- Utilizzare gli IoC Container per gestire tutte le dipendenze esterne. Di seguito sono riportati alcuni dei contenitori / framework: Ninject, autofac, structureMap, Unity block, Castle Windsor.
- Creare ViewModel per ogni View. Creare un ViewModel specifico per ogni visualizzazione. Il ruolo del ViewModel dovrebbe interessare solo il binding di dati e non dovrebbe contenere alcuna logica.
- Utilizzare HtmlHelper. Per generare view html utilizzare HtmlHelper. Se l'attuale HtmlHelper non è sufficiente estenderlo utilizzando i metodi di estensione. Questo manterrà la progettazione controllata.
- Decorare action methods con verbi appropriati come Get o Post, se applicabile.
- Utilizzare l'attributo OutputCache.
- Decorare gli action methods più utilizzati con OutputCache attribute.
- Controller e Domain logic. Cercare di separare il controller dal dominio logico. Il controller deve essere responsabile solo delle seguenti funzioni:
    - convalidare l'input;
    - ottenere i dati relativi alla view dal modello;
    - ritornare la view appropriata o reindirizzare ad un altro metodo di azione appropriato.
- Utilizzare il modello Post-Redirect-Get. Il modello PRG viene utilizzato per evitare l'avvio del browser classico quando si aggiorna una pagina dopo il POST. Ogni volta che fai una richiesta POST, una volta completata la richiesta, effettua un reindirizzamento. In questo modo, quando l'utente aggiorna la pagina, verrà eseguita l'ultima richiesta GET piuttosto che il POST. Questo consente di evitare problemi di usabilità non necessari e impedisce che la richiesta iniziale venga eseguita due volte evitando così possibili problemi di duplicazione.
- View e presentation logic: la View non deve contenere presentation logic. Non ci dovrebbe essere alcuna logica di dominio nelle viste. Le viste devono essere solamente responsabili della visualizzazione dei dati. Per esempio se un pulsante "Elimina" deve essere visualizzato solo dall'utente con ruolo "Amministratore", ciò dovrebbe essere estratto in un helper HTML.

7.9 PHP

PHP è un linguaggio di scripting lato server, progettato per lo sviluppo web ma anche usato come linguaggio di programmazione generico. Originariamente creato da Rasmus Lerdorf nel 1994, PHP è ora distribuito da The PHP Group. PHP originariamente significava "home page personale", ma ora è l'acronimo ricorsivo PHP: Hypertext Preprocessor.
Il codice PHP può essere incorporato nel codice HTML oppure può essere utilizzato in combinazione con vari sistemi di modelli Web, sistemi di gestione dei contenuti Web e framework web. Il codice PHP viene solitamente elaborato da un interprete PHP implementato come modulo nel web server. Il codice PHP può essere ancora incorporato nel codice HTML, ma più frequentemente può essere utilizzato in combinazione con vari sistemi di modelli Web, sistemi di gestione dei contenuti Web e framework. Il codice PHP viene solitamente elaborato da un interprete PHP implementato come modulo nel web server.

Segue un elenco delle principali vulnerabilità e contromisure da adottare.

7.9.1 Cross-site scripting (XSS)

Come riconoscerla

Il Cross Site Scripting consiste nella possibilità che un attaccante possa inserire nella pagina dell'applicazione, quindi nel codice HTML, script che, una volta eseguiti, possano trarre in inganno i legittimi utenti, trafugare informazioni e predisporre nuovi attacchi.
Questa minaccia, enormemente diffusa, è dovuta allo scarso controllo dell'input da parte delle web application.