

[<< Back to Dashboard](#)

## Contents

- [ColdFusion SQL Injection Protection Best Practices](#)
  - [Introduction](#)
  - [Understanding The HTTP Methods](#)
  - [Protecting Your Code](#)
  - [Golden Rule of Web Applications](#)
  - [Related Guides](#)

# ColdFusion SQL Injection Protection Best Practices

## Introduction

This section delineates some best practices when dealing with SQL injection attempts or just plain old URL/FORM variable manipulations, when building ColdFusion web applications.

## Understanding The HTTP Methods

First of all, you also need to understand what a POST, GET, DELETE, PUT are used for. The method attribute of the FORM element specifies the HTTP method used to send the form to the processing agent. This attribute may take the following values:

- **GET:** With the HTTP *GET* method, the form data set is appended to the URI specified by the action attribute (with a question-mark ("?") as separator) and this new URI is sent to the processing agent.
- **POST,PUT,DELETE:** The form data set is included in the body of the form and sent to the processing agent, with expectations of either a save, update or delete.

The *GET* method should be used when the form is idempotent (i.e., causes no side-effects). Many database searches have no visible side effects and make ideal applications for the *GET* method. *If the service associated with the processing of a form causes side effects (for example, if the form modifies a database or subscription to a service), the POST,PUT, or DELETE method should be used.*

The most important fact is that the *GET* method should be used for idempotent transactions. Here is the definition for idempotent:

"Idempotent operation means that it can be repeated without causing any errors or inconsistencies if the operation is carried out once or many times".

Thanks to Roger Benningfield:

"Allowing GET requests to change the state of server resources can be a very dangerous game, without so much as a whiff of malicious behavior. An app that allows clients to change or delete data just by fetching a URI is asking for trouble in 2006."

There is a time for using *GET* and a time for using *POST,PUT,DELETE* and **ALL OF THEM should not trust the client data.**

## Protecting Your Code

As for security, FORM variables are just as easy to modify as URL variables. However, there are several ways to protect from attacks, SQL injection or plain mischief:


1. `<CFPARAM>` tag with strict data-type and scaling will prevent URL and FORM variable abuse. This tag binds incoming FORM or URL variables to specific types and even create default values for them.
2. Use of `<CFQUERYPARAM>` is the most common form of stopping URL/FORM variable abuse with direct SQL and it makes your queries faster as well (If your db supports it). Use this tag to protect any part of your SQL statements that come from an external source, even *ORDER BY* or *GROUP BY* statements.
3. Get into the habit of adding the *maxlength* attribute to *cfqueryparam*, to limit the number of characters/digits to bind a column with.
4. Use of Stored Procedures will prevent SQL injection as all variables are binded by the use of `<CFPROCPARAM>`.
5. Try to always use a custom error page that cannot display to the user the entire error message. This will hide your internal information and encapsulate your errors.
6. It is the role of the developer to protect the database and its contents, no matter if you are using a POST or a GET.
7. Do not trust the client on the incoming data; always do data type checking and authorization/authentication checking on the server-side.

## Golden Rule of Web Applications

Never ever ever trust the incoming data. It is YOUR responsibility to protect your code.

## Related Guides

- [Model-View-Controller Demystified](#)
- [ColdFusion Development Best Practices](#)
- [Database Naming Conventions](#)

 Categories:

- [theory](#)