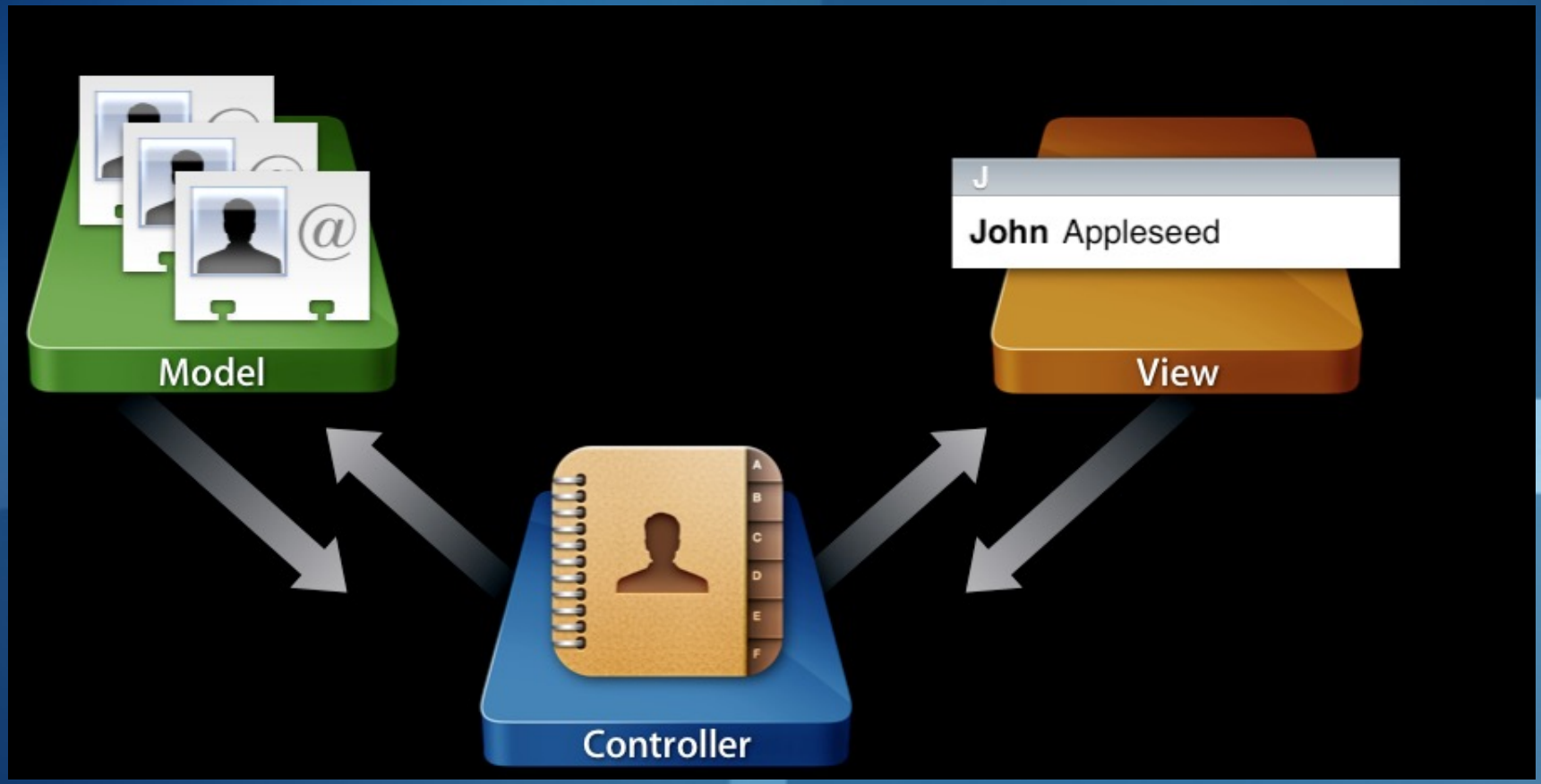
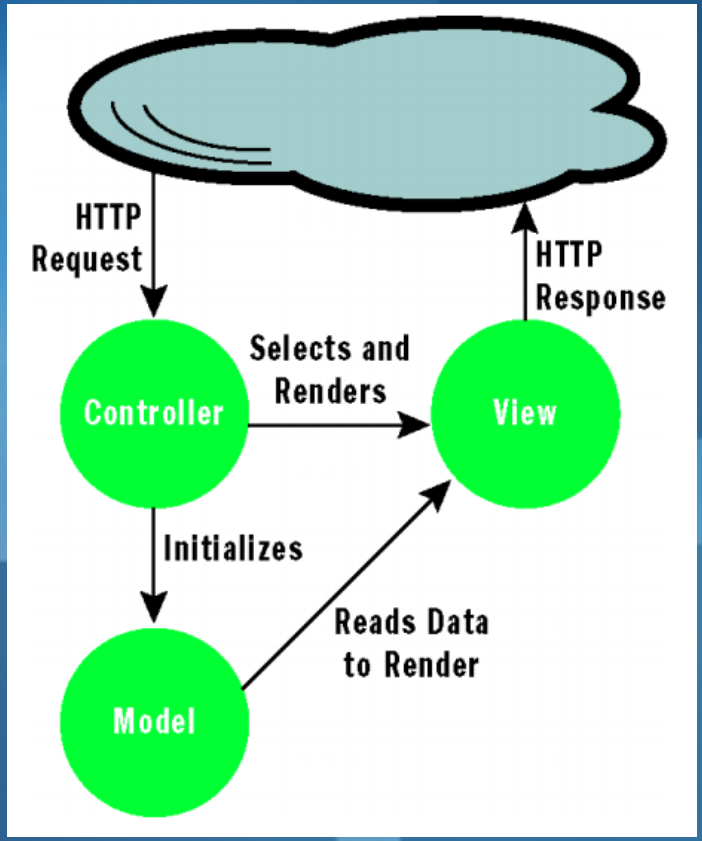
# Introducción a MVC

