عنوان: طراحی گردش کاری با استفاده از State machines - قسمت اول
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۰ ۲۱:۱۰
آدرس: www.dotnettips.info
گروه‌ها: Workflow, State machine

State machine چیست؟

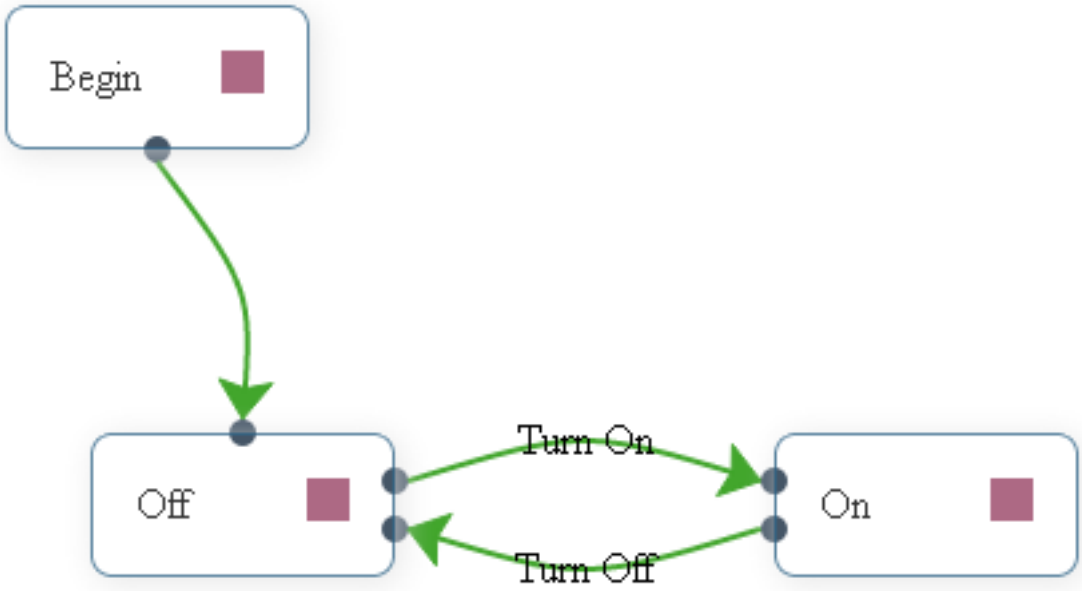
State machine مدلی است بیانگر نحوه واکنش سیستم به وقایع مختلف. یک ماشین حالت وضعیت جاری قسمتی از سیستم را نگهداری کرده و به ورودی‌های مختلف پاسخ می‌دهد. این ورودی‌ها در نهایت وضعیت سیستم را تغییر خواهند داد.
نحوه پاسخگویی یک ماشین حالت (State machine) را به رویدادی خاص، انتقال (Transition) می‌نامند. در یک انتقال مشخص می‌شود که ماشین حالت بر اساس وضعیت جاری خود، با دریافت یک رویداد، چه عکس العملی را باید بروز دهد. عموما (و نه همیشه) در حین پاسخگویی ماشین حالت به رویدادهای رسیده، وضعیت آن نیز تغییر خواهد کرد. در اینجا گاهی از اوقات پیش از انجام عملیاتی، نیاز است شرطی بررسی شده و سپس انتقالی رخ دهد. به این شرط، guard گفته می‌شود.
بنابراین به صورت خلاصه، یک ماشین حالت، مدلی است از رفتاری خاص، تشکیل شده از حالات، رویدادها، انتقالات، اعمال (actions) و شرطها (Guards). در اینجا:
- یک حالت (State)، شرطی منحصربفرد در طول عمر ماشین حالت است. در هر زمان مشخصی، ماشین حالت در یکی از حالات از پیش تعریف شده خود قرار خواهد داشت.
- یک رویداد (Event)، اتفاقی است که به ماشین حالت اعمال می‌شود؛ یا همان ورودی‌های سیستم.
- یک انتقال (Transition)، بیانگر نحوه رفتار ماشین حالت جهت پاسخگویی به رویداد وارده بر اساس وضعیت جاری خود می‌باشد. در طی یک انتقال، سیستم از یک حالت به حالتی دیگر منتقل خواهد شد.
- برای انجام یک انتقال، نیاز است یک شرط (Guard/Conditional Logic) بررسی شده و در صورت true بودن آن، انتقال صورت گیرد.
- یک عمل (Action)، بیانگر نحوه پاسخگویی ماشین حالت در طول دوره انتقال است.

