Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Ambientes Virtuais de Execução

2017

A biblioteca **HtmlReflect** que permite obter uma representação em HTML para um objecto de domínio ou *array* de objectos. Esta biblioteca é usada na aplicação web **MovHubWebApp** para automatizar a construção de vistas HTML (*views*) com base em objectos de domínio (exemplo: *movie*, *person*, etc).

A arquitectura desta aplicação encontra-se dividida nos componentes apresentados na figura seguinte, que têm correspondência com os projectos com o mesmo nome que integram a solução MovHub.sln.

|  |  |
| --- | --- |
| C:\MyFolder\ISEL\pg5 ave - 2017-2018 - 1º sem\trabs\movhub-arch.jpg |  |

A aplicação MovhubWebApp é uma aplicação Web que disponibiliza informação sobre filmes. A fonte de dados desta aplicação é dada pela API RESTful: <https://api.themoviedb.org>. Por sua vez o acesso a esta fonte de dados é mediado pela classe **TheMovieDbClient** da biblioteca **MovHubDb**. As informações da fonte de dados são obtidas dos seguintes *endpoints* da API RESTful <https://api.themoviedb.org>:

1. Pesquisa: [*https://developers.themoviedb.org/3/search/search-movies*](https://developers.themoviedb.org/3/search/search-movies)
2. Detalhes de um filme: <https://developers.themoviedb.org/3/movies>
3. Lista de personagens de um filme: <https://developers.themoviedb.org/3/movies/get-movie-credits>
4. Detalhes de um actor: <https://developers.themoviedb.org/3/people>
5. Participações de um actor: <https://developers.themoviedb.org/3/people/get-person-movie-credits>

Exemplos:

1. *https://api.themoviedb.org/3/search/movie?api\_key=\*\*\*\*\*&query=war+games*
2. *https://api.themoviedb.org/3/movie/860?api\_key=\*\*\*\*\**
3. *https://api.themoviedb.org/3/movie/860/credits?api\_key=\*\*\*\*\**
4. *https://api.themoviedb.org/3/person/3489?api\_key=\*\*\*\*\**
5. *https://api.themoviedb.org/3/person/4756/movie\_credits?api\_key=\*\*\*\*\**

A aplicação MovhubWebApp disponibiliza 4 *endpoints* cujas páginas têm um modelo de navegação de acordo com o diagrama seguinte:



As imagens seguintes apresentam o aspecto final das páginas de MovHubWebvApp integradas com informação produzida pelo HtmlReflect, para os seguintes pedidos:

* http://localhost:3000/movies?title=drive
* http://localhost:3000/movies/1018
* http://localhost:3000/movies/1018/credits
* http://localhost:3000/person/15008/movies



A biblioteca **HtmlReflect** permite obter uma representação em HTML para um objecto de domínio ou *array* de objectos. Na listagem seguinte é apresentada a API da classe principal Htmlect:

|  |
| --- |
| public class Htmlect {  public string ToHtml(object obj) {...}  public string ToHtml(object[] arr) {...}  } |

Por exemplo, dado um objecto com as propriedades: OriginalTitle igual a “Mulholland Drive”, ReleaseDate com a data 2001-05-16 e VoteAverage igual a 7,6, então o método ToHtml(object obj) deve retornar para este objecto:

|  |
| --- |
| <ul class='list-group'>  <li class='list-group-item'><strong>OriginalTitle</strong>: Mulholland Drive</li>  <li class='list-group-item'><strong>ReleaseDate</strong>: 2001-05-16</li>  <li class='list-group-item'><strong>VoteAverage</strong>: 7,6</li>  </ul> |

Por sua vez, para um *array* com 4 instâncias do mesmo tipo que o caso anterior, com informação dos filmes: “*Drive*”, “*Mulholand Drive*”, “*Drive Angry*” e “*Drive*”, o método ToHtml(object[] obj) deve retornar para esse *array*:

|  |
| --- |
| <table class='table table-hover'>  <thead>  <tr><th>OriginalTitle</th><th>ReleaseDate</th><th>VoteAverage</th></tr>  </thead>  <tbody>  <tr><td>Drive</td><td>2011-08-06</td><td>7,4</td></tr>  <tr><td>Mulholland Drive</td><td>2001-05-16</td><td>7,6</td></tr>  <tr><td>Drive Angry</td><td>2011-02-24</td><td>5,3</td></tr>  <tr><td>Drive</td><td>1997-08-06</td><td>6,6</td></tr>  </tbody>  </table> |

O comportamento dos métodos ToHtml() pode ainda ser customizado através de anotações sobre as propriedades de uma entidade de domínio. Para tal a biblioteca HtmlReflect disponibiliza os seguintes *custom attributes*:

