[ASP.NET](http://ASP.NET) MVC Fundamentals

Action Results



|  |  |
| --- | --- |
| **Type** | **Helper Method** |
|  |  |
| ViewResult | **View()** |
|  |  |
| PartialViewResult | **PartialView()** |
|  |  |
| ContentResult | **Content()** |
|  |  |
| RedirectResult | **Redirect()** |
|  |  |
| RedirectToRouteResult | **RedirectToAction()** |
|  |  |
| JsonResult | **Json()** |
|  |  |
| FileResult | **File()** |
|  |  |
| HttpNotFoundResult | **HttpNotFound()** |
|  |  |
| EmptyResult |  |
|  |  |



Action Parameters

Sources

* Embedded in the URL: /movies/edit/1
* In the query string: /movies/edit?id=1
* In the form data

!1

[ASP.NET](http://ASP.NET) MVC Fundamentals

Convention-based Routes

routes.MapRoute(

“MoviesByReleaseDate”,

“movies/released/{year}/{month}”,

new {

controller = “Movies”,

action = “MoviesReleaseByDate”

},

new {

year = @“\d{4}, month = @“\d{2}”

}isFavorite = false;

}

Attribute Routes

[Route(“movies/released/{year}/{month}”)

public ActionResult MoviesByReleaseDate(int year, int month)

{

}

To apply a constraint use a colon:

month:regex(\\d{2}):range(1, 12)

!2

[ASP.NET](http://ASP.NET) MVC Fundamentals

Passing Data to Views

Avoid using ViewData and ViewBag because they are fragile. Plus, you have to do extra casting, which makes your code ugly. Pass a model (or a view model) directly to a view:

return View(movie);

Razor Views

@if (…)

{

// C# code or HTML

}

@foreach (…)

{

}

Render a class (or any attributes) conditionally:

@{

var className = Model.Customers.Count > 5 ? “popular” : null;

}

<h2 class=“@className”>…</h2>

!3

[ASP.NET](http://ASP.NET) MVC Fundamentals

Partial Views

To render:

@Html.Partial(“\_NavBar”)

!4