چگونه می‌توان الگوی ماشین حالت را تشخیص داد؟

اکثر برنامه‌های وب، متشکل از پیاده سازی چندین ماشین حالت می‌باشند؛ مانند ثبت نام در سایت، درخواست یک کتاب از کتابخانه، ارسال درخواست‌ها و پاسخگویی به آن‌ها و یا حتی ارسال یک مطلب در سایت، تائید و انتشار آن.
البته عموما در حین طراحی برنامه‌ها، کمتر به این نوع مسایل به شکل یک ماشین حالت نگاه می‌شود. به همین جهت بهتر است معیارهایی را برای شناخت زود هنگام آن‌ها مدنظر داشته باشیم:
- آیا در جدول بانک اطلاعاتی خود فیلدهایی مانند State (حالت) یا Status (وضعیت)دارید؟ اگر بله، به این معنا است که در حال کار با یک ماشین حالت هستید.
- عموما فیلدهای Bit و Boolean، بیانگر حضور ماشین‌های حالت هستند. مانند IsPaid ، IsPublished و یا حتی داشتن یک فیلد timeStamp که می‌تواند NULL بپذیرد نیز بیانگر استفاده از ماشین حالت است؛ مانند فیلدهای paid_at، published_at و یا confirmed_at.
- داشتن رکوردهایی که تنها در طول یک بازه زمانی خاص، معتبر هستند. برای مثال آبونه شدن در یک سایت در طول یک بازه زمانی مشخص.
- اعمال چند مرحله‌ای؛ مانند ثبت نام در سایت و دریافت ایمیل فعال سازی. سپس فعال سازی اکانت از طریق ایمیل.

مثالی ساده از یک ماشین حالت

یک کلید برق را در نظر بگیرید. این کلید دارای دو حالت (states) روشن و خاموش است. زمانی که خاموش است، با دریافت رخدادی (event)، به وضعیت (state/status) روشن، منتقل خواهد شد (Transition) و برعکس.



در اینجا حالات با مستطیل‌های گوشه گرد نمایش داده شده‌اند. انتقالات توسط فلش‌هایی انحناء دار که حالات را به یکدیگر متصل می‌کنند، مشخص گردیده‌اند. برچسب‌های هر فلش، مشخص کننده نام رویدادی است که سبب انتقال و تغییر حالت می‌گردد. با شروع یک ماشین حالت، این ماشین در یکی از وضعیت‌های از پیش تعیین شده‌اش قرار خواهد گرفت (initial state)؛ که در اینجا حالت خاموش است.
این نوع نمودارها می‌توانند شامل جزئیات بیشتری نیز باشند؛ مانند برچسب‌هایی که نمایانگر اعمال قابل انجام در طی یک انتقال هستند.

