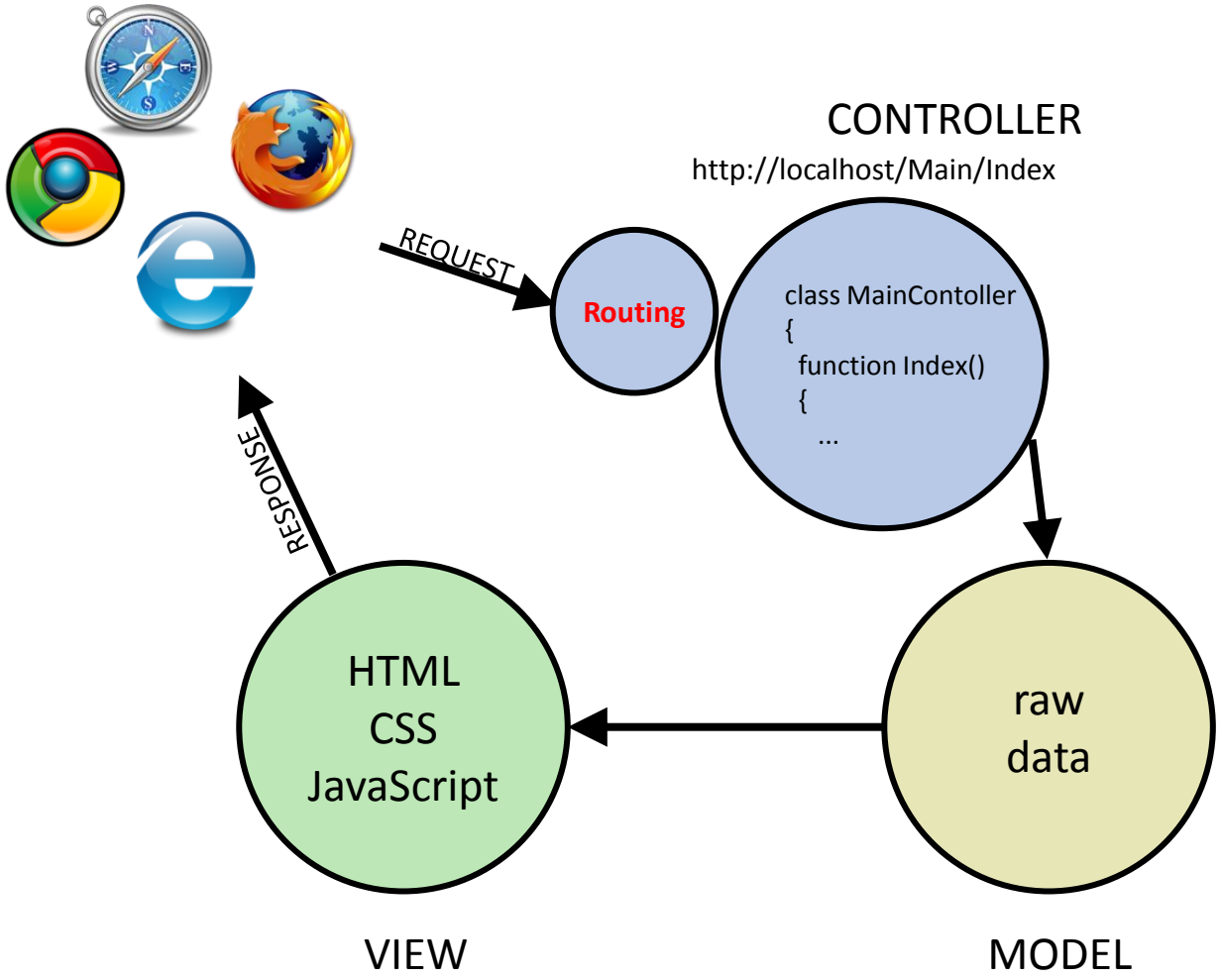
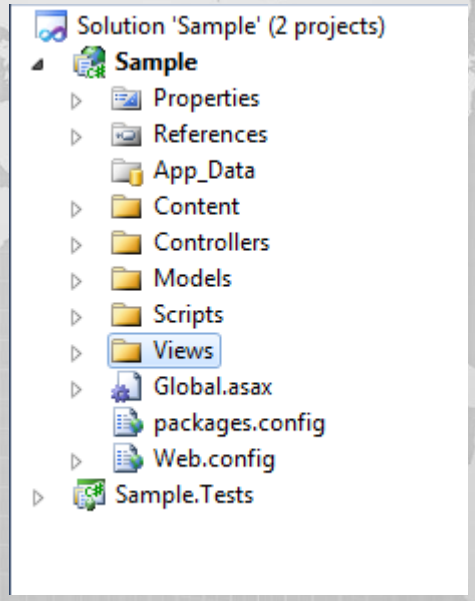
ASP.NET MVC

