ASP.NET and SQL Injection

How to protect ASP.NET application from SQL Injection vulnerabilities? The first step we can take to protect our software is to understand the SQL Injection vulnerability problem. SQL Injection is a vulnerability that allows hackers to inject malicious code into your SQL script. More info about SQL Injection may be found in my SQL Injection and PHP article.

How to prevent SQL Injection in ASP.NET applications? I do not recommend you to use regular expressions to prevent users or hackers from entering characters that may allow them to break into the database. It is not the best practice in ASP.NET application. Do not try to remove any symbols from the parameters with the data received from the WEB site users. The best practice is to use **parameterized SQL queries**. You have to use parameters in you queries!

Let's take a look at the next example that use parameterized query to prevent SQL Injection attack on my ASP.NET application:

```
string TagName = Request.QueryString["tag"];

SqlCommand blogRecord = new SqlCommand(
     "SELECT * FROM blog WHERE tagname = @tagname", connection
);

blogRecord.Parameters.Add("@tagname",
    SqlDbType.NChar).Value = TagName;

SqlDataReader contentReader = blogRecord.ExecuteReader();
```

It is not the real query from the ProfWebDev.com blog source code :). It is almost equal to my real application code. I changed the table and column names to unreal. I hope you are not a hacker, but I don't want to tell my real database objects names.

In the example I use parameter in the SQL query "SELECT * FROM blog WHERE tagname = *@tagname*". In the next line of code I create parameter and set the value to it. Now I can use SqlCommand to get data without fair SQL Injection. The parameter *tag* I get from *Request.QueryString* may contain any characters.

Using parameters is an easiest and powerful method to protect your WEB application from SQL Injection. I recommend you to use it in ASP.NET applications. Do not try to use regular expressions or string concatenations.