رسم ماشین‌های حالت در برنامه‌های وب، به کمک کتابخانه jsPlumb

کتابخانه‌های زیادی برای رسم فلوچارت، گردش‌های کاری، ماشین‌های حالت و امثال آن جهت برنامه‌های وب وجود دارند و یکی از معروف‌ترین‌های آن‌ها کتابخانه jsPlumb است. این کتابخانه به صورت یک افزونه jQuery طراحی شده است؛ اما به عنوان افزونه‌ای برای کتابخانه‌های MooTools و یا YUI3/Yahoo User Interface 3 نیز قابل استفاده می‌باشد. کتابخانه jsPlumb در مرورگرهای جدید از امکانات ترسیم SVG و یا HTML5 Canvas استفاده می‌کند. برای سازگاری با مرورگرهای قدیمی‌تر مانند IE8 به صورت خودکار به VML سوئیچ خواهد کرد. همچنین این کتابخانه امکانات ترسیم تعاملی قطعات به هم متصل شونده را نیز دارا است (شبیه به طراح یک گردش کاری). البته برای اضافه شدن امکاناتی مانند کشیدن و رها کردن در آن نیاز به jQuery-UI نیز خواهد داشت.
برای نمونه اگر بخواهیم مثال فوق را توسط jsPlumb ترسیم کنیم، روش کار به صورت زیر خواهد بود:

```
<!doctype html>
<html>
<head>
    <title>State Machine Demonstration</title>
    <style type="text/css">
        #opened
        {
            left: 10em;
            top: 5em;
        }

        #off
        {
            left: 12em;
            top: 15em;
        }
```

```
        #on
        {
            left: 28em;
            top: 15em;
        }

        .w
        {
            width: 5em;
            padding: 1em;
            position: absolute;
            border: 1px solid black;
            z-index: 4;
            border-radius: 1em;
            border: 1px solid #346789;
            box-shadow: 2px 2px 19px #e0e0e0;
            -o-box-shadow: 2px 2px 19px #e0e0e0;
            -webkit-box-shadow: 2px 2px 19px #e0e0e0;
            -moz-box-shadow: 2px 2px 19px #e0e0e0;
            -moz-border-radius: 0.5em;
            border-radius: 0.5em;
            opacity: 0.8;
            filter: alpha(opacity=80);
            cursor: move;
        }

        .ep
        {
            float: right;
            width: 1em;
            height: 1em;
            background-color: #994466;
            cursor: pointer;
        }

        .labelClass
        {
            font-size: 20pt;
        }
    </style>
    <script type="text/javascript" src="jquery.min.js"></script>
    <script type="text/javascript" src="jquery-ui.min.js"></script>
    <script type="text/javascript" src="jquery.jsPlumb-all-min.js"></script>
    <script type="text/javascript">
        $(document).ready(function () {

            jsPlumb.importDefaults({
                Endpoint: ["Dot", { radius: 5}],
                HoverPaintStyle: { strokeStyle: "blue", lineWidth: 2 },
                ConnectionOverlays: [
["Arrow", { location: 1, id: "arrow", length: 14, foldback: 0.8}]
]
            });

            jsPlumb.makeTarget($(".w"), {
                dropOptions: { hoverClass: "dragHover" },
                anchor: "Continuous"
            });

            $(".ep").each(function (i, e) {
                var p = $(e).parent();
                jsPlumb.makeSource($(e), {
                    parent: p,
                    anchor: "Continuous",
                    connector: ["StateMachine", { curviness: 20}],
                    connectorStyle: { strokeStyle: '#42a62c', lineWidth: 2 },
                    maxConnections: 2,
                    onMaxConnections: function (info, e) {
                        alert("Maximum connections (" + info.maxConnections + ") reached");
                    }
                });
            });

            jsPlumb.bind("connection", function (info) {
            });

            jsPlumb.draggable($(".w"));

            jsPlumb.connect({ source: "opened", target: "off" });
            jsPlumb.connect({ source: "off", target: "on", label: "Turn On" });
            jsPlumb.connect({ source: "on", target: "off", label: "Turn Off" });
```

```
        });
    </script>
</head>
<body>
    <div class="w" id="opened">
        Begin
        <div class="ep">
        </div>
    </div>
    <div class="w" id="off">
        Off
        <div class="ep">
        </div>
    </div>
    <div class="w" id="on">
        On
        <div class="ep">
        </div>
    </div>
</body>
</html>
```

