|  |  |
| --- | --- |
| 14 down vote accepted | In simple words it prevents external get requests. So, nobody can use your methods from other sites.  How it works. You are having AntiForgeryToken in your Html.BeginForm in View.  @using (Html.BeginForm()){  @Html.AntiForgeryToken()  //\*\* fields of form  }  When you submit form, you sends data to your Controller method. If method has ValidateAntiForgeryToken attribute, it validates if data you are sending has your ForgeryToken.  [ValidateAntiForgeryToken]  public ViewResult Update()  {  }  ForgeryToken is generated once per session. |

This is to prevent a Cross-Site Request Forgery (CSRF). It's pretty standard behavior to click 'Save' sumbit a form and perform some action on the server, i.e. save a user's details. How do you know the user submitting the form is the user they claim to be? In most cases you'd use some cookie or windows based auth.

What if an attacker lures you to a site which submits exactly the same form in a little hidden IFRAME? Your cookies get submitted intact and the server doesn't see the request as any different to a legit request. (As gmail has discovered: <http://www.gnucitizen.org/blog/google-gmail-e-mail-hijack-technique/>)

The anti-forgery token prevents this form of attack by creating a additional cookie token everytime a page is generated. The token is both in the form and the cookie, if the form and cookie don't match we have a CSRF attack (as the attacker wouldn't be able to read the anti-forgery token using the attack described above).