StandardController Class

Use a StandardController when defining an extension for a standard controller.

Namespace

ApexPages

Usage

StandardController objects reference the pre-built Visualforce controllers provided by Salesforce. The only time it is necessary to refer to a StandardController object is when defining an extension for a standard controller. StandardController is the data type of the single argument in the extension class constructor.

Instantiation

You can instantiate a StandardController in the following way:

```
ApexPages.StandardController sc = new ApexPages.StandardController(sObject);
```

Example

The following example shows how a StandardController object can be used in the constructor for a standard controller extension:

```
public class myControllerExtension {

    private final Account acct;

    // The extension constructor initializes the private member
    // variable acct by using the getRecord method from the standard
    // controller.
    public myControllerExtension(ApexPages.StandardController stdController) {
        this.acct = (Account)stdController.getRecord();
    }

    public String getGreeting() {
        return 'Hello ' + acct.name + ' (' + acct.id + ')';
    }
}
```

The following Visualforce markup shows how the controller extension from above can be used in a page:

```
<apex:page standardController="Account" extensions="myControllerExtension">
    {!greeting} <p/>
    <apex:form>
```

StandardController Constructors

The following are constructors for `StandardController`.

- **StandardController(controllerSObject)**
  Creates a new instance of the `ApexPages.StandardController` class for the specified standard or custom object.

StandardController(controllerSObject)

Creates a new instance of the `ApexPages.StandardController` class for the specified standard or custom object.

Signature

```
public StandardController(SObject controllerSObject)
```

Parameters

controllerSObject

Type: SObject

A standard or custom object.

StandardController Methods

The following are methods for `StandardController`. All are instance methods.

- **addFields(fieldNames)**
  When a Visualforce page is loaded, the fields accessible to the page are based on the fields referenced in the Visualforce markup. This method adds a reference to each field specified in `fieldNames` so that the controller can explicitly access those fields as well.
- **cancel()**
  Returns the PageReference of the cancel page.
- **delete()**
  Deletes record and returns the PageReference of the delete page.
- **edit()**
  Returns the PageReference of the standard edit page.
- **getId()**
  Returns the ID of the record that is currently in context, based on the value of the `id` query string parameter in the Visualforce page URL.
- **getRecord()**
  Returns the record that is currently in context, based on the value of the `id` query string parameter in the Visualforce page URL.
- **reset()**
  Forces the controller to reacquire access to newly referenced fields. Any changes made to the record prior to this method call are discarded.
- **save()**
  Saves changes and returns the updated PageReference.
- **view()**
  Returns the PageReference object of the standard detail page.

addFields(fieldNames)

```
public Void addFields(List<String> fieldNames)
```

Parameters

fieldNames

Type: List<String>

Return Value

Type: Void

Usage

This method should be called before a record has been loaded–typically, it's called by the controller's constructor. If this method is called outside of the constructor, you must use the `reset()` method before calling `addFields()`.

The strings in `fieldNames` can either be the API name of a field, such as AccountId, or they can be explicit relationships to fields, such as `something__r.myField__c`.

This method is only for controllers used by dynamicVisualforce bindings.

cancel()

Returns the PageReference of the cancel page.

Signature

```
public System.PageReference cancel()
```

Return Value

Type: System.PageReference

delete()

Deletes record and returns the PageReference of the delete page.

Signature

```
public System.PageReference delete()
```

Return Value

Type: System.PageReference

edit()

Returns the PageReference of the standard edit page.

Signature

```
public System.PageReference edit()
```

Return Value

Type: System.PageReference

getId()

```
public String getId()
```

Return Value

Type: String

getRecord()

Returns the record that is currently in context, based on the value of the `id` query string parameter in the Visualforce page URL.

Signature

```
public SObject getRecord()
```

Return Value

Type: sObject

Usage

Note that only the fields that are referenced in the associated Visualforce markup are available for querying on this SObject. All other fields, including fields from any related objects, must be queried using a SOQL expression.

> **Tip**
>
> You can work around this restriction by including a hidden component that references any additional fields that you want to query. Hide the component from display by setting the component's `rendered` attribute to `false`.

Example

```
<apex:outputText
value="{!account.billingcity}
{!account.contacts}"
rendered="false"/>
```

reset()

Forces the controller to reacquire access to newly referenced fields. Any changes made to the record prior to this method call are discarded.

Signature

```
public Void reset()
```

Return Value

Type: Void

Usage

This method is only used if `addFields` is called outside the constructor, and it must be called directly before `addFields`.

This method is only for controllers used by dynamicVisualforce bindings.

save()

Return Value

Type: System.PageReference

view()

Returns the PageReference object of the standard detail page.

Signature

```
public System.PageReference view()
```

Return Value

Type: System.PageReference

DID THIS ARTICLE SOLVE YOUR ISSUE?
Let us know so we can improve!

Share your feedback