مستندات کامل jsPlumb را در سایت آن می‌توان ملاحظه نمود.
در مثال فوق، ابتدا css و فایل‌های js مورد نیاز ذکر شده‌اند. توسط css، مکان قرارگیری اولیه المان‌های متناظر با حالات، مشخص می‌شوند.
سپس زمانیکه اشیاء صفحه در دسترس هستند، تنظیمات jsPlumb انجام خواهد شد. برای مثال در اینجا نوع نمایشی Endpointها به نقطه تنظیم شده است. موارد دیگری مانند مستطیل نیز قابل تنظیم است. سپس نیاز است منبع و مقصدها به کتابخانه jsPlumb معرفی شوند. به کمک متد jsPlumb.makeTarget، تمام المان‌های دارای کلاس w به عنوان منبع و با شمارش divهایی با class=ep، مقصدهای قابل اتصال تعیین شده‌اند (jsPlumb.makeSource). متد jsPlumb.bind یک callback function است و هربار که اتصالی برقرار می‌شود، فراخوانی خواهد شد. متد jsPlumb.draggable تمام عناصر دارای کلاس w را قابل کشیدن و رها کردن می‌کند و در آخر توسط متدهای jsPlumb.connect، مقصد و منبع‌های مشخصی را هم متصل خواهیم کرد. نمونه نهایی تهیه شده برای بررسی بیشتر.

برای مطالعه بیشتر

Finite-state machine

UML state machine

UML 2 State Machine Diagrams

مثال‌هایی در این مورد

نظرات خوانندگان

نویسنده: omid
تاریخ: ۱۳۹۱/۱۰/۱۱ ۷:۷

سلام
با تشکر از مطلب بسیار جالبی که بیان کردید . می‌خواستم بدونم آیا این امکان وجود داره که سمت سرور بعد از طراحی ، گردش کار رو ذخیره کنیم ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۱ ۱۱:۷

jsPlumb یک سری callback function داره که زمان اتصال نودها و یا زمان قطع اتصالات فراخوانی خواهند شد:

```
jsPlumb.bind("jsPlumbConnection", function(connectionInfo) {
  // update your data model here.
});

jsPlumb.bind("jsPlumbConnectionDetached", function(connectionInfo) {
  // update your data model here.
});
```

در اینجا شما فرصت خواهید داشت اطلاعات مدل مورد نظر را به روز کنید.
connectionInfo دریافتی یک شیء جاوا اسکریپتی است شامل connection, source, sourceEndpoint, sourceId, target, targetEndpoint, targetId

نویسنده: علیرضا
تاریخ: ۱۳۹۱/۱۰/۱۱ ۱۲:۲۳

سلام،
آیا چنین افزونه یا تکنیکی برای پیاده سازی SM داخل برنامه‌های تحت ویندوز (.net) وجود داره؟
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۱ ۱۲:۳۵

در قسمت بعد این مساله دنبال خواهد شد.

نویسنده: امیر بختیاری
تاریخ: ۱۳۹۱/۱۰/۱۱ ۱۳:۱۵

با سلام
با تشکر از مطلب خوبی که بیان کردید.
امکان تعریف WorkFlowهای پیچیده و با شرایط و تنظیمات پیچیده نیز وجود دارد؟
به غیر از این کامپوننت ، کامپوننت‌های دیگری هم وجود داره ؟ اگر ممکنه لیستی از این کامپوننت‌ها را قرار بدهید.
همچنین میشه بعد از رسم کامل workFlow ذخیره سازی را انجام بدیم و بعد دوباره به همان شکل لود کنیم ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۱ ۱۴:۱۶

- بله. در قسمت بعد این مساله با معرفی یک کتابخانه مدیریت ماشین‌های حالت، دنبال خواهد شد.

- موارد دیگری مانند WireIt ، (^) draw.io ، raphaeljs و jGraph هم برای رسم گراف هستند.
- بله. باید کمی به jQuery Ajax آشنا باشید. می‌تونید اشیایی رو که قرار هست در صفحه ترسیم بشن به صورت آرایه‌ای از اشیاء جاوا اسکریپتی تعریف کنید. هر شیء دارای source و target است به علاوه مختصات x و y. نهایتا برای ارسال آن به سرور از طریق jQuery Ajax خواهید داشت:

```
JSON.stringify(whole_object)
```

برای دریافت لیست اشیاء هم به صورت JSON از سرور و رسم آن در سمت کلاینت با JSON.decode می‌تونید شروع کنید.

