pinq Documentation

Release 0.2.0

David Shriver

April 08, 2016

Contents

CHAPTER 1

User Guide

This section is still in progress. This section provides instructions for installation and use, as well as some general background information.

CHAPTER 2

API Documentation

This sections provides documentation on the functions, classes, and methods of pinq.

2.1 Interface

2.1.1 Main Interface

pinq.**as_queryable**(*iterable*)

Constructs a queryable object using *iterable* as the base data.

Parameters **iterable** (*Iterable*) – iterable object to make queryable.

Returns a queryable object with the specified iterable as the underlying data

Return type Queryable object

Raises **TypeError** – if iterable is not an Iterable

Usage:

```
>>> import pinq
>>> queryable = pinq.as_queryable(range(100))
```

2.1.2 Queryable's and Their Methods

class pinq.queryable.**Queryable**(*iterator*)

Bases: object

A wrapper for iterable objects to allow querying of the underlying data.

aggregate(*accumulator*, *seed=None*, *result_transform=<function identity>*)

Applies an accumulator function over a sequence.

Parameters

- **accumulator** (*function*) – The accumulator function to apply.
- **seed** – (optional) The initial accumulator value.
- **result_transform** (*function*) – (optional) A transform function to apply to the result.

Returns The accumulated value.

Raises

- **TypeError** – if 'accumulator' is not callable
- **TypeError** – if 'result_transform' is not callable

all(*predicate*)
Determines whether all elements of the sequence satisfy a condition.

Parameters **predicate** (*function*) – A function to test each element for a condition.

Returns True if every element satisfies the condition, or the sequence is empty.

Return type bool

Raises **TypeError** – if 'predicate' is not callable

any(*predicate=<function true>*)
Determines whether any element of the sequence satisfies a condition.

Parameters **predicate** (*function*) – (optional) A function to test each element for a condition.

Returns True if any element satisfies the condition.

Return type bool

Raises **TypeError** – if 'predicate' is not callable

average(*transform=<function identity>*)
Computes the average of the elements in the sequence.

Parameters **transform** (*function:*) – (optional) A transform function to invoke on each element of the sequence.

Returns The average value of the elements in the sequence.

Return type float

Raises **TypeError** – if 'transform' is not callable

cast(*to_type*)
Casts the elements of the sequence to the specified type.

Parameters **to_type** (*type*) – The type to cast elements to.

Returns The elements of the sequence cast to the specified type.

Return type Queryable

Raises **TypeError** – if 'to_type' is not callable

concat(*other*)
Concatenates two sequences.

Parameters **other** (*Iterable*) – The sequence to concatenate to this sequence.

Returns A Queryable containing the concatenated elements of the two input sequences.

Return type Queryable

Raises **TypeError** – if 'other' is not Iterable

contains(*value*, *equality_comparer=<built-in function eq>*)
Determines whether the sequence contains the specified value.

Parameters

- **value** – The value to find in the sequence.

- **equality_comparer** (*function*) – (optional) An equality comparer to compare values.

Returns True if the sequence contains the specified value.

Return type bool

Raises **TypeError** – if 'equality_comparer' is not callable

count (*predicate=<function true>*)
Returns the number of elements in the sequence.

Parameters **predicate** (*function*) – (optional) A function to test each element for a condition:

Returns The number of elements that satisfy the specified condition.

Return type int

Raises **TypeError** – if 'predicate' is not callable

default_if_empty (*default_value=None*)
Returns the sequence or a sequence with a single default value if the sequence is empty.

Parameters **default_value** – (optional) The default value to return.

:return:This sequence, or a sequence containing 'default_value' if it is empty. :rtype: Queryable

difference (*other*, *key_selector=<function identity>*)
Returns the set difference of the two sequences.

Parameters

- **other** (*Iterable*) – An iterable of elements to be removed from this sequence.
- **key_selector** (*function*) – (optional) An function to select a key for comparing values.

Returns The set difference of this sequence and the provided sequence.

Return type Queryable

Raises

- **TypeError** – if 'other' is not an Interable
- **TypeError** – if 'key_selector' is not callable

distinct (*key_selector=<function identity>*)
Returns distinct elements fromt the sequence.

Parameters **key_selector** (*function*) – (optional) An function to select a key for comparing values.

Returns A sequence of distinct elements from this sequence.

