

Email (class in `highton.models.email`), 22

EmailAddress (class in `highton.models.email_address`), 24

EMAILS\_OFFSET (highton.models.case.Case attribute), 7

EMAILS\_OFFSET (highton.models.company.Company attribute), 12

EMAILS\_OFFSET (highton.models.deal.Deal attribute), 18

EMAILS\_OFFSET (highton.models.person.Person attribute), 32

encode() (highton.fields.boolean\_field.BooleanField method), 47

encode() (highton.fields.date\_field.DateField method), 48

encode() (highton.fields.datetime\_field.DatetimeField method), 48

encode() (highton.fields.field.Field method), 49

encode() (highton.fields.integer\_field.IntegerField method), 49

encode() (highton.fields.list\_field.ListField method), 50

encode() (highton.fields.object\_field.ObjectField method), 50

encode() (highton.fields.string\_field.StringField method), 51

encode() (highton.models.address.Address method), 4

encode() (highton.models.associated\_party.AssociatedParty method), 5

encode() (highton.models.attachment.Attachment method), 6

encode() (highton.models.case.Case method), 8

encode() (highton.models.category.Category method), 10

encode() (highton.models.comment.Comment method), 11

encode() (highton.models.company.Company method), 13

encode() (highton.models.contact.Contact method), 16

encode() (highton.models.contact\_data.ContactData method), 17

encode() (highton.models.deal.Deal method), 19

encode() (highton.models.deal\_category.DealCategory method), 22

encode() (highton.models.email.Email method), 23

encode() (highton.models.email\_address.EmailAddress method), 25

encode() (highton.models.group.Group method), 26

encode() (highton.models.highton\_model.HightonModel method), 27

encode() (highton.models.instant\_messenger.InstantMessenger method), 28

encode() (highton.models.note.Note method), 30

encode() (highton.models.party.Party method), 31

encode() (highton.models.person.Person method), 33

encode() (highton.models.phone\_number.PhoneNumber method), 36

encode() (highton.models.subject\_data.SubjectData method), 37

encode() (highton.models.tag.Tag method), 38

encode() (highton.models.task.Task method), 39

encode() (highton.models.task\_category.TaskCategory method), 41

encode() (highton.models.twitter\_account.TwitterAccount method), 44

encode() (highton.models.user.User method), 43

encode() (highton.models.web\_address.WebAddress method), 45

ENDPOINT (highton.models.address.Address attribute), 3

ENDPOINT (highton.models.associated\_party.AssociatedParty attribute), 5

ENDPOINT (highton.models.attachment.Attachment attribute), 6

ENDPOINT (highton.models.case.Case attribute), 7

ENDPOINT (highton.models.category.Category attribute), 9

ENDPOINT (highton.models.comment.Comment attribute), 10

ENDPOINT (highton.models.company.Company attribute), 12

ENDPOINT (highton.models.contact.Contact attribute), 15

ENDPOINT (highton.models.contact\_data.ContactData attribute), 16

ENDPOINT (highton.models.deal.Deal attribute), 18

ENDPOINT (highton.models.deal\_category.DealCategory attribute), 21

ENDPOINT (highton.models.email.Email attribute), 23

ENDPOINT (highton.models.email\_address.EmailAddress attribute), 24

ENDPOINT (highton.models.group.Group attribute), 25

ENDPOINT (highton.models.highton\_model.HightonModel attribute), 27

ENDPOINT (highton.models.instant\_messenger.InstantMessenger attribute), 28

ENDPOINT (highton.models.note.Note attribute), 29

ENDPOINT (highton.models.party.Party attribute), 31

ENDPOINT (highton.models.person.Person attribute), 32

ENDPOINT (highton.models.phone\_number.PhoneNumber attribute), 35

ENDPOINT (highton.models.subject\_data.SubjectData



- attribute), 36
- ENDPOINT (highton.models.tag.Tag attribute), 37
- ENDPOINT (highton.models.task.Task attribute), 39
- ENDPOINT (highton.models.task\_category.TaskCategory attribute), 41