# ¿Qué es MVC?



# ¿Qué es MVC?

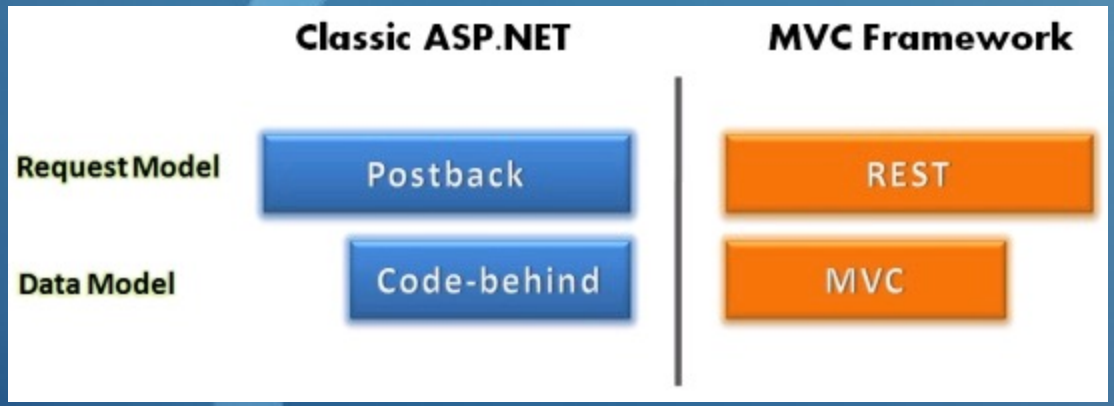

HTTP
Request
HTTP
Response
Selects and
Renders
Controller
View
Initializes
Reads Data
to Render
Model

# MVC Hoy

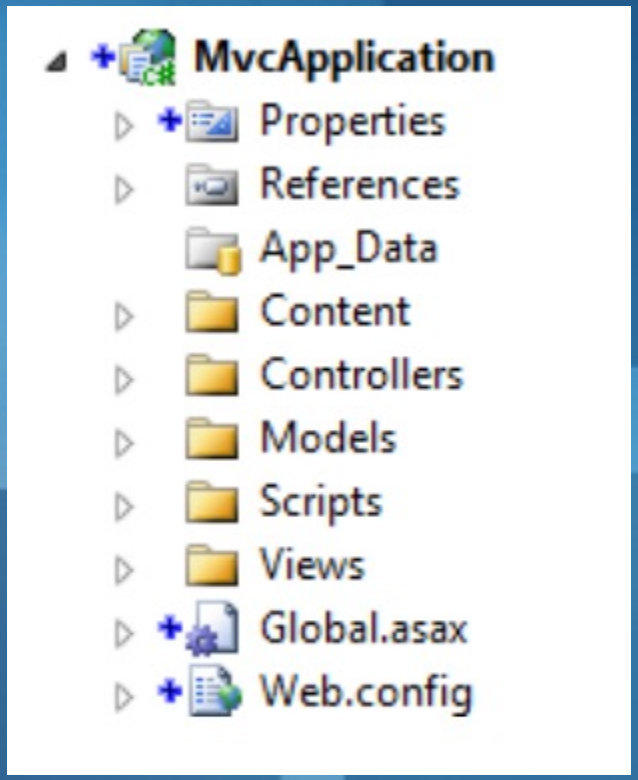
- Ruby on Rails
- Django - Python
- Spring MVC
- Zend Framework - PHP

# ASP.NET MVC

- No + Web Forms
- No + Form Controls
- No + ViewState
- No + Eventos
- No + Ajax ½



# ASP.NET MVC


MvcApplication
Properties
References
App_Data
Content
Controllers
Models
Scripts
Views
Global.asax
Web.config

# Controladores

- Son clases que contienen acciones (métodos)
- Siempre tienen el nombre controller

```
public class HomeController : Controller
{
        public ActionResult Index()
        {
            ViewBag.Message = "Welcome to ASP.NET MVC!";

            return View();
        }

        public ActionResult About()
        {
            return View();
        }
}
```

# Tipos de ActionResult

```
public class HomeController : Controller
{
        public ActionResult Index() …
```

- Subtipos de ActionResult
  - ViewResult
  - EmptyResult
  - RedirectResult
  - JsonResult
  - JavaScriptResult
  - FileStreamResult
  - .....

# Tipos de ActionResult

```
public ActionResult Ejemplo()
{
    if (…….)
      return View("nombrevista");
    else
      return Json(misdatosjson);
}
```

```
public ActionResult Ejemplo()
{
    ……
    return RedirectToAction("Index", "Home");
}
```

# Vistas

# Vistas

- Trabajan con ViewData / ViewBag / Vistas tipadas + ViewModel / TempData / Session / Application
- JQuery + Ajax (no + update panel)
- Partial Views
- Html helper class

# Vistas

```
public ActionResult Index()
{
    ViewBag.Nombre = "Juan";
    ViewData["Apellido"] = "Perez";
    return View();
}
```

```
<h2>Ejemplo</h2>

@ViewBag.Nombre
@ViewData["Apellido"]
```

# Vistas

```
public ActionResult Index()
{
    TempData["Message"] = "Ejemplo saludo";

    return RedirectToAction("Flash");
}

public ActionResult Flash()
{
    return View();
}
```

```
<h2>Ejemplo</h2>

@{
    string message = TempData["Message"] as string;
    if (message != null)
    {
        <div>@message</div>
    }
}
```

# View Engines

- ASPX
- Razor, CSHTML
- NHaml
- Spark
- Brail
- nVelocity

Strongly typed vs non Strongly typed

# Razor

```
<ul id="products">

    <% foreach(var p in products) { %>
        <li><%=p.Name%> ($<%=p.Price%>)</li>
    <% } %>

</ul>
```

```
<ul id="products">

    @foreach(var p in products) {
        <li>@p.Name ($@p.Price)</li>
    }

</ul>
```

# Reglas Razor

- Los bloques de código Razor son encerrados entre @{ ... }.
- Las expresiones en línea (variables y funciones) comienzan con @.
- Las sentencias de código terminan con punto y coma (;).
- Las variables son declaradas con la palabra clave “var”.
- Las cadenas de caracteres (strings) son encerradas entre comillas.
- El código es case sensitive.