Return type Queryable

Raises **TypeError** – if 'key_selector' is not callable

element_at (*index*)
Returns the element at the specified location in the sequence.

Parameters **index** (*int*) – The zero-based index of the element to retrieve.

Returns The element at the specified location in the sequence.

Raises

- **TypeError** – if 'index' is not an int
- **IndexError** – if 'index' is less than zero or larger than the number of elements

element_at_or_default(*index*, *default_value=None*)
Returns the element at the specified index or a default value if it is out of range.

Parameters

- **index** (*int*) – The zero-based index of the element to retrieve.
- **default_value** – (optional) The default value if the index is out of range.

Returns The element at the specified location in the sequence.

Raises **TypeError** – if 'index' is not an int

except_values(*other*, *key_selector=<function identity>*)
Returns the set difference of the two sequences.

Parameters

- **other** (*Iterable*) – An iterable of elements to be removed from this sequence.
- **key_selector** (*function*) – (optional) An function to select a key for comparing values.

Returns The set difference of this sequence and the provided sequence.

Return type Queryable

Raises

- **TypeError** – if 'other' is not an Interable
- **TypeError** – if 'key_selector' is not callable

first(*predicate=<function true>*)
Returns the first element in the sequence.

Parameters **predicate** (*function*) – (optional) A function to test each element for a condition.

Returns The first element of the sequence satisfying the condition.

Raises

- **TypeError** – if 'predicate' is not callable
- **ValueError** – if the sequence is empty
- **ValueError** – if no element satisfies the condition

first_or_default(*predicate=<function true>*, *default_value=None*)
Returns the first element in the sequence or a default value if empty.

Parameters

- **predicate** (*function*) – (optional) A function to test each element for a condition.
- **default_value** – (optional) The default value to return if empty.

Returns The first element of the sequence satisfying the condition.

Raises **TypeError** – if 'predicate' is not callable

group_by(*key_selector*, *value_transform=<function identity>*, *result_transform=<function identity>*)
Groups the elements of the sequence according to the specified key selector function.

Parameters

- **key_selector** (*function*) – A function to extract the key for each element.
- **value_transform** (*function*) – A transform function to be applied to each element.
- **result_transform** (*function*) – A transform function to be applied to each group.

Returns A sequence where each element represents the transformation of a group and its key.

Return type Queryable

Raises

- **TypeError** – if 'key_selector' is not callable
- **TypeError** – if 'value_transform' is not callable
- **TypeError** – if 'result_transform' is not callable

group_join(*other*, *key_selector*, *other_key_selector*, *result_transform*)
Correlates the elements of the two sequences and groups the results.

Parameters

- **other** (*Iterable*) – The sequence to join this sequence.
- **key_selector** (*function*) – A function to extract a key from each element of this sequence.
- **other_key_selector** (*function*) – A function to extract a key from each element of 'other'.
- **result_transform** (*function*) – A function to create a result from an item and its matching group.

Returns The elements of the two sequences after performing a grouped join.

Return type Queryable

Raises

- **TypeError** – if 'other' is not an Iterable
- **TypeError** – if 'key_selector' is not callable
- **TypeError** – if 'other_key_selector' is not callable
- **TypeError** – if 'result_transform' is not callable

intersect(*other*, *key_selector=<function identity>*)
Returns the set intersection of the two sequences.

Parameters

- **other** (*Iterable*) – A sequence to compute the intersection with.
- **key_selector** (*function*) – (optional) A function to extract a key for each element for comparison.

Returns A sequence of distinct elements that are in both of the provided seequences.

Return type Queryable

Raises

- **TypeError** – if 'other' is not an Iterable
- **TypeError** – if 'key_selector' is not callable

join (*other*, *key_selector*, *other_key_selector*, *result_transform*)
Correlates the elements of the two sequences.

Parameters

- **other** (*Iterable*) – The sequence to join this sequence.
- **key_selector** (*function*) – A function to extract a key from each element of this sequence.
- **other_key_selector** (*function*) – A function to extract a key from each element of 'other'.
- **result_transform** (*function*) – A function to create a result from an item and its match.

Returns The elements of the two sequences after performing an inner join.

Return type Queryable

Raises

- **TypeError** – if 'other' is not an Iterable
- **TypeError** – if 'key_selector' is not callable
- **TypeError** – if 'other_key_selector' is not callable
- **TypeError** – if 'result_transform' is not callable

last (*predicate=<function true>*)
Returns the last item of the sequence.