MVC


CONTROLLER
http://localhost/Main/Index
REQUEST
Routing
class MainContoller
{
function Index()
{
...
RESPONSE
HTML
CSS
JavaScript
raw
data
VIEW
MODEL

Project structure

- Scripts. Content – accessible directly by IIS
- Controllers – place to locate all controllers
  - Naming convention {Name}Controller
- Views – place to locate files which will generate HTML output
  - Naming convention {ControllerName}\{ControllerActionName}
- App_Data – shortcut to IIS virtual directory with inaccessible data to user (Databses)
- Models – Ignored by IIS
- Global.asax – application starting point. MVC initialization goes here.



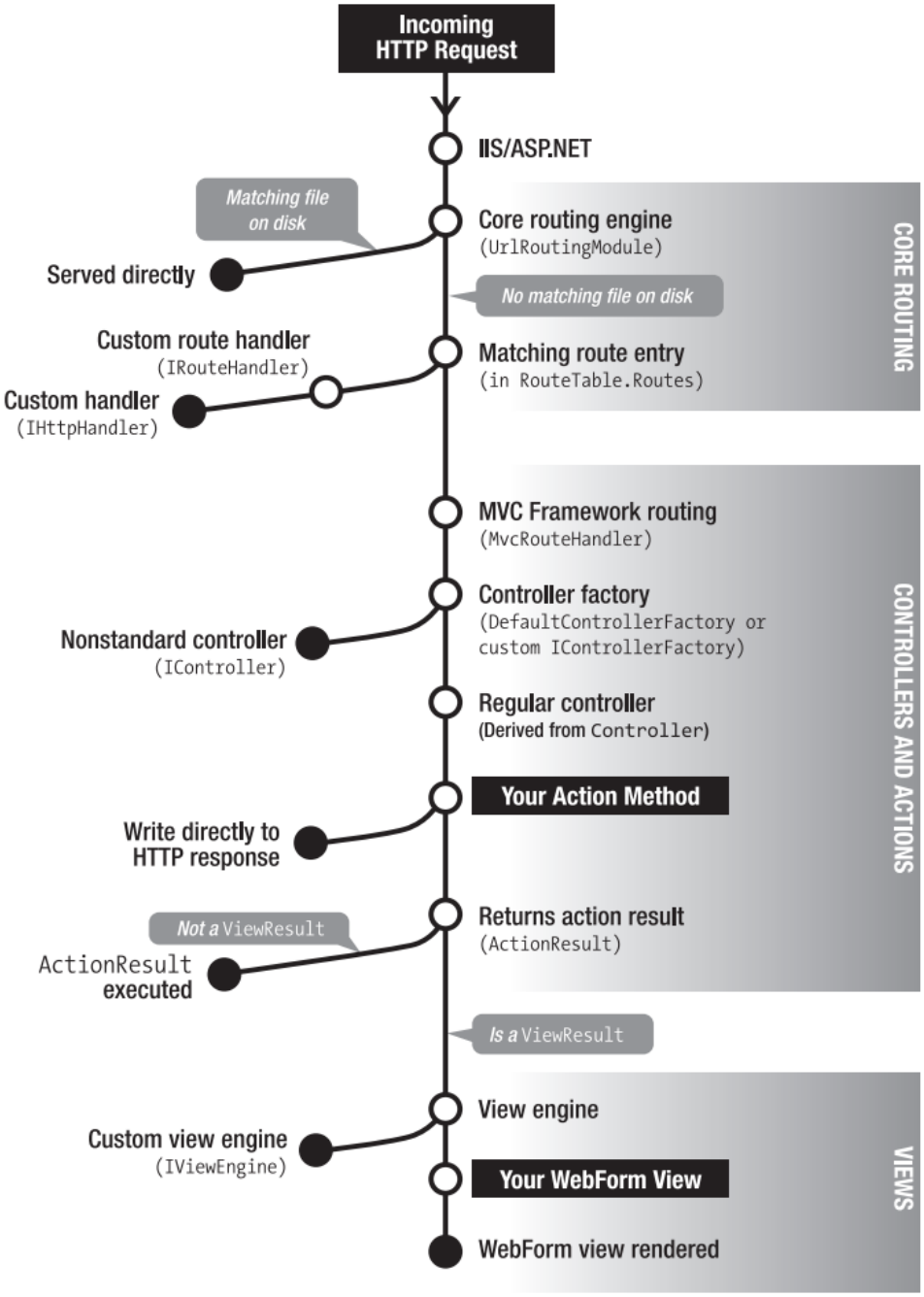
Routing

- Before

Incoming URL	Might Correspond To
http://mysite.com/default.aspx	e:\webroot\default.aspx
http://mysite.com/admin/login.aspx	e:\webroot\admin\login.aspx
http://mysite.com/articles/AnnualReview	File not found! Send error 404.

- After

Incoming URL	Might Correspond To
http://mysite.com/photos	{ controller = "Gallery", action = "Display" }
http://mysite.com/admin/login	{ controller = "Auth", action = "Login" }
http://mysite.com/articles/AnnualReview	{ controller = "Articles", action = "View", contentItemName = "AnnualReview" }


Incoming HTTP Request
IIS/ASP.NET
Matching file on disk
Core routing engine (UrlRoutingModule)
Served directly
No matching file on disk
Custom route handler (IRouteHandler)
Matching route entry (in RouteTable.Routes)
Custom handler (IHttpHandler)
CORE ROUTING
MVC Framework routing (MvcRouteHandler)
Controller factory (DefaultControllerFactory or custom IControllerFactory)
Nonstandard controller (IController)
Regular controller (Derived from Controller)
Your Action Method
Write directly to HTTP response
Not a ViewResult
Returns action result (ActionResult)
ActionResult executed
CONTROLLERS AND ACTIONS
Is a ViewResult
View engine
Custom view engine (IViewEngine)
Your WebForm View
WebForm view rendered
VIEWS

Routing

binary studio

Typical routing scenario:

```
routes.MapRoute(
    "Default", // Route name
    "{controller}/{action}/{id}", // URL with parameters
    new { controller = "Home", action = "Index", id = UrlParameter.Optional } // Parameter defaults
