IMKInputController Class Reference

Cocoa > Internationalization

2007-06-06

Contents

IMKInputController Class Reference 5

Overview 5
Tasks 5
  Initializing an Input Controller 5
  Working with Ranges 6
  Managing the Delegate 6
  Getting the Client and Server Objects 6
  Tracking Selections 6
  Managing Composition 6
  Hiding the User Interface 7
  Working with Custom Commands 7
Instance Methods 7
  annotationSelected:forCandidate: 7
  cancelComposition 7
  candidateSelected: 8
  candidateSelectionChanged: 8
  client 8
  compositionAttributesAtRange: 9
  delegate; 9
  doCommandBySelector:commandDictionary: 9
  hidePalettes 10
  initWithServer:delegate:client: 10
  markForStyle:atRange: 11
  menu 12
  replacementRange 12
  selectionRange 12
  server 13
  setDelegate: 13
  updateComposition 13

Document Revision History 15

Index 17

IMKInputController Class Reference

Inherits from	NSObject
Conforms to	IMKMouseHandling IMKStateSetting NSObject (NSObject)
Framework	System/Library/Frameworks/InputMethodKit.framework
Availability	Available in Mac OS X v10.5 and later.
Declared in	IMKInputController.h
Related sample code	NumberInput_IMKit_Sample

Overview

The `IMKInputController` class provides a base class for custom input controller classes. The `IMKServer` class, which is allocated in the main function of an input method, creates an input controller object for each input session created by a client application. For every input session there is a corresponding `IMKInputController` object.

An `IMKInputController` object controls text input on the input method side. It manages events and text from the applications and converted text from the input method engine. `IMKInputController` implements fully the `IMKStateSetting` and `IMKMouseHandling` protocols. Typically you do not need to override this class, but you do need to provide a delegate object that implements the methods that your are interested in. The `IMKInputController` versions of the protocol methods check whether the delegate object implements a method, and calls the delegate version if it exists.

Tasks

Initializing an Input Controller

`– initWithServer:delegate:client:` (page 10)
Initializes the input control by setting the delegate.

Working with Ranges

- `compositionAttributesAtRange:` (page 9)
  Returns a dictionary of text attributes.
- `selectionRange` (page 12)
  Returns where the range of the selection that should be placed inside marked text.
- `replacementRange` (page 12)
  Returns the range in the client document that the text should replace.
- `markForStyle:atRange:` (page 11)
  Returns a dictionary of text attributes that can mark a range of an attributed string to send to a client.

Managing the Delegate

- `delegate;` (page 9)
  Returns the delegate for input controller object.
- `setDelegate:` (page 13)
  Sets the delegate for input controller object.

Getting the Client and Server Objects

- `server` (page 13)
  Returns the server object that manages the input controller.
- `client` (page 8)
  Returns the client object associated with the input controller.

Tracking Selections

- `annotationSelected:forCandidate:` (page 7)
  Sends the selected candidate string and annotation string to the input controller.
- `candidateSelectionChanged:` (page 8)
  Informs an input controller that the current candidate selection in the candidate window has changed.
- `candidateSelected:` (page 8)
  Informs an input controller that a new candidate is selected.

Managing Composition

- `updateComposition` (page 13)
  Informs the input controller that the composition has changed.
- `cancelComposition` (page 7)
  Stops the current composition and replaces marked text with the original text.

Hiding the User Interface

`- hidePalettes` (page 10)
: Informs an input method that it should close any visible user interface.

Working with Custom Commands

`- doCommandBySelector:commandDictionary:` (page 9)
: Passes commands that are not generated as part of the text input process.

`- menu` (page 12)
: Returns a menu of commands that are specific to an input method.

Instance Methods

annotationSelected:forCandidate:

Sends the selected candidate string and annotation string to the input controller.

```
- (void)annotationSelected:(NSAttributedString*)annotationString
    forCandidate:(NSAttributedString*)candidateString
```

Parameters

`annotationString`
: The annotation string associated with the candidate.

