PageReference Class

A PageReference is a reference to an instantiation of a page. Among other attributes, PageReferences consist of a URL and a set of query parameter names and values.

Namespace

System

Use a PageReference object:

- To view or set query string parameters and values for a page
- To send the user to a different page as the result of an action method

Instantiation

In a custom controller or controller extension, you can refer to or instantiate a PageReference in one of these ways.

- ```
  Page.existingPageName
  ```

  Refers to a PageReference for a Visualforce page that has already been saved in your organization. By referring to a page in this way, the platform recognizes that this controller or controller extension is dependent on the existence of the specified page and will prevent the page from being deleted while the controller or extension exists.

- ```
  PageReference pageRef = new PageReference('partialURL');
  ```

  Creates a PageReference to any page that is hosted on the Lightning platform. For example, setting *'partialURL'* to `'/apex/HelloWorld'` refers to the Visualforce page located at `http://mySalesforceInstance/apex/HelloWorld`. Likewise, setting `'partialURL'` to `'/' + 'recordID'` refers to the detail page for the specified record.

  This syntax is less preferable for referencing other Visualforce pages than `Page.existingPageName` because the PageReference is constructed at runtime, rather than referenced at compile time. Runtime references are not available to the referential integrity system. Consequently, the platform doesn't recognize that this controller or controller extension is dependent on the existence of the specified page and won't issue an error message to prevent user deletion of the page.

- ```
  PageReference pageRef = new PageReference('fullURL');
  ```

  Creates a PageReference for an external URL. For example:

  ```
  PageReference pageRef = new PageReference('http://www.google.com');
  ```
  ```

```
PageReference pageRef = ApexPages.currentPage();
```

Request Headers

Here's a non-exhaustive list of headers that are set on requests.

Header	Description
Host	The host name requested in the request URL. This header is always set on Lightning Platform Site requests and My Domain requests. This header is optional on other requests when HTTP/1.0 is used instead of HTTP/1.1.
Referer	The URL that is either included or linked to the current request's URL. This header is optional.
User-Agent	The name, version, and extension support of the program that initiated this request, such as a web browser. This header is optional and can be overridden in most browsers to be a different value. Therefore, this header can't be relied upon.
CipherSuite	If this header exists and has a non-blank value, this means that the request is using HTTPS. Otherwise, the request is using HTTP. The contents of a non-blank value are not defined by this API, and can be changed without notice.
X-Salesforce-SIP	The source IP address of the request. This header is always set on HTTP and HTTPS requests that are initiated outside of Salesforce's data centers. **Note** If a request passes through a content delivery network (CDN) or proxy server, the source IP address might be altered, and no longer the original client IP address.
X-Salesforce-Forwarded-To	The fully qualified domain name of the Salesforce instance that is handling this request. This header is always set on HTTP and HTTPS requests that are initiated outside of Salesforce's data centers.

Example: Retrieving Query String Parameters

This example shows how to use a PageReference object to retrieve a query string parameter in the current page URL. In this example, the `getAccount` method references the `id` query string parameter.

```
public with sharing class MyController {
   public Account getAccount() {
        return [SELECT Id, Name FROM Account WITH USER_MODE
                WHERE Id = :ApexPages.currentPage().getParameters().get('Id')];
    }
}
```

This page markup calls the `getAccount` method from that controller.

Note

For this example to render properly, you must associate the Visualforce page with a valid account record in the URL. For example, if `001D000000IRt53` is the account ID, the resulting URL should be:

```
https://Visualforce_Url/apex/MyFirstPage?id=001D000000IRt53
```

Replace *Visualforce_URL* with the Visualforce URL for your org. For production, this URL is in the format *`MyDomainName--PackageName`*`.vf.force.com`, and if your installed package is unmanaged, the package name is `c`. For more information on the format of the URLs that Salesforce serves for your org, see My Domain Login and Application URL Formats and Partitioned Domains in Salesforce Help.

The `getAccount` method uses an embedded SOQL query to return the account specified by the `id` parameter in the URL of the page. To access `id`, the `getAccount` method uses the `ApexPages` namespace.

- First the `currentPage` method returns the `PageReference` instance for the current page. `PageReference` returns a reference to a Visualforce page, including its query string parameters.
- Using the page reference, use the `getParameters` method to return a map of the specified query string parameter names and values.
- Then a call to the `get` method specifying `id` returns the value of the `id` parameter itself.

Example: Navigating to a New Page as the Result of an Action Method

Any action method in a custom controller or controller extension can return a PageReference object as the result of the method. If the `redirect` attribute on the PageReference is set to `true`, the user navigates to the URL specified by the PageReference.

This example shows how this can be implemented with a `save` method. In this example, the PageReference returned by the `save` method redirects the user to the detail page for the account record that was just saved.

```
public class mySecondController {
    Account account;

    public Account getAccount() {
        if(account == null) account = new Account();
        return account;
    }

    public PageReference save() {
        // Add the account to the database.
        insert account;
        // Send the user to the detail page for the new account.
        PageReference acctPage = new ApexPages.StandardController(account).view();
        acctPage.setRedirect(true);
        return acctPage;
    }
}
```

```
<apex:page controller="mySecondController" tabStyle="Account">
    <apex:sectionHeader title="New Account Edit Page" />
    <apex:form>
        <apex:pageBlock title="Create a New Account">
            <apex:pageBlockButtons location="bottom">
                <apex:commandButton action="{!save}" value="Save"/>
            </apex:pageBlockButtons>
            <apex:pageBlockSection title="Account Information">
                <apex:inputField id="accountName" value="{!account.name}"/>
                <apex:inputField id="accountSite" value="{!account.site}"/>
            </apex:pageBlockSection>
       </apex:pageBlock>
    </apex:form>
</apex:page>
```

Example: Redirect Users to a Replacement Experience Cloud Site

This example shows how to redirect a user attempting to access a retired feedback site to a self-service help site. If the `redirect` attribute is set to `true` on the PageReference for the feedback site, the user navigates to the URL specified by the PageReference. The `redirectCode` attribute defines the redirection type for search engine optimization in public Experience Cloud sites.

```
public class RedirectController {
    // Redirect users to the self-service help site    public PageReference redirect() {
        final PageReference target = new
        PageReference(Site.getBaseSecureUrl() + '/SiteLogin');
        target.setRedirect(true);
        // This is a permanent redirection
        target.setRedirectCode(301);
        return target;
    }
}
```

This example shows how to call the RedirectController class from the retired site page.

```
<apex:page controller="RedirectController" action="{!redirect}"/>
```

> **Note**
>
> To redirect a page that's served by a third-party CDN, configure that CDN to pass the origin IP address via the `true-client-ip` HTTP header on the page. For more information, see Prerequisites for a Custom Domain That Uses a Third-Party Service or CDN in Salesforce Help.

- PageReference Constructors
- PageReference Methods

MuleSoft
Tableau
Commerce Cloud
Lightning Design System
Einstein
Quip

Component Library
APIs
Trailhead
Sample Apps
Podcasts
AppExchange

Events and Calendar
Partner Community
Blog
Salesforce Admins
Salesforce Architects