);
```

Routing with Constraint:

```
routes.MapRoute(
    null, // Route name
    "Sync/{tag}", // URL with parameters
    new { controller = "MySync", action = "GetPhotoByTag", tag = "hat" },// Parameter defaults
    new { tag=@"\d+"} //Constrainst only to numbers
);

routes.MapRoute(
    null, // Route name
    "Secret/{category}/{subcategory}/{id}", // URL with parameters
    new { controller = "Home", action = "Dino",
        category = string.Empty, subcategory = string.Empty, id = (int?)null } // Parameter defaults
        ,new {id = new YearRouteConstraint()}
);
```

Controllers

- IController
  - All controllers implement this interface
- Controller
  - Basic realization of Icontroller
  - Invokes method using naming convention from routing table
  - Process parameters and bind data
  - Built by MVC controller factory

- IAsyncController
- AsyncController
  - Asynchronous realization of controller
  - Method pairs of {MethodName}Async + {MethodName}Complited
  - Dramatic increase of scalability if action has a lot of IO operations

Result

- ActionResult
- ViewResult
- RedirectResult
- FileResult
- ContentResult
- JsonResult
- CustomResult

```
namespace System.Web.Mvc
{
  public abstract class ActionResult
  {
    public abstract void ExecuteResult(ControllerContext context);
  }
}
```

```
TextWriter output = context.HttpContext.Response.Output;
this.View.Render(new ViewContext(context, this.View, this.ViewData, this.TempData, output), output);
if (viewEngineResult == null)
  return;
viewEngineResult.ViewEngine.ReleaseView(context, this.View);
```

Controllers Extensibility

binary studio

- Design of the whole framework is based on idea of extensibility
- IControllerFactory
- Exposing Controller factory interface to the framework user allows modification of the core functionality of each controller
- Inversion of Control, Dependency Injection
- Other extensibility points: action filters, model binders, action results, validations, view engines, and even more

Unit testing

binary studio

- Highly testable due to exposed API, almost everything can be mocked
- What is mocking?
- Example

Result

Clean

Robust

Extensible

Scalable

Testable