`candidateString`
: The candidate string that the user moved to.

Discussion
This method is called when the user moves to a candidate.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

cancelComposition

Stops the current composition and replaces marked text with the original text.

```
- (void)cancelComposition
```

Discussion
This method calls the method `originalString:` to obtain the original text and sends that text to the client using a call to the `IMKTextInput` protocol method `insertText:replacementRange:`

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

candidateSelected:

Informs an input controller that a new candidate is selected.

```
- (void)candidateSelected:(NSAttributedString*)candidateString
```

Parameters

`candidateString`
: The changed candidate string.

Discussion
The candidate object is the user’s final choice from the candidate window. The candidate window is closed before this method is called.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- candidateSelectionChanged:` (page 8)

Declared In
`IMKInputController.h`

candidateSelectionChanged:

Informs an input controller that the current candidate selection in the candidate window has changed.

```
- (void)candidateSelectionChanged:(NSAttributedString*)candidateString
```

Parameters

`candidateString`
: The changed candidate string.

Discussion
Note this method is called to indicate user activity in the candidate window. The candidate object might not be the user’s final selection.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- candidateSelected:` (page 8)

Declared In
`IMKInputController.h`

client

Returns the client object associated with the input controller.

```
- (<IMKTextInput, NSObject>))client
```

Return Value
The client object. The returned object is an autoreleased object.

Discussion
The client object conforms to the `IMKTextInput` protocol.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

compositionAttributesAtRange:

Returns a dictionary of text attributes.

```
- (NSMutableDictionary*) compositionAttributesAtRange:(NSRange)range
```

Parameters

`range`
The range of text whose attributes you want to obtain.

Return Value
The dictionary of text attributes. The default implementation returns an empty dictionary.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

delegate;

Returns the delegate for input controller object.

```
- (id)delegate
```

Return Value
The delegate object. The returned object is an autoreleased object.

See Also
`- setDelegate:` (page 13)

doCommandBySelector:commandDictionary:

Passes commands that are not generated as part of the text input process.

```
- (void)doCommandBySelector:(SEL)aSelector
    commandDictionary:(NSDictionary*)infoDictionary
```

Parameters

aSelector

A selector that represents a command from the text input menu.

infoDictionary

A dictionary that contains two key-value pairs:

- `kIMKCommandMenuItemName`, whose value is an `NSMenuItem` object. That is, the item selected by the user.
- `kIMKCommandClientName`, whose value is the current client—`id<IMKTextInput, NSObject>`.

Discussion

The default implementation checks if the input controller object (that is, self) responds to the selector. If so, it sends the message `performSelector:withObject:` to the input controller class. The object parameter in that case is the `infoDictionary` parameter.

This method is called when a user selects a command from the text input menu. To support this, an input method must provide actions for each menu item that is placed in the menu. For example, `(void)menuAction:(id)sender`. Note that the sender in this instance is the info dictionary.

Availability

Available in Mac OS X v10.5 and later.

See Also

`– menu` (page 12)

Declared In

`IMKInputController.h`

hidePalettes

Informs an input method that it should close any visible user interface.

```
- (void)hidePalettes
```

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IMKInputController.h`

initWithServer:delegate:client:

Initializes the input control by setting the delegate.

```
- (id)initWithServer:(IMKServer*)server delegate:(id)delegate client:(id)inputClient
```

Parameters

server

The server object for the controller.

delegate

The delegate object.

inputClient
: The client object that will send messages to the controller using the server object. The client object must confirm to the `IMKTextInput` protocol.

Return Value
The initialized input controller object.