نویسنده: امیران
تاریخ: ۱۳۹۳/۱۰/۰۳ ۱۳:۴۸

برای به گردش درآوردن آبجکت در Flow ایجاد شده تحت وب ایده ای می‌توانید بدهید؟
قاعدتا چندین راه (مانند موارد دیگر) برای آن وجود دارد اما راه بهینه بدون استفاده از WF و مستقل از شیء چیست؟
با جستجوی در وب چیزی دستگیرم نشد. اولین راه حلی که به نظرم می‌رسد در سمت application با mvc کنترلی با تحریف OnActionExecuting و OnResultExecuted داشته باشیم برای پردازش Stateهای مختلف و در متدهای ذکر شده ذخیره در بانک اطلاعاتی را انجام دهیم و در سمت دیتابیس هم جدولی داشته باشیم که با تغییر وضعیت، شناسه Workflow و شناسه State را برای نگهداشتن تاریخچه وضعیت‌ها و پیگیری آن را ذخیره کنیم. البته همچنان برای بررسی شروط هنگام transitionها به نتیجه ای نرسیده ام

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۰۳ ۱۴:۶

قسمت بعدی این سری را مطالعه کنید. نحوه‌ی طراحی یک گردش کاری توسط کتابخانه‌ی Stateless بیان شده. نمونه مثالی هم برای آن ذکر شده‌است. راه اندازی آن در وب یا دسکتاپ تفاوتی نمی‌کند. متد Save کلاس BlogPostManager که در ابتدای کار فراخوانی شود، قسمت بعدی گردش کاری فعال می‌شود. نفر بعدی بر اساس تصمیم خود (مثلا پس از دریافت ایمیل حاصل از save و دریافت آدرسی برای واکنش)، این گردش کاری را به حالتی دیگر ارتقاء خواهد داد.

نویسنده: امیران
تاریخ: ۱۳۹۳/۱۰/۰۸ ۱۵:۴۸

با توجه به اینکه اطلاعات مربوط به State ها، همینطور Transitionها و ... در بانک اطلاعاتی ذخیره می‌شود برای به گردش درآوردن باید با fetch کردن داده‌ها از بانک اطلاعات یک Object را به گردش درآوریم. حال برای ساختن و کانفیگ کلاس مربوط به State Machine به نظر می‌رسد یک راه این باشد:

```
private StateMachine<string, string> stateMachine;
        private StateMachineCOM source;
        private string startState;
        public delegate void UnhandledTriggerDelegate(State state,  StateConfig trigger);
        public delegate void EntryExitDelegate();
        public delegate bool GuardClauseDelegate();
        public string Id;
        public EntryExitDelegate OnEntry = null;
        public EntryExitDelegate OnExit = null;
        public GuardClauseDelegate GuardClauseFromToTrigger = null;
        public UnhandledTriggerDelegate OnUnhandledTrigger = null;

        public StateMachineRequest(StateMachineCOM source, string startStateId)
        {
            this.source = source;
            this.startState = startStateId;
        }

        public void Configure()
        {
            this.stateMachine = new StateMachine<string, string>(startState);
            var states = source.States;
            states.ForEach(state =>
            {
                var triggers = source.StateConfigs.AsQueryable()
```

```
                                    .Where(config => config.FromStateId == state.StateId)
                                    .Select(config => new {Id=config.TransitionId.ToString(), From=
config.FromStateId.ToString(), To= config.ToStateId.ToString(), Permit=config.PermiteAction })
                                    .ToList();

                triggers.ForEach(trig =>
                {
                    this.stateMachine.Configure(state.StateId.ToString())
                });
            });
        }

        public bool TryFireTrigger(string TrigerId)
        {
            if (!stateMachine.CanFire(TrigerId))
            {
                return false;
            }
            stateMachine.Fire(TrigerId);
            return true;
        }

        public string GetCurrentState()
        {
            return this.stateMachine.State;
        }
```

باشد یعنی Stateها Transitionها و ... را بعد از Fetch کردن از بانک اطلاعاتی به State Machine ارسال کنیم. حالا برای در نظر گرفتن شروط مربوط به OnEntry و OnExit یا GuardClauseFromToTrigger پیشنهاد شما توجه به اینکه براساس State می‌بایست این متدها ساخته شوند چیست؟