* HtmlIgnoreAttribute – Indica que a propriedade anotada não deve ser incluída no HTML produzido pelo método ToHtml().
* HtmlAsAttribute – Especifica o formato HTML em que deve ser apresentada a propriedade anotada. Para tal recebe na sua construção uma String com um template HTML que usa os identificadores {name} e {value} para indicar onde devem constar o nome e valor da propriedade. Exemplo de utilização:

[HtmlAs("<div class='card-body bg-light'><div><strong>**{name}**</strong>:</div>**{value}**</div>")]

A classe de domínio Movie apresenta alguns casos de utilização da anotação HtmlIgnore e HtmlAs:

|  |
| --- |
| public class Movie {  [HtmlIgnore] public int Id { get; set; }  public String OriginalTitle { get; set; }  [HtmlAs("<li class='list-group-item'><a href='/movies/{value}/credits'>Cast and crew </a></li>")]  public String Credits { get { return Id.ToString(); } }  [HtmlIgnore] public long Budget { get; set; }  public double Popularity { get; set; }  public double VoteAverage { get; set; }  public String ReleaseDate { get; set; }  [HtmlAs("<div class='card-body bg-light'><div><strong>{name}</strong>:</div>{value}</div>")]  public String Overview { get; set; }  } |

Dada por exemplo uma instância da classe Movie representando a informação do filme “*Before Sunset*” então o resultado do método ToHtml(object obj) para esse objecto deve ser:

|  |
| --- |
| <ul class='list-group'>  <li class='list-group-item'><strong>OriginalTitle</strong>:&nbspBefore Sunset</li>  <li class='list-group-item'><a href='/movies/80/credits'>Cast and crew for this movie</a></li>  <li class='list-group-item'><strong>VoteAverage</strong>:&nbsp7,6</li>  <li class='list-group-item'><strong>ReleaseDate</strong>:&nbsp2004-02-10</li>  <div class='card-body bg-light'><div><strong>Overview</strong>:</div>Nine years ago two strangers met by chance and spent a night in Vienna that ended before sunrise. They are about to meet for the first time since. Now they have one afternoon to find out if they belong together.</div>  </ul> |

O comportamento do método ToHtml(object[] obj) para *arrays* também é customizável pelas anotações HtmlIgnoreAttribute e HtmlAsAttribute.

A biblioteca HtmlEmit tem uma API semelhante à de HtmlReflect mas uma implementação distinta que melhora o seu desempenho.

Algumas das operações realizadas via Reflection tais como: ler nomes ou valores de propriedades de Entidade de Domínio (e.g. Person, Movie, etc ) são realizadas directamente com base em código IL emitido em tempo de execução através da API de System.Reflection.Emit.

O HtmlEmit é **configurável** para determinadas entidades de domínio. Assim o utilizador poderá **especificar** qual o HTML resultante da transformação de um objecto de domínio. Para tal o HtmlEmit disponibiliza os seguintes métodos que **especificam** para um determinado tipo de entidade de domínio T:

* ForTypeDetails<T>(Func<T, string> transf) – o HTML resultante da transformação de uma instância de T numa vista HTML de detalhes.
* ForTypeInTable<T>(IEnumerable<string> headers, Func<T, string> transf) – os títulos das colunas e o HTML parcial de uma linha de uma tabela resultante da transformação de cada elemento de uma sequência de objectos de T.
* ForSequenceOf<T>(Func<IEnumerable<T>, string> transf) -- o HTML resultante da transformação de uma sequência de objectos de T numa vista de listagem de várias instâncias de T.

Dada a instância HtmlEmit html, o exemplo seguinte ilustra cada um dos casos de utilização dos métodos anteriores:

|  |
| --- |
| html.ForTypeDetails<Student>(st =>  "<p>" +  "<strong>" + st.Name + "</strong> (" + st.Nr +")" +  "</p>");  html.toHtml(new Student(6543, "Ze")); // => <p><strong>Ze</strong> (6543)</p>  IEnumerable<string> headers = new string[]{ "Budget", "Vote Average" };  html.ForTypeInTable<Movie>(headers, mov =>  {  const string template = "<tr><td>{0}</td><td>{1}</td></tr>";  return String.Format(template, mov.Budget, mov.VoteAverage);  });  html.ForSequenceOf<Student>(stds =>  {  string liNrs = stds.Aggregate("", (prev, st) => prev + "<li>" + st.Nr + "</li>");  return "<h1>Student Numbers</h1><ul>" + liNrs + "</ul>";  }); |