- ENDPOINT (highton.models.twitter\_account.TwitterAccount attribute), 44
- ENDPOINT (highton.models.user.User attribute), 42
- ENDPOINT (highton.models.web\_address.WebAddress attribute), 45
- F**
- FALSE (highton.fields.boolean\_field.BooleanField attribute), 47
- Field (class in highton.fields.field), 49
- G**
- get() (highton.models.attachment.Attachment method), 6
- get() (highton.models.case.Case class method), 8
- get() (highton.models.comment.Comment class method), 11
- get() (highton.models.company.Company class method), 13
- get() (highton.models.deal.Deal class method), 19
- get() (highton.models.deal\_category.DealCategory class method), 22
- get() (highton.models.email.Email class method), 24
- get() (highton.models.group.Group class method), 26
- get() (highton.models.note.Note class method), 30
- get() (highton.models.person.Person class method), 33
- get() (highton.models.tag.Tag class method), 38
- get() (highton.models.task.Task class method), 39
- get() (highton.models.task\_category.TaskCategory class method), 42
- get() (highton.models.user.User class method), 43
- Group (class in highton.models.group), 25
- H**
- highton.fields.boolean\_field (module), 47
- highton.fields.date\_field (module), 48
- highton.fields.datetime\_field (module), 48
- highton.fields.field (module), 49
- highton.fields.integer\_field (module), 49
- highton.fields.list\_field (module), 50
- highton.fields.object\_field (module), 50
- highton.fields.string\_field (module), 51
- highton.models.address (module), 3
- highton.models.associated\_party (module), 4
- highton.models.attachment (module), 5
- highton.models.case (module), 6
- highton.models.category (module), 9
- highton.models.comment (module), 10
- highton.models.company (module), 11
- highton.models.contact (module), 15
- highton.models.contact\_data (module), 16
- highton.models.deal (module), 17
- highton.models.deal\_category (module), 21
- highton.models.email (module), 22
- highton.models.email\_address (module), 24
- highton.models.group (module), 25
- highton.models.highton\_model (module), 27
- highton.models.instant\_messenger (module), 27
- highton.models.note (module), 28
- highton.models.party (module), 30
- highton.models.person (module), 32
- highton.models.phone\_number (module), 35
- highton.models.subject\_data (module), 36
- highton.models.tag (module), 37
- highton.models.task (module), 38
- highton.models.task\_category (module), 40
- highton.models.twitter\_account (module), 44
- highton.models.user (module), 42
- highton.models.web\_address (module), 44
- HightonModel (class in highton.models.highton\_model), 27
- I**
- InstantMessenger (class in highton.models.instant\_messenger), 27
- IntegerField (class in highton.fields.integer\_field), 49
- L**
- list() (highton.models.case.Case class method), 8
- list() (highton.models.company.Company class method), 13
- list() (highton.models.deal.Deal class method), 19
- list() (highton.models.deal\_category.DealCategory class method), 22
- list() (highton.models.group.Group class method), 26
- list() (highton.models.person.Person class method), 34
- list() (highton.models.tag.Tag class method), 38
- list() (highton.models.task\_category.TaskCategory class method), 42
- list() (highton.models.user.User class method), 43
- list\_all() (highton.models.task.Task class method), 40
- list\_assigned() (highton.models.task.Task class method), 40
- list\_comments() (highton.models.email.Email method), 24
- list\_comments() (highton.models.note.Note method), 30
- list\_completed() (highton.models.task.Task class method), 40
- list\_emails() (highton.models.case.Case method), 8
- list\_emails() (highton.models.company.Company method), 14
- list\_emails() (highton.models.deal.Deal method), 20
- list\_emails() (highton.models.person.Person method), 34
- list\_notes() (highton.models.case.Case method), 8