# Razor

```
@{ var nombre = "Pedro"; }

<html>
 <body>
  <h2>Hola @nombre, son las @DateTime.Now</h2>
  <ul>
  @for (int i = 0; i < 10; i++) {
    <li>@i</li>
  }
 </ul>
 </body>
</html>
```

# Razor

```
@{
    int number = 1;
    string message = "Number is" " + number;
}

<p>Your Message: @message</p>
```

# Razor

```
<h1>Razor Example</h1>

<h3>
    Hello @name, the year is @DateTime.Now.Year
</h3>

<p>
    Checkout <a href="/Products/Details/@productId">this product</a>
</p>
```

# Razor

```
@if (DateTime.Now.Year == 2010) {
    <span>
        if year is 2010 then print this <br/>
        multi-line text block and
        the date: @DateTime.Now
    </span>
}
```

# Consideraciones en Razor

```
@{ int a = 10; int b = 3; }

<html>
 <body>
     El valor de 10 - 3 es: @a-b
 </body>
</html>
```

# Consideraciones en Razor

```
@{ int a = 10; int b = 3; }

<html>
 <body>
     El valor de 10 - 3 es: @(a-b)
 </body>
</html>
```

# Consideraciones en Razor

```
<html>
 <body>
     @for (int i = 0; i < 10; i++)
     {
        El valor de i es: @i <br />
     }
 </body>
</html>
```

# Consideraciones en Razor

```
<html>
 <body>
     @for (int i = 0; i < 10; i++)
     {
         <span>El valor de i es:</span> @i <br />
     }
 </body>
</html>
```

# Consideraciones en Razor

```
<html>
 <head>
      <style>
          @media screen
          {
            body { font-size: 13px;}
          }
      </style>
 </head>
 <body>
    Hola!!!!
 </body>
</html>
```

# Consideraciones en Razor

```
<html>
 <head>
      <style>
          @@media screen
          {
            body { font-size: 13px;}
          }
      </style>
 </head>
 <body>
    Hola!!!!
 </body>
</html>
```

# Helpers

# Helpers

- Son métodos que generar diferentes tipos de salidas que podemos utilizar en la vistas. Ej: links

```
@Html.ActionLink("TextoLink","MiAction")

@Html.ActionLink("TextoLink","MiAction","MiController")

@Html.ActionLink("TextoLink","MiAction","MiController", new {id =
1234, nombre = "Juan"}, new {@class = "linkbutton"})

@Html.RouteLink("MiRuta", new { producto = 1, categoria = 23 })
```

# Helpers: html form

- BeginForm()
- EndForm()
- TextArea()
- TextBox()
- CheckBox()
- RadioButton()
- ListBox()
- DropDownList()
- Hidden()
- Password()

# Helpers: html form

```
[HttpPost]
public ActionResult Nuevo(FormCollection formCollection)
{
// Codigo...
}
```

```
[HttpPost]
public ActionResult Nuevo(string nombre, string apellido)
{
// Codigo...
}
```

```
[HttpPost]
public ActionResult Nuevo(ClienteViewModel cliente)
{
// Codigo...
}
```

# Helpers: html form

```
public class TrabajadorViewModel
{
    public string Nombre { get; set; }
    public string Apellido { get; set; }
    public bool EsCasado { get; set; }
}
```

```
@using EjemploMVC.Models
@model TrabajadorViewModel

<form method="post">
    Nombre: @Html.TextBoxFor(x=>x.Nombre) <br />
    Apellido: @Html.TextBoxFor(x=>x.Apellido) <br />
    Es casado: @Html.CheckBoxFor(x=>x.EsCasado) <br />
    <input type="submit" value="Enviar"/>
</form>
```

# Helpers: html form “mejorado”

- HtmlDisplayFor: Renderiza el HTML necesario para visualizar una propiedad (sin poder ser editada)
- HtmlEditorFor: Renderiza el HTML necesario para poder editar una propiedad.

# Helpers: html form “mejorado”

```
public class TrabajadorViewModel
{
    public string Nombre { get; set; }
    public string Apellido { get; set; }
    public bool EsCasado { get; set; }
}
```

```
@using (Html.BeginForm()) {
    …
    @Html.LabelFor(x => x.Nombre)
    …
    @Html.EditorFor(x => x.Nombre)
    …
    <input type="submit" value="Enviar"/>
}
```

# Validación

```
public class ClienteViewModel
{
    [Required(ErrorMessage = "Falta el nombre)]
    [DisplayName("Nombre completo")]

    public string Nombre { get; set; }

    [Required(ErrorMessage = "Falta el email)]
    [DisplayName("Email")]
    [RegularExpression(@"\S+@\S+\.\S+")]
    public string Email { get; set; }
}
```

# Validación

```
[HttpPost]
public ActionResult Guardar(ClienteViewModel clienteViewModel)
{
    if (!ModelState.IsValid)
    {
        .....
        return View(clienteViewModel);
    }
    return View("ok");
}
```