Discussion
Methods in the `IMKStateSetting` and `IMKMouseHandling` protocols that are implemented by the delegate object always include a client parameter. Methods in the `IMKInputController` class do not need to take a client because the `initWithServer:delegate:client:` method stores the client object you supply as an ivar when it initializes the `IMKInputController` object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

markForStyle:atRange:

Returns a dictionary of text attributes that can mark a range of an attributed string to send to a client.

```
- (NSDictionary*)markForStyle:(NSInteger)style atRange:(NSRange)range
```

Parameters

style
: A style, which should be one of the following values: `kTSMHiliteSelectedRawText`, `kTSMHiliteConvertedText`, or `kTSMHiliteSelectedConvertedText`. See the `AERegistry.h` header file for the definition of these values.

range
: The range (that is, a clause) to mark.

Return Value
The dictionary of text attributes. The returned object should be an autoreleased object.

Discussion
This utility function can be called by input methods to mark each range (i.e. clause) of marked text. T

The default implementation first calls the method `compositionAttributesAtRange:` (page 9) to obtain the additional attributes that an input method wants to include, such as font or glyph information. Then, it adds the appropriate underline and underline color information to the attributes dictionary for the style parameter. Finally it adds the style value as the dictionary value. The key for the style value is `NSMarkedClauseSegmentAttributeName`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

menu

Returns a menu of commands that are specific to an input method.

```
- (NSMenu*)menu
```

Return Value
The menu object. This object is an autoreleased object.

Discussion
This method is called whenever the menu needs to be drawn so that an input method can update the menu to reflect the current state.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- doCommandBySelector:commandDictionary:` (page 9)

Related Sample Code
NumberInput_IMKit_Sample

Declared In
`IMKInputController.h`

replacementRange

Returns the range in the client document that the text should replace.

```
- (NSRange)replacementRange
```

Return Value
The range to replace.

Discussion
This method is called by `updateComposition` (page 13) to obtain the range in the client document where marked text should be placed. The default implementation returns an `NSRange` object whose location and length are `NSNotFound`. That indicates that the marked text should be placed at the current insertion point. Input methods that insert marked text somewhere other than at the current insertion point should override this method.

An example of an input method that might override this method would be one replaces words with synonyms. That input method would watch for certain words and when it detects such a word it would replaced the word by marked text that was a synonym of the word.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

selectionRange

Returns where the range of the selection that should be placed inside marked text.

```
- (NSRange)selectionRange
```

Return Value
The range of the selection. This object should be an autoreleased object.

Discussion
This method is called by `updateComposition` (page 13) to obtain the selection range for marked text. The default implementation sets the selection range at the end of the marked text. You should override this method if your input method provides font or glyph information.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

server

Returns the server object that manages the input controller.

```
- (IMKServer*)server
```

Return Value
The server object. The returned object is an autoreleased object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`IMKInputController.h`

setDelegate:

Sets the delegate for input controller object.

```
- (void)setDelegate:(id)newDelegate
```

Parameters

`newDelegate`
The delegate object to set.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- delegate;` (page 9)

Declared In
`IMKInputController.h`

updateComposition

Informs the input controller that the composition has changed.

```
- (void)updateComposition
```

Discussion

This method calls the protocol method `composedString:` to obtain the current composition. The current composition is sent to the client by a call to the method `setMarkedText:selectionRange:replacementRange:`.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`IMKInputController.h`

Document Revision History

This table describes the changes to *IMKInputController Class Reference*.

Date	Notes
2007-06-06	New document that describes the class that controls input on the input method side.

Index

A

annotationSelected:forCandidate: instance method 7

C

cancelComposition instance method 7
candidateSelected: instance method 8
candidateSelectionChanged: instance method 8
client instance method 8
compositionAttributesAtRange: instance method 9

D

delegate; instance method 9
doCommandBySelector:commandDictionary: instance method 9

H

hidePalettes instance method 10

I

initWithServer:delegate:client: instance method 10

M

markForStyle:atRange: instance method 11
menu instance method 12

R

replacementRange instance method 12

S

selectionRange instance method 12
server instance method 13
setDelegate: instance method 13

U

updateComposition instance method 13