- آیا بهتر است delegate پارامتر دریافت کند؟

اگر بله پیاده سازی آن در هنگام کانفیگ به چه صورت است؟ به این صورت ؟

```
this.stateMachine.Configure(state.StateId.ToString())
                        .OnEntry(() => { if (state.OnEnter) OnEntry(trig.Id);})
                        .OnExit(() => { if (state.OnExit) OnExit(trig.Id); })
                        .PermitIf(trig.From, trig.To, () => { if (trig.Permit) return
GuardClauseFromToTrigger(); return true; });
```

- اگر خیر چگونه می‌توان این متدها را بصورت دینامیک ایجاد کرد و به هنگام کانفیگ ماشین حالت به آن انتساب داد و بعد در هنگام گردش آبجکت به آن دسترسی داشت؟

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۱۰/۰۸ ۱۶:۳۳

بهتر هست مطلب قسمت دوم رو در همون قسمت دوم پیگیری کنید. اگر قسمت دوم رو مطالعه کرده باشید، یک قسمت، طراحی کلی state machine هست (مثل کلاس BlogPostStateMachine). در قسمت بعدی اون طراحی manager که کار با دیتابیس در اونجا انجام میشه (مثل کلاس BlogPostManager). الان شما مطالب رو مخلوط کردید که نیازی نبوده. در کلاس BlogPostManager جاهایی که کامنت شده، قسمت‌هایی هست که از دیتابیس اطلاعات رو می‌خونه. اون قسمت‌ها رو خودتون پر کنید.

نویسنده: امیران
تاریخ: ۱۳۹۳/۱۰/۰۸ ۱۶:۴۸

ممنون از توجه شما

بدلیل اینکه شروع بحث از این پست بود و جناب نصیری به ان پاسخ داده بودند در اینجا دوباره مطرح کردم.

در بخش دوم با استفاده از افزونه مورد نظر و code generator آن کلاس سازنده ماشین حالت ایجاد می‌شود. یعنی در حال Design با رسم گرافیکی آن کلاس مزبور:

```
public class BlogPostStateMachine
  {
    // .... مثال قسمت دوم
```

```
  }
```

ساخته می‌شود و به گردش درآوردن شیء در آن هم بصورت واضح در بخش دوم توضیح داده شده است. اما نحوه ایجاد این کلاس بصورت dynamic و با استفاده از واکشی داده‌های مربوط به State ها، Transitionها و ... از بانک اطلاعاتی سئوال اصلی من است. همانگونه در پست قبلی اشاره کردم با استفاده از کلاس اشاره شده و ارسال لیست Stateها و Transitionها به آن می‌توان تنها با :

```
StateMachineRequest smr = new StateMachineRequest(smc, startId);
            smr.Configure();
```

ماشین حالت را ایجاد و کانفیگ نمود و به آن دسترسی داشت. البته بررسی شروط هنگام تغییر وضعیت یا هنگام ورود و خروج از یک وضعیت یکی از ابهامات آن است.

نویسنده: امیران
تاریخ: ۱۳۹۳/۱۰/۱۰ ۱۰:۱۸

بررسی شروط مانند GuardFromStateToAnother، OnExit ، OnEntry هم به روش زیر مرتفع می‌شود:

```
var smr = new StateMachineRequest(workflowData, startId);
            smr.GuardClauseFromToTrigger = new
StateMachineRequest.GuardClauseDelegate(this.OnFromStateToState);
            smr.OnEntry = new StateMachineRequest.EntryExitDelegate(this.OnEntryState);
            smr.OnExit = new StateMachineRequest.EntryExitDelegate(this.OnExitState);
            smr.Configure();

public bool OnFromStateToState(string id)
        {
            // TODO check can go to next state
            return true;
        }

        public void OnEntryState(string stateId)
        {
            // TODO
        }

        public void OnExitState(string stateId)
        {
            // TODO save data + save state + send an email ,Etc
        }
```