# Validación

```
@using (Html.BeginForm()) {
      @Html.ValidationSummary(true)
     <fieldset>
           <legend>Cliente</legend>

           <div class="editor-label">
                 @Html.LabelFor(model => model.Nombre)
           </div>
          <div class="editor-field">
                @Html.EditorFor(model => model.Nombre)
          @Html.ValidationMessageFor(model => model.Nombre)
          </div>

           <div class="editor-label">
                 @Html.LabelFor(model => model.Email)
           </div>
          <div class="editor-field">
                @Html.EditorFor(model => model.Email)
                 @Html.ValidationMessageFor(model => model.Email)
           </div>

           <p>
                <input type="submit" value="Crear" />
          </p>
      </fieldset>
}
```

# Validación

```
[HttpPost]
public ActionResult Guardar(ClienteViewModel clienteViewModel)
{

            if (ModelState.IsValid)
            {
                  if (clienteService.ExisteCliente(……))
                  {
                        ModelState.AddModelError("", "Ya existe….");
                  }
            }

            if (!ModelState.IsValid)
            {
                  return View(clienteViewModel);
            }
            else
            {
                  // Codigo normal
                  return View("ok");
            }

}
```

# Atributos de validación propios

```
[AttributeUsage(AttributeTargets.Property)]
public class CuitValidationAttribute : ValidationAttribute
{
      public override bool IsValid(object value)
      {
         return .......
      }
}
```

```
public class ClienteViewModel
{
      [CuitValidator(ErrorMessage = "Error en el CUIT")]
      public string CuitActual { get; set; }
}
```

# Helpers propios

```
@helper SaludarHelper(string texto)
{
    <label>@texto</label>
}

<div>
@SaludarHelper("Hola")
</div>
```

```
@* \App_Code\MiHelper1.cshtml *@
@MiHelper1.SaludarHelper("Hola2")
```

# Helpers propios

```
namespace MvcApplicationXYZ.MisHelpers
{
    public static class MiHelper
    {
        public static MvcHtmlString MiLabel(this HtmlHelper html,
string texto)
        {
            var tagBuilder = new TagBuilder("label");
            tagBuilder.SetInnerText(texto);
            tagBuilder.Attributes.Add("data-texto", texto);
            return MvcHtmlString.Create(tagBuilder.ToString());
        }
    }
}
```

```
@using MvcApplication1.MisHelpers
....
@Html.MiLabel("Hola!!!")
```

# Partial Views

# Partial Views

- Una vista parcial permite definir una vista que se representará dentro de una vista primaria.

```
@Html.Partial("_TablaArticulos")

@{Html.RenderPartial("_TablaArticulos");}



@Html.Action("Listar", "Articulos")

@{Html.RenderAction("Listar", "Articulos");}
```

# Partial Views

```
@model IEnumerable<MvcApplicationEjemplo.Models.Articulo>
@if (Model != null)
{
    <table>
    @foreach (var item in Model)
    {
      <tr>
        <td>@item.Titulo</td>
      </tr>
    }
    </table>
}
```

```
<div>
   @Html.Partial("_MiVistaParcial", Model.Datos)
</div>
```

# Layouts

# Layouts

```
<!DOCTYPE html>
<html>
<head><title>Ejemplo Layout</title></head>
<body>
    <div>@RenderBody()</div>
    <footer>@RenderSection("Footer", false)</footer>
</body>
</html>
```

```
@{
    Layout = "~/Views/Shared/_MiLayout.cshtml";
}

<h1>Contenido general del Body!!!</h1>

@section Footer {
    Ejemplo footer.
}
```

# Layouts

```
<!DOCTYPE html>
<html>
<head><title>Ejemplo Layout</head>
<body>
    <div>@RenderBody()</div>
    <footer>
    @if (IsSectionDefined("Footer")) {
       RenderSection("Footer");
    }
    else
    {
       <span>Ejemplo default!</span>
    }
</footer>
</body>
</html>
```

# Layouts

```
public ActionResult Index()
{
 RegisterModel model = new RegisterModel();
 //…
 return View("Index", "_AdminLayout", model);
}
```

# Bundling and Minification

# Bundling and Minification

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo Layout</title>
    @Styles.Render("~/Content/themes/base/css", "~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
    @Scripts.Render("~/bundles/jquery")
    @RenderSection("scripts", required: false)
</head>
<body>
    <div>@RenderBody()</div>
    <footer>@RenderSection("Footer", false)</footer>
</body>
</html>
```

# Bundling and Minification

```
bundles.Add(new StyleBundle("~/Content/themes/base/css").Include(
            "~/Content/themes/base/jquery.ui.core.css",
            "~/Content/themes/base/jquery.ui.resizable.css",
            "~/Content/themes/base/jquery.ui.selectable.css",
            "~/Content/themes/base/jquery.ui.accordion.css",
            "~/Content/themes/base/jquery.ui.autocomplete.css",
            "~/Content/themes/base/jquery.ui.button.css",
            "~/Content/themes/base/jquery.ui.dialog.css",
            "~/Content/themes/base/jquery.ui.slider.css",
            "~/Content/themes/base/jquery.ui.tabs.css",
            "~/Content/themes/base/jquery.ui.datepicker.css",
            "~/Content/themes/base/jquery.ui.progressbar.css",
            "~/Content/themes/base/jquery.ui.theme.css"));
```

# Bundling and Minification

```
public static void RegisterBundles(BundleCollection bundles)
{
    bundles.Add(new ScriptBundle("~/bundles/jquery").Include(
                "~/Scripts/jquery-{version}.js"));

    //BundleTable.EnableOptimizations = true;
}
```

```
<system.web>
    <compilation debug="true" />
</system.web>
```

# Ajax

# ASP.NET MVC + AJAX

