Metodika pre kódové konvencie

Verzia 9.11.2015

Tabuľka 1. Autori

Autor	Rola
Peter Vrana	

Tabuľka 2. História zmien

Verzia	Dátum	Autor	Popis
1.0	9.11.2015	Peter Vrana	Vytvorenie dokumentu

Obsah

1 Úvod

Cieľom tejto metodiky je zjednotiť menné konvencie a formátovanie kódu napísaného v rôznych programovacích jazykoch (C#, CSHTML[1]) do jednotnej štandardizovanej podoby. Tak, aby bol vytvorený kód jednoduchšie čitateľný a možno aj pochopiteľný pre vývojárov. Táto metodika je určená pre všetkých členov tímu, vzhľadom na to, že všetci sa určitým spôsobom podieľajú na vývoji.

1.1. Písanie zdrojového kódu v jazyku C#

Jazyk C# je hlavným Business jazykom využitým v rámci projektu PerConIK, preto najväčšia časť tejto metodiky bude pojednávať o štandardných kódových konvenciách dodržiavaných v tomto jazyku.

1.1.1. Formátovanie kódu

V rámci projektu každý člen využíva rovnaké IDE (angl. Integrated development environment) Visual Studio 2015. Dôležité je, že kód sa vytvára v rovnakom editore, ktorý je súčasťou tohto vývojového prostredia. Rozhodli sme sa pre zachovanie štandardných konvencií nijako nemeniť prednastavené formátovacie funkcie tohto editora a využívať automatické formátovanie poskytnuté editorom. Dôležité je, aby každý člen vývojového tímu po dokončení úprav kódu použil klávesové skratky Ctrl+A (štandardná klávesová skratka pre označenie všetkého textu v otvorenom dokumente), Ctrl+K+D – (štandardná klávesová skratka pre automatické formátovanie textu podľa nastavenia editora). V prípade že sa jedná o rozsiahle triedy je vhodné pred odoslaním kódu do hromadného úložiska aplikovať aj klávesovú skratku Ctrl+M+O, ktorá „zbalí“ (angl. collapse) všetky regióny kódu. Pre vývojára, ktorý upravuje kód neskôr je oveľa jednoduchšie sa potom v kóde zorientovať a „rozbaliť“ ten región ktorý potrebuje upravovať. Poprípade aplikáciou klávesovej skratky Ctrl+M+L „rozbaliť“ všetky regióny.

1.1.2. Regióny kódu

Každá rozsiahlejšia trieda sa bude organizovať do regiónov, názvy regiónov sa píšu anglicky, so začiatočným veľkým písmenom. Rozsiahlejšia znamená, ak obsah ľubovoľného z regiónov, ktoré budú vymenované presahuje, rozsah obrazovky, budú na danú triedu aplikované regióny,

[1] C# - programovací jazyk zameraný na objektovo orientovanú paradigmu
CSHTML – HTML obohatený o C#

výnimkou sú triedy špecifické pre MVC architektúru (Controllers, ViewModels...), pri ktorých by takéto členenie nemalo zmysel.

Do úvahy pripadá nasledujúce všeobecné rozdelenie triedy do regiónov:

- Private Fields – uzatvára všetky privátne premenné
- Constructors – uzatvára všetky konštruktory
- Properties – uzatvára vlastnosti (setteri a getteri)
- Public Methods – uzatvára všetky verejné metódy
- Protected Methods – uzatvára všetky prekonané metódy
- Private Methods – uzatvára všetky privátne metódy

Trieda môže byť rozdelená na regióny aj podľa skupín funkcií, ktoré spolu úzko súvisia v prípade, že takéto rozdelenie zvyšuje čitateľnosť a prehľadnosť kódu. Triedy sú štruktúrovane jednotne, poradie je určené podľa vyššie uvedených regiónov, a teda na začiatku triedy sú privátne premenné nasledujú konštruktory...

```
#region Properties

 // názvy vlastností sa píšu v PascalCase
 // vlastnosti sú uzavreté v regióne Properties

 public string Name
 {
     get { return name; }
     set { this.name = value; }
 }
 public string SurName
 {
     get { return surName; }
     set { this.surName = value; }
 }

 public DateTime BirthDate
 {
     get { return birthDate; }
     set { birthDate = value; }
 }


 #endregion
```

Obr. 1 Použitie regiónu Properties na uzavretie vlastností do jedného celku.

Na obrázku 1 je vidieť aká je syntax pre vytváranie regiónov sa využíva kľúčové slovo region uvedené hashom - #, nasledované názvom regiónu.

1.1.3. Konvencie pomenovaní (angl. Naming conventions)

Všeobecne platí, že názvy sa píšu anglicky, tak, aby čo najlepšie vystihovali objekt, akciu, rozhranie..., ktoré zastupujú, sú zakázané názvy, ktoré nemajú zmysluplný význam. (jednopísmenové názvy premenných a pod.).

- **So začiatočným veľkým písmenom** sa píšu:
  Názvy tried, enumerácií, rozhraní, metód, regiónov, vlastností a menných priestorov ,v prípade viacslovných názvov sa každé ďalšie slovo v názve píše s úvodným veľkým písmenom. Názvy by mali byť podstatné mená, prípustné sú aj prídavné mená, ktoré ich špecializujú. V niektorých prípadoch môžu byť názvy tried zložené z názvu a suffixu špecifického pre .Net MVC ako napr. Controller, ViewModel, Model napr. AccountController. Názvy rozhraní sa zapisujú s prefixom I za ktorým nasleduje názov samotného rozhranie napr. IRepositoryManager.
- **So začiatočným malým písmenom** sa píšu:
  Názvy lokálnych premenných triedy, názvy parametrov metód, názvy lokálne definovaných premenných v rámci metód, pričom opäť platí, že pri viac slovných premenných sa každé ďalšie slovo píše so začiatočným veľkým písmenom. Špeciálnou skupinou sú lokálne premenné tried, ktoré sa píšu so začiatočným malým písmenom a s prefixom _, napr. _sessionManager.