list\_notes() (highton.models.company.Company method), 14

list\_notes() (highton.models.deal.Deal method), 20

list\_notes() (highton.models.person.Person method), 34

list\_tags() (highton.models.case.Case method), 9

list\_tags() (highton.models.company.Company method), 14

list\_tags() (highton.models.deal.Deal method), 20

list\_tags() (highton.models.person.Person method), 34

list\_tasks() (highton.models.case.Case method), 9

list\_tasks() (highton.models.company.Company method), 14

list\_tasks() (highton.models.deal.Deal method), 20

list\_tasks() (highton.models.person.Person method), 34

list\_today() (highton.models.task.Task class method), 40

list\_upcoming() (highton.models.task.Task class method), 40

ListField (class in highton.fields.list\_field), 50

## M

MAPPING (highton.fields.boolean\_field.BooleanField attribute), 47

me() (highton.models.user.User class method), 43

## N

Note (class in highton.models.note), 28

NOTES\_OFFSET (highton.models.case.Case attribute), 7

NOTES\_OFFSET (highton.models.company.Company attribute), 12

NOTES\_OFFSET (highton.models.deal.Deal attribute), 18

NOTES\_OFFSET (highton.models.person.Person attribute), 32

## O

ObjectField (class in highton.fields.object\_field), 50

OFFSET (highton.models.company.Company attribute), 12

OFFSET (highton.models.deal.Deal attribute), 18

OFFSET (highton.models.person.Person attribute), 32

## P

Party (class in highton.models.party), 30

people() (highton.models.company.Company method), 14

Person (class in highton.models.person), 32

PhoneNumber (class in highton.models.phone\_number), 35

## S

search() (highton.models.company.Company class method), 14

search() (highton.models.person.Person class method), 34

SEARCH\_OFFSET (highton.models.company.Company attribute), 12

SEARCH\_OFFSET (highton.models.person.Person attribute), 32

StringField (class in highton.fields.string\_field), 51

SubjectData (class in highton.models.subject\_data), 36

## T

Tag (class in highton.models.tag), 37

TAG\_NAME (highton.models.address.Address attribute), 3

TAG\_NAME (highton.models.associated\_party.AssociatedParty attribute), 5

TAG\_NAME (highton.models.attachment.Attachment attribute), 6

TAG\_NAME (highton.models.case.Case attribute), 7

TAG\_NAME (highton.models.category.Category attribute), 9

TAG\_NAME (highton.models.comment.Comment attribute), 10

TAG\_NAME (highton.models.company.Company attribute), 12

TAG\_NAME (highton.models.contact.Contact attribute), 15

TAG\_NAME (highton.models.contact\_data.ContactData attribute), 16

TAG\_NAME (highton.models.deal.Deal attribute), 18

TAG\_NAME (highton.models.deal\_category.DealCategory attribute), 21

TAG\_NAME (highton.models.email.Email attribute), 23

TAG\_NAME (highton.models.email\_address.EmailAddress attribute), 24

TAG\_NAME (highton.models.group.Group attribute), 25

TAG\_NAME (highton.models.highton\_model.HightonModel attribute), 27

TAG\_NAME (highton.models.instant\_messenger.InstantMessenger attribute), 28

TAG\_NAME (highton.models.note.Note attribute), 29

TAG\_NAME (highton.models.party.Party attribute), 31

TAG\_NAME (highton.models.person.Person attribute), 32

TAG\_NAME (highton.models.phone\_number.PhoneNumber attribute), 35

TAG\_NAME (highton.models.subject\_data.SubjectData attribute), 36

TAG\_NAME (highton.models.tag.Tag attribute), 37

TAG\_NAME (highton.models.task.Task attribute), 39

TAG\_NAME (highton.models.task\_category.TaskCategory attribute), 41

TAG\_NAME (highton.models.twitter\_account.TwitterAccount attribute), 44

TAG\_NAME (highton.models.user.User attribute), 42

TAG\_NAME (highton.models.web\_address.WebAddress attribute), 45

Task (class in highton.models.task), 38

TaskCategory (class in highton.models.task\_category), 40

[to\\_serializable\\_value\(\)](#) (high-ton.fields.boolean\_field.BooleanField method), [47](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.fields.date\_field.DateField method), [48](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.fields.datetime\_field.DatetimeField method), [48](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.fields.field.Field method), [49](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.fields.integer\_field.IntegerField method), [49](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.fields.list\_field.ListField method), [50](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.fields.object\_field.ObjectField method), [50](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.fields.string\_field.StringField method), [51](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.address.Address method), [4](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.associated\_party.AssociatedParty method), [5](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.attachment.Attachment method), [6](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.case.Case method), [9](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.category.Category method), [10](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.comment.Comment method), [11](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.company.Company method), [14](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.contact.Contact method), [16](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.contact\_data.ContactData method), [17](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.deal.Deal method), [20](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.deal\_category.DealCategory method), [22](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.email.Email method), [24](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.email\_address.EmailAddress method), [25](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.group.Group method), [26](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.highton\_model.HightonModel method), [27](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.instant\_messenger.InstantMessenger method), [28](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.note.Note method), [30](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.party.Party method), [31](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.person.Person method), [35](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.phone\_number.PhoneNumber method), [36](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.subject\_data.SubjectData method), [37](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.tag.Tag method), [38](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.task.Task method), [40](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.task\_category.TaskCategory method), [42](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.twitter\_account.TwitterAccount method), [44](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.user.User method), [43](#)  
[to\\_serializable\\_value\(\)](#) (high-ton.models.web\_address.WebAddress method), [45](#)  
[TRUE](#) (high-ton.fields.boolean\_field.BooleanField attribute), [47](#)  
[TwitterAccount](#) (class in high-ton.models.twitter\_account), [44](#)

## U

[update\(\)](#) (high-ton.models.case.Case method), [9](#)  
[update\(\)](#) (high-ton.models.comment.Comment method), [11](#)  
[update\(\)](#) (high-ton.models.company.Company method), [15](#)  
[update\(\)](#) (high-ton.models.deal.Deal method), [20](#)  
[update\(\)](#) (high-ton.models.deal\_category.DealCategory method), [22](#)  
[update\(\)](#) (high-ton.models.email.Email method), [24](#)  
[update\(\)](#) (high-ton.models.group.Group method), [26](#)  
[update\(\)](#) (high-ton.models.note.Note method), [30](#)  
[update\(\)](#) (high-ton.models.person.Person method), [35](#)  
[update\(\)](#) (high-ton.models.task.Task method), [40](#)

`update()` (highton.models.task\_category.TaskCategory method), [42](#)  
`update_status()` (highton.models.deal.Deal method), [20](#)  
`User` (class in highton.models.user), [42](#)

## W

`WebAddress` (class in highton.models.web\_address), [44](#)