Parameters **predicate** (*function*) – (optional) A function to test each element for a condition.

Returns The last item in the sequence that satisfies the condition.

Raises

- **TypeError** – if 'predicate' is not callable
- **ValueError** – if the source iterable is empty
- **ValueError** – if no element satisfies condition

last_or_default (*predicate=<function true>*, *default_value=None*)
Returns the last item of the sequence.

Parameters

- **predicate** (*function*) – (optional) A function to test each element for a condition.
- **default_value** – (optional) The default value to return if empty.

Returns The last item in the sequence that satisfies the condition.

Raises

- **TypeError** – if 'predicate' is not callable
- **ValueError** – if the source iterable is empty
- **ValueError** – if no element satisfies condition

long_count (*predicate=<function true>*)
Returns the number of elements in the sequence.

Parameters **predicate** (*function*) – (optional) A function to test each element for a condition:

Returns The number of elements that satisfy the specified condition.

Return type int

Raises **TypeError** – if 'predicate' is not callable

max(*transform=<function identity>*)
Returns the maximum element in the sequence.

Parameters **transform** (*function*) – (optional) A transformation function to apply to each element.

Returns The maximum element in the sequence.

Return type int

Raises **TypeError** – if 'transform' is not callable

min(*transform=<function identity>*)
Returns the minimum element in the sequence.

Parameters **transform** (*function*) – (optional) A transformation function to apply to each element.

Returns The minimum element in the sequence.

Return type int

Raises **TypeError** – if 'transform' is not callable

of_type(*of_type*)
Filters the elements based on the specified type.

Parameters **of_type** (*type*) – The type to keep.

Returns The elements of the sequence with the specified type.

Return type Queryable

Raises **TypeError** – if 'of_type' is not a type

order_by(*key_selector*)
Sorts the elements of the sequence in ascending order according to a key.

Parameters **key_selector** (*function*) – A function to extract a key from an element.

Returns The elements of the sequence sorted in ascending order.

Return type OrderedQueryable

Raises **TypeError** – if 'key_selector' is not callable

order_by_descending(*key_selector*)
Sorts the elements of the sequence in descending order according to a key.

Parameters **key_selector** (*function*) – A function to extract a key from an element.

Returns The elements of the sequence sorted in descending order.

Return type OrderedQueryable

Raises **TypeError** – if 'key_selector' is not callable

reverse()
Reverses the order of the elements in the sequence.

Returns The elements of the sequence in reverse order.

Return type `Queryable`

select (*selector*)
Returns the elements of the sequence after applying a transform function to each element.

Parameters **selector** (*function*) – A transform function to apply to each element.

Returns The elements of the sequence after applying the transform function.

Return type `Queryable`

Raises **TypeError** – if 'selector' is not callable

select_many (*selector*, *result_transform=<function select_i.<locals>.<lambda>>*)
Projects each element to a sequence and flattens the resulting sequences.

Parameters

- **selector** (*function*) – A function to transform each element into a sequence.
- **result_transform** (*function*) – (optional) A transform function for items of the selected sequence.

Returns A flattened sequence of transformed elements.

Return type `Queryable`

Raises

- **TypeError** – if 'selector' is not callable
- **TypeError** – if 'result_transform' is not callable

sequence_equal (*other*, *equality_comparer=<built-in function eq>*)
Determines whether two sequences are equal.

Parameters

- **other** (*Iterable*) – The sequence to compare elements to.
- **equality_comparer** (*function*) – (optional) The equality comparison function to use.

Returns True if the sequences are equal, false otherwise.

Return type bool

Raises

- **TypeError** – if 'other' is not an Iterable
- **TypeError** – if 'equality_comparer' is not callable

single (*predicate=<function true>*)
Returns the only element of the sequence.

Parameters **predicate** (*function*) – (optional) A function to test an element for a condition.

Returns The single element of the sequence satisfying the condition.

Raises

- **TypeError** – if 'predicate' is not callable
- **ValueError** – if the sequence is empty

- **ValueError** – if no element satisfies the condition
- **ValueError** – if more than one element satisfies the condition

single_or_default(*predicate=<function true>*, *default_value=None*)
Returns the only element of the sequence.

Parameters

- **predicate** (*function*) – (optional) A function to test an element for a condition.
- **default_value** – (optional) The default value to return if empty.

Returns The single element of the sequence satisfying the condition.

Raises

- **TypeError** – if 'predicate' is not callable
- **ValueError** – if more than one element satisfies the condition

skip(*num*)
Skips a specified number of elements in the sequence and returns the remaining elements.

Parameters **num** (*int*) – The number of elements to skip.

Returns A sequence containing the elements after position 'num'.

Return type Queryable

Raises **TypeError** – if 'num' is not an int

skip_while(*predicate*)
Skip elements of the sequence while the specified condition is true.

Parameters **predicate** (*function*) – The condition to check for.

Returns Elements of the sequence after the first item to fail the specified condition.

Return type Queryable

Raises **TypeError** – if 'condition' is not callable

sum(*transform=<function identity>*)
Computes the sum of the sequence by invoking a transform on each element.

Parameters **transform** (*function*) – (optional) A transform function to apply to each element.

Returns The sum of the elements of the sequence.

Return type int

Raises **TypeError** – if 'transform' is not callable

take(*num*)
Takes the specified number of elements from the start of the sequence.

Parameters **num** (*int*) – The number of elements to take.

Returns The specified number of elements from the start of the sequence.

Return type Queryable

Raises **TypeError** – if 'num' is not an int

take_while(*predicate*)
Takes elements from the start of the sequence while the specified condition holds.

Parameters **predicate** (*function*) – The condition to check for.

Returns The elements from the start of the sequence that satisfy the condition.

Return type Queryable

Raises **TypeError** – if 'predicate' is not callable

to_dict(*key_selector*, *value_selector=<function identity>*)
Creates a dictionary object according to the specified key selector function.

Parameters

- **key_selector** (*function*) – A function to extract the key for the dictionary entry.
- **value_selector** (*function*) – (optional) A function to extract the value for the dictionary entry.

Returns A dictionary of the elements in the sequence.

Return type dict

Raises

- **TypeError** – if 'key_selector' is not callable
- **TypeError** – if 'value_selector' is not callable

to_dictionary(*key_selector*, *value_selector=<function identity>*)
Creates a dictionary object according to the specified key selector function.

Parameters

- **key_selector** (*function*) – A function to extract the key for the dictionary entry.
- **value_selector** (*function*) – (optional) A function to extract the value for the dictionary entry.

Returns A dictionary of the elements in the sequence.

Return type dict

Raises

- **TypeError** – if 'key_selector' is not callable
- **TypeError** – if 'value_selector' is not callable

to_list()
Creates a list object from the sequence.

Returns A list of the elements in the sequence.

Return type list

union(*other*, *key_selector=<function identity>*)
Returns the set union of two sequences.

Parameters

- **other** (*Iterable*) – The second sequence to produce the union with.
- **key_selector** (*function*) – (optional) A function to extract a key for comparison.

Returns The set union of the two sequences.

Return type Queryable

Raises

- **TypeError** – if 'other' is not an Iterable
- **TypeError** – if 'key_selector' is not callable

where(*predicate*)
Filters the sequence of values based on the specified condition.

Parameters **predicate** – A function to check an element for a condition.

Returns The elements of the sequence that satisfy the condition.

Return type Queryable

Raises **TypeError** – if 'predicate' is not callable

zip(*other*, *result_transform*)
Applies a function to the corresponding elements of the two sequences.

Parameters

- **other** (*Iterable*) – The other input sequence.
- **result_transform** (*function*) – A function that combines the corresponding elements.

Returns A sequence of elements of the two sequences combined using 'result_transform'.

Return type Queryable

Raises

- **TypeError** – if 'other' is not an Iterable
- **TypeError** – if 'result_transform' is not callable

class pinq.queryable.**OrderedQueryable**(*iterator*, *keys*)
Bases: *pinq.queryable.Queryable*

A wrapper for ordered sequences.

then_by(*key_selector*)
Performs a subsequenct ordering on the elements of an ordered sequence.

Parameters **key** (*function*) – A function to extract a key to use for comparisons.

Returns The elements of the sequence in ascending order according to the key

Return type Queryable

Raises **TypeError** – if 'key_selector' is not callable

then_by_descending(*key_selector*)
Performs a subsequenct ordering on the elements of an ordered sequence.

Parameters **key** (*function*) – A function to extract a key to use for comparisons.

Returns The elements of the sequence in descending order according to the key

Return type Queryable

Raises **TypeError** – if 'key_selector' is not callable

p

U

W

Z