```
public ActionResult Listado(string categoria)
{
    ......
    return PartialView("_ListadoProductos", productos);
}
```

```
$.get("../Listado", { categoria: 'general'}, function(data) {
        .........
});
```

# ASP.NET MVC + AJAX

```
[HttpPost]
public ActionResult Listado(string categoria)
{
    ......
    return PartialView("_ListadoProductos", productos);
}
```

```
$.post("../Listado", { categoria: 'general'}, function(data) {
        .........
});
```

# ASP.NET MVC + AJAX

```
[HttpPost]
public JsonResult EjemploJsonPost(Persona persona)
{
    return Json(……);
}
```

```
$.ajax({
   type: "POST",
   url: "/Ejemplo/EjemploJsonPost",
   dataType: 'json',
   contentType: 'application/json; charset=utf-8',
   data: JSON.stringify({ nombre: 'juan', edad: 3 }),
   success: function (data) { console.log(data) },
   error: function (data) { console.log(data) }
});
```

# Routing

# Routing

- Permite mapear URL a controladores / acciones
- Friendly URLs
- /products.aspx?categoryid=8 vs /products/heladeras

# Routing

```
public class MvcApplication : System.Web.HttpApplication
{
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                "Default",
                "{controller}/{action}/{id}",
                new { controller = "Home",
                action = "Index", id = "" }
            );

        }

        protected void Application_Start()
        {
            RegisterRoutes(RouteTable.Routes);
        }
}
```

# Routing

- http://servidor.com/Autos/Ver/10
  - controller = Autos
  - action = Ver
  - id = 10
- http://servidor.com/Autos
  - controller = Autos
  - action = Index (valor por defecto)
  - id = No hay
- http://servidor.com
  - controller = Home (valor por defecto)
  - action = Index (valor por defecto)
  - id = No hay

# Routing

```
public class MvcApplication : System.Web.HttpApplication
{
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            routes.MapRoute("BuscadorAutos","buscadorautos/buscar",
              new { controller = "autos", action = "buscar" }
            );

            routes.MapRoute(
                "Default",
                "{controller}/{action}/{id}",
                new { controller = "Home",
                action = "Index", id = "" }
            );

        }

        protected void Application_Start()
        {
            RegisterRoutes(RouteTable.Routes);
        }
}
```

# Routing + QueryString

```
public class ProductosController : Controller
{
    public ActionResult ListdoDeProductos()
    {
        return View();
    }

    /* /Productos/VerProducto o …/VerProducto?idProducto  */
    public ActionResult VerProducto(int? idProducto)
    {
        return View();
    }
}
```

# Routing + QueryString

```
@Url.Action("Ver", "Home", new { id = 20 })
@Url.Action("Ver", "Home", new { id = 20, otro_id=30})


Url 1: /Home/Ver/20
Url 2: /Home/Ver/20?otro_id=30


public ActionResult Ver(string id, string otro_id)
{
    return View();
}
```

# Routing: Ejemplos

```
routes.MapRoute(
    "BuscadorAutos",
    "autos/buscar",
    new { controller = "autos", action = "buscar" }
);
```

*Url ejemplo*
*/autos/buscar?marca=toyota&color=rojo&model=corola*

```
routes.MapRoute(
    "BuscadorAutos1",
     "autos/buscar/marca/{marca}/color/{color}/modelo/{model}",
     new {controller = "autos", action = "buscar"}
);
```

*Url ejemplo*
*/autos/buscar/marca/toyota/color/rojo/model/corola*

# Routing: Ejemplos

```
routes.MapRoute(
        name: "Buscar",
        url: "Buscar/{tipoProducto}/{nombreProducto}",
        defaults: new { controller = "Productos", action =
"Buscar", tipoProducto = "Todos"});

-----------------------------------------------------------

public class ProductosController…… {

public ActionResult Buscar(string tipoProducto, string
nombreProducto = "Todos")
{
    string tipoProductoBuscar = Server.HtmlEncode(tipoProducto);
    string nombreProductoBuscar =
Server.HtmlEncode(nombreProducto);
    ...
}
```

# Routing Constraints

```
routes.MapRoute("blog", "{year}/{month}/{day}",
    new { controller = "blog", action = "index" },
    new { year = @"\d{4}", month = @"\d{2}", day = @"\d{2}" });
```

# Ignorar patrones de URL

```
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
```

http://www.servidor.com/xxx.axd/yyy

```
routes.IgnoreRoute("miruta/{*pathInfo}");
```

# Routing MVC 5

```
public class RouteConfig
{
    public static void RegisterRoutes(RouteCollection routes)
    {
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

        routes.MapMvcAttributeRoutes();

        routes.MapRoute(
            name: "Default",
            url: "{controller}/{action}/{id}",
            defaults: new { controller = "Home",
                action = "Index",
                id = UrlParameter.Optional
        );
    }
}
```



```
[Route("productos/{productId:int}/{productTitle}")]
public ActionResult Detail(int productId)
{
    // Search product details and assign the model.
    return View();
}
```

# Routing MVC 5

```
[Route("productos/{value?}")]
public ActionResult Info(string value)
{
    if (string.IsNullOrEmpty(value))
    {
        ViewBag.Description = "Sin valor";
    }
    else
    {
        ViewBag.Description = value;
    }

    return View();
}
```

```
[RoutePrefix("productos")]
public class ProductsController : Controller
{
    [Route]
    public ActionResult Index()
    {
        return View();
    }

    [Route("{value=Valor por defualt}")]
    public ActionResult Info(string value)
    {
        ViewBag.Description = value;
        return View();
    }

    [Route("{productId:int}/{productTitle}")]
    public ActionResult Detail(int productId)
    {
        // Search product details and assign the model.
        return View();
    }
}
```

# Routing MVC 5

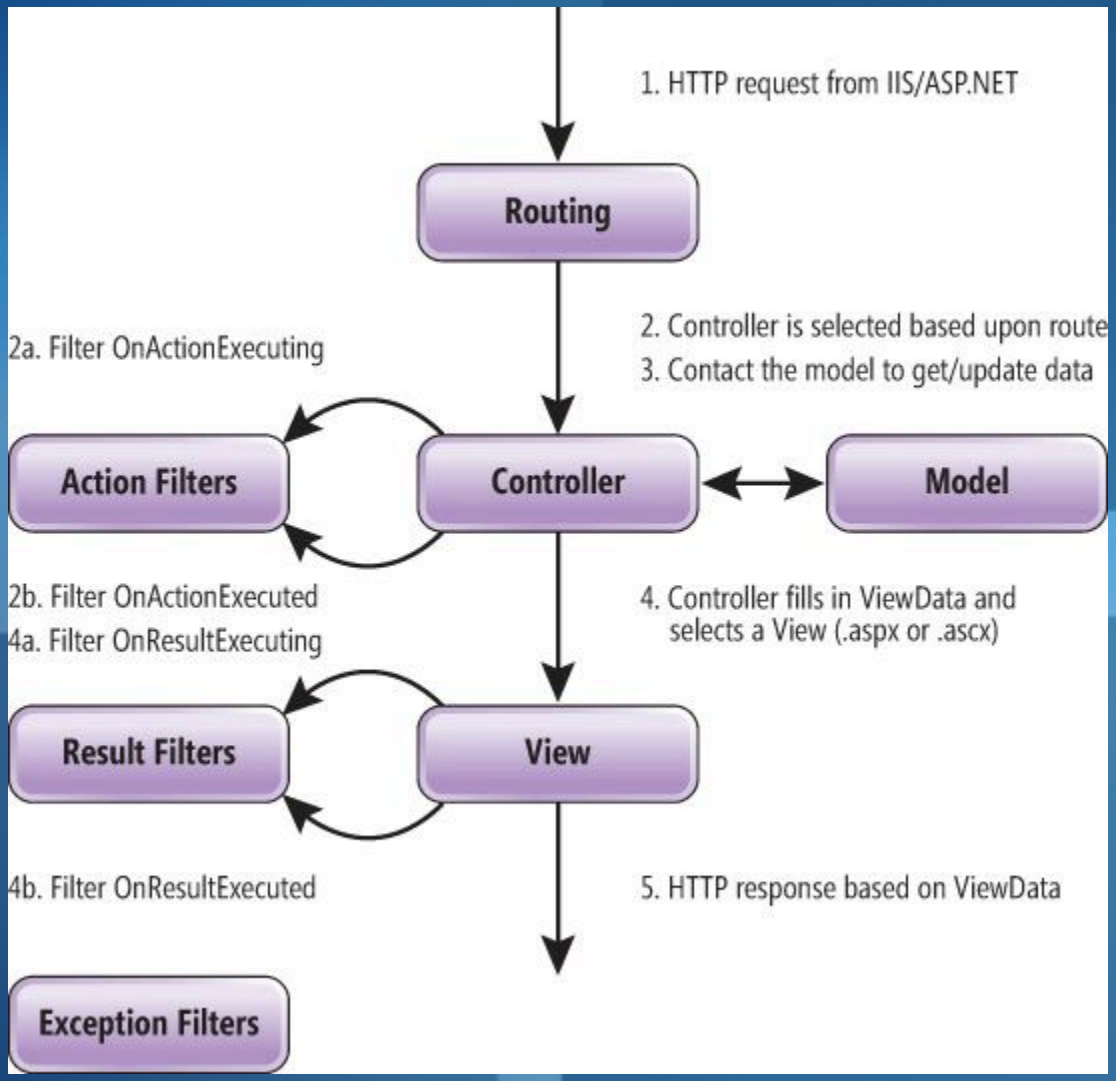
| Constraint | Description | Example |
|---|---|---|
| alpha | Matches uppercase or lowercase Latin alphabet characters (a-z, A-Z) | {x:alpha} |
| bool | Matches a Boolean value. | {x:bool} |
| datetime | Matches a **DateTime** value. | {x:datetime} |
| decimal | Matches a decimal value. | {x:decimal} |
| double | Matches a 64-bit floating-point value. | {x:double} |
| float | Matches a 32-bit floating-point value. | {x:float} |
| guid | Matches a GUID value. | {x:guid} |
| int | Matches a 32-bit integer value. | {x:int} |
| length | Matches a string with the specified length or within a specified range of lengths. | {x:length(6)}<br>{x:length(1,20)} |
| long | Matches a 64-bit integer value. | {x:long} |
| max | Matches an integer with a maximum value. | {x:max(10)} |
| maxlength | Matches a string with a maximum length. | {x:maxlength(10)} |
| min | Matches an integer with a minimum value. | {x:min(10)} |
| minlength | Matches a string with a minimum length. | {x:minlength(10)} |
| range | Matches an integer within a range of values. | {x:range(10,50)} |
| regex | Matches a regular expression. | {x:regex(^\d{3}-\d{3}-\d{4}$)} |

# Filtering

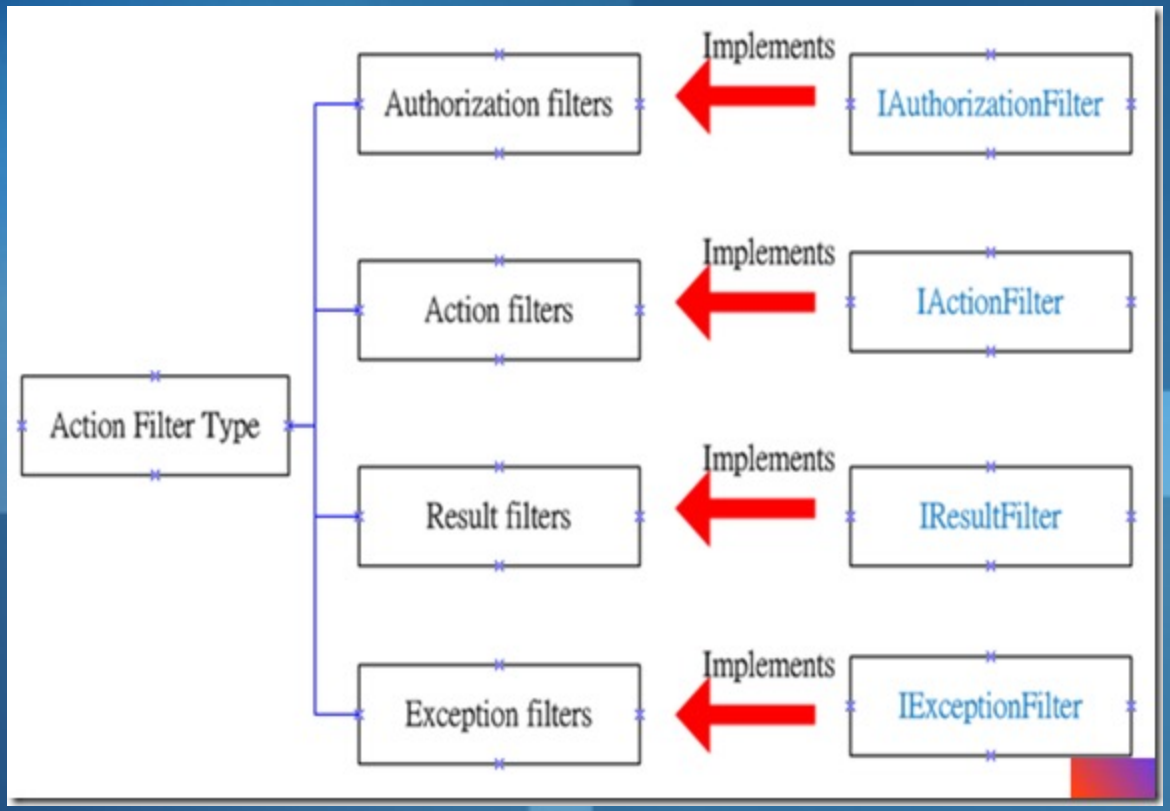
# Filtering

A veces se desea ejecutar la lógica antes de llamar a un método de acción o después de ejecutar un método de acción. Para ello, ASP.NET MVC proporciona filtros. Los filtros son clases personalizadas que proporcionan un método declarativo y de programación para agregar el comportamiento previo y posterior a la acción a los métodos de acción del controlador.

# Filtering


1. HTTP request from IIS/ASP.NET
Routing
2. Controller is selected based upon route
3. Contact the model to get/update data
2a. Filter OnActionExecuting
Action Filters
Controller
Model
2b. Filter OnActionExecuted
4a. Filter OnResultExecuting
4. Controller fills in ViewData and selects a View (.aspx or .ascx)
Result Filters
View
4b. Filter OnResultExecuted
5. HTTP response based on ViewData
Exception Filters

# Filtering


Action Filter Type
Authorization filters
Action filters
Result filters
Exception filters
Implements
IAuthorizationFilter
Implements
IActionFilter
Implements
IResultFilter
Implements
IExceptionFilter

# Filtering: ejemplos

| Nombre | Descripción |
| --- | --- |
| Authorize | Restringe una acción para los roles o usuarios especificados. |
| HandleError | Permite especificar una vista que se mostrará en caso de excepción no controlada. |
| OutputCache | Almacena en caché la salida de un controlador |
| ValidateAntiForgeryToken | Ayuda a evitar peticiones de falsificación de petición (pishing) |

# Filtering: ejemplos