1.1.4. Spoločné štandardy pre písanie kódu

Pre zavedenie spoločnej syntaxi a vyhýbaniu sa rôznym formám takzv. „syntactic sugger“ a chybám sme sa rozhodli zadefinovať nasledujúce spoločné konvencie.

- Vetvenie je vždy ohraničené kučeravými zátvorkami, aj keď v nich je len jeden príkaz, zamedzuje sa tak chybám.
- Pre vyhýbanie sa chybám sa bude všade používať takzvané bezpečné pretypovanie, pre referenčné typy pomocou príkazu **as** pre nereferenčné sa typ premennej najskôr skontroluje pomocou príkazu **is.**
- Metóda, ktorá vracia kolekciu, **nikdy nevracia null** namiesto toho vráti prázdnu kolekciu
- Všetky kritické volania sú obalené prostredníctvom **try/catch/finally** bloku a prípadne vzniknuté výnimky sa logujú.
- S prúdmi dát (angl. streams) sa manipuluje vo vnútri using blokov, tak sa automaticky zabezpečí uzatvorenie prúdu a uvoľnenie prostriedkov.
- Všade kde je to možné sa využíva implicitné typovanie pomocou **var.**
- Pre zjednodušenie kódu je povolené používať coallescence operátor **??.**

```
public void SecureCastingSample(BasicUser basicUser)
{
    // pre bezpečnú konverziu referenčných typov sa využíva
    // pretypovanie s kľúčovým slovom as pre referenčné typy
    // ak by sme použili tento tvar
    // var specialUser = (SpecialUser) basicUser; program by
    // skončil výnimkou ak basicUser nie je typu SpecialUser
    var specialUser = basicUser as SpecialUser;
    if(specialUser != null)
    {

    }
}

public void SecureCastingNonReferentialTypesSample(object someObject)
{
    // pre bezpečnú typovú konverziu nereferenčných typov sa využíva
    // nasledujúca syntax
    int number = 0;
    if (someObject is int)
    {
        number = (int)someObject;
    }
}
```

Obr. 2 Príklad bezpečné pretypovanie, vetvenie vždy používa, kučeravé zátvorky, použitie var

Zvyšné ukážky je možné pozrieť si v jednoduchej okomentovanej triede, ktorá vznikla na tieto účely, ktorá sa nachádza ... (todo treba napísať kde bude príloha stránka tímu ?)

1.1.5. Komentáre

Jazyk C# tak ako viaceré jazyky poskytuje 3 typy komentárov, blokový komentár, ktorý je vidieť v ukážke na Obr. 2 ďalej viac riadkový komentár so štandardnou syntaxou /**/ a komentár na dokumentovanie kódu, ktorý sa automaticky generuje po zadaní /// v IDE VS. V projekte je odporúčané používať iba prvý typ komentára na zakomentovávanie časti kódu, prípadne krátke vysvetlenie (Pre zakomentovanie väčšej časti kódu sa odporúča klávesová skratka Ctrl+K+C, odkomentovanie Ctrl+K+U). Na dokumentovanie sa odporúča používať generovaný komentár.

```
/// <summary>
/// Returns full name of user
/// </summary>
/// <param name="separator">Is used as separator between name and surName</param>
/// <param name="degree">Optional parameter, Scientific degree of user</param>
/// <returns></returns>

// názvy parametrov metód sa píšu v camelCase, všetky voliteľné
// parametre sa zapisujú na posledné miesta v liste parametrov spolu
// s priradením ich základnej hodnoty

public string GetFullName(string separator, string degree = null)
{
    // ak je typ premennej očividný z pravej strany priradenia, v tomto prípade
    // string
    // je vhodné využívať implicitné typovanie (var)
    var fullName = (degree != null) ? degree + this.name + separator +
                this.surName : this.name + separator + this.surName;
    return fullName;
}
```

Obr. 3 Komentáre

1.1.6. Adresárová štruktúra projektu

Unit testy sa umiestňujú do AdministrationPortal.Tests, integračné testy sa umiestňujú do AdmnistrationPortal.IntegrationTests, vlastný projekt sa volá AdministartionPortal.

Súbory v projekte majú adresárovú štruktúru danú použitým frameworkom .Net MVC, ktorú treba dodržovať nasleduje zoznam adresárov s očakávaným obsahom:

- AppStart – obsah daný frameworkom, umiestňujú sa sem migrácie DBS
- Common – spoločné triedy, rozhrania, enumerácie používané naprieč celým projektom
- Content - kaskádové štýly, obrázky
- Controllers – MVC kontroleri
- Data – triedy pre prístup k dátam (DBS)
- fonts – fonty
- Models – dátové modely
- Security – podporné triedy pre autorizáciu a autrntifikáciu
- Scripts - .js súbory
- Views – cshtml pohľady

1.2. Kód v .cshtml

Pre názvy cshtml dokumentov platia štandardné pravidlá pomenovania definované v tejto metodike s jednou výnimkou. Pohľady, ktoré sú zdieľané (častým príkladom je _loginPartial.cshtml) sa pomenúvