  - Utilizzare DataReader per recuperare i dati e caricarli in un DataSet. Non passare questo DataSet tra i diversi livelli. Passare entità personalizzate tra i diversi livelli.
- **Resource**: Chiusura delle risorse. Una delle problematiche più comuni ai programmatori è la sistematica chiusura delle risorse e/o connessioni aperte. Capita spesso, infatti, che per errori e/o eccezioni impreviste non gestite al meglio, alcune risorse rimangano in attesa di una chiusura che non arriverà mai. Per cui, chiudere le connessioni quando non in uso, migliora la sicurezza e le prestazioni. Prevedere meccanismi di chiusura automatica delle risorse (attraverso un gestore che viene eseguito ad ogni uscito dal blocco).
- **Inizialiazzazione delle variabili**. Inizializzare la variabile @ start e usarla in una fase successiva provoca molte operazioni PUSH / POP. Quindi, inizializzare le variabili al momento/posto giusto. Le variabili intere non devono essere inizializzate a zero perché vengono inizializzate automaticamente. Le variabili stringhe invece, devono essere inizializzate esplicitamente.
- **Richiesta http**: Utilizzare Fiddler. Utilizza Fiddler per intercettare le richieste HTTP e per sapere quale richiesta richiede più tempo. Individua anche le eccezioni causate durante ogni richiesta HTTP.
- **URL**: Rewriting URL. Per gli URL che dispongono di informazioni riservate, è consigliabile implementare URL Rewriters. Gli URL devono essere coerenti.
- **Settings**: Application settings.
  - Fissare un valore per la Content-Length. Questo impone la connessione aperta per un tempo limitato (prestabilito) e la chiusura automatica della stessa quando viene superato il limite temporale dichiarato.
  - Crittografare le stringhe di connessione sul server.
  - Assicurarsi che tutte le DLL di riferimento siano presenti nel GAC.
  - Disattivare il tracciamento e il debug. Set <retail = "true" /> nel file machine.config - obbliga il debug a essere falso, disattiva la traccia di output e reindirizza alla pagina di errore personalizzata piuttosto che alla pagina di errore effettiva.
- **Web services**: Impedire il sovraccarico dei servizi web
  - Impedire il sovraccarico dei servizi web tramite attacchi DoS (Denial of Service):
  - Controllare se si tratta di una prima visita o la visita ripetuta per la stessa funzione dal medesimo IP.
  - Utilizzare connessioni SQL attendibili nei servizi Web.
  - Assicurarsi che ci siano chiamate asincrone ai servizi web.
- **Eccezioni**: Gestire le eccezioni
  - Registrare le eccezioni e visualizzare il messaggio appropriato all'utente. Definire una classe base MyException. La classe deve definire:
  - Informazioni utili per l'utente: Cosa è successo; Cosa è stato colpito; Quali sono le azioni da intraprendere; Altre iInformazioni di supporto.
  - Informazioni utili per la registrazione dell'eccezione: Nome del server, Istanza id, ID utente, Stack di chiamata, Nome Assembly & Versione, Fonte, tipo e messaggio di eccezione, Redirect secondo il livello di errore, Livello di applicazione (Cattura errori in global.asax nella funzione Application_Error), Livello di pagina (Utilizza la funzione Page_Error per registrare gli errori).

7.8.9.2 ASP.NET MVC

ASP.NET MVC è una parte del framework .NET che permette di creare siti scalabili suddividendo la logica di programmazione in base al metodo Model-View-Controller. Segue un elenco di best practices per lo sviluppo sicuro:

- Isolate Controllers. Isolare i controllers dalle dipendenze, da HttpContext, dalle classi di accesso ai dati, dalla configurazione, dalla registrazione, ecc. L'isolamento può essere ottenuto creando classi di wrapper e utilizzando un contenitore IoC (Inversion of Control).