```
[OutputCache(Duration = 20)]
public ActionResult EjemploCache ()
{
    string respuesta = string.Format("Hora cachea!!!! {0}",
DateTime.Now);
    return Content(respuesta);
}
```

```
[Authorize]
public ActionResult EjemploSeguro()
{
  …
}
```

# Filtering: ejemplos

```
[AllowAnonymous]
public ActionResult LoginUsuario(string usuario, string clave)
{
  …
  FormsAuthentication.SetAuthCookie(username, false);
}
```

```
@using (Html.BeginForm()){
@Html.AntiForgeryToken()
  …
}
```

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult ActualizarDatos(…)
{
  …
}
```

# Filtros globales

```
public class FilterConfig
{
    public static void RegisterGlobalFilters(GlobalFilterCollection
filters)
    {
        filters.Add(new AuthorizeAttribute);
        filters.Add(new HandleErrorAttribute());
    }
}
```

# Filtros propios

```
public class UnhandledExceptionFilter : ExceptionFilterAttribute {
        private static readonly ILog log = LogManager.GetLogger(typeof(…));

        public override void OnException(HttpActionExecutedContext context)
        {
            string url = "";
            if (context.Request.RequestUri != null)
            {
                url = context.Request.RequestUri.ToString();
            }

            log.Error("Error general url = " + url, context.Exception);

        }
}
```

```
[UnhandledExceptionFilter]
public ActionResult EjemploError ()
{
    ……
}
```

# Filtros propios

```
public class MiActionFilterAttribute : FilterAttribute, IActionFilter
{
    public void OnActionExecuting(ActionExecutingContext filterContext)
    {
        …
    }

    public void OnActionExecuted(ActionExecutedContext filterContext)
    {
        …
    }
}
```

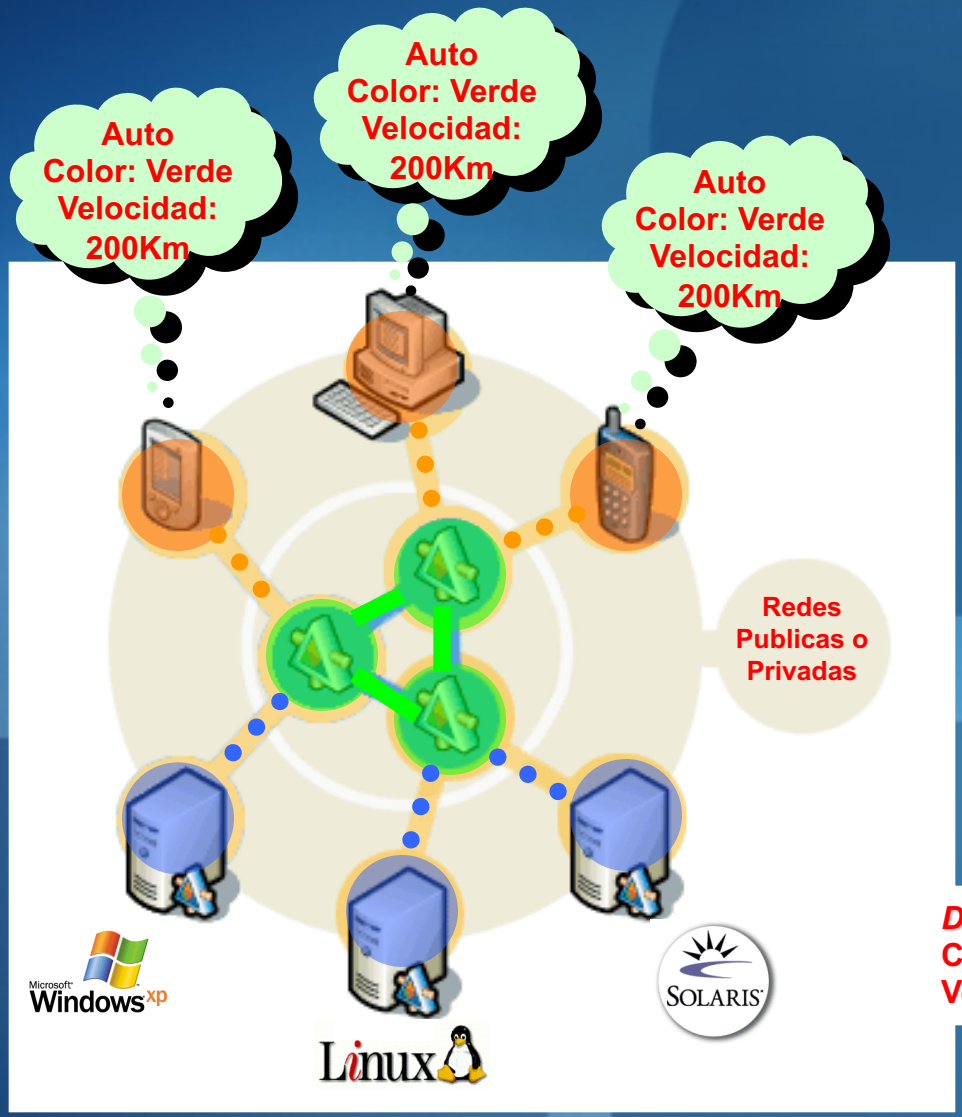
```
[MiActionFilter]
public ActionResult Index() {...}
```

# Web API

# ASP.NET Web API

ASP.NET Web API es un marco que facilita la creación de servicios HTTP disponibles para una amplia variedad de clientes, entre los que se incluyen exploradores y dispositivos móviles. ASP.NET Web API es la plataforma perfecta para crear aplicaciones RESTful en .NET Framework.

# http://www.ventadeautos.com.ar/autos/1




***Datos***: Auto
Color: Verde
Velocidad: 200Km

# Características de REST

- poco acoplados
- encapsulados
- simples
- rápidos
- reutilizables
- sin información de estado (“stateless”)
- autónomos e independientes de una plataforma o lenguaje
- localizables
- solución orientada a la Web 2.0
- basados en estándares simples HTTP (GET – POST – PUT –DELETE)

# Ejemplo de un Api REST simple

http://www.xyz.com/autos (listado - GET)
http://www.xyz.com/autos/1 (consulta - GET)
http://www.xyz.com/autos/1/borrar (GET)
http://www.xyz.com/autos/1 (actualizar - POST)
http://www.xyz.com/autos/nuevo (POST)

# Ejemplo de un Api REST Full

http://www.xyz.com/autos (listado - GET)
http://www.xyz.com/autos/1 (consulta - GET)
http://www.xyz.com/autos/1 (DELETE)
http://www.xyz.com/autos/1 (PUT)
http://www.xyz.com/autos (POST)

# Web API

```
public static class WebApiConfig
{
   public static void Register(HttpConfiguration config)
   {
     config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/v1/{controller}/{action}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );

     config.Filters.Add(new AuthorizeAttribute());

     var configuration = GlobalConfiguration.Configuration;

     configuration.Formatters.Remove(configuration.Formatters.XmlFormatter);
     var formatters = configuration.Formatters;
     var jsonFormatter = formatters.JsonFormatter;
     var settings = jsonFormatter.SerializerSettings;
     settings.Formatting = Newtonsoft.Json.Formatting.Indented;
     settings.ContractResolver = new CamelCasePropertyNamesContractResolver();
   }
}
```

# Web API

```
[HttpPost]
public HttpResponseMessage BuscarContactos([FromBody] JObject datos)
{

    ……
    var respuesta  = new { …… };
    return Request.CreateResponse(HttpStatusCode.OK, respuesta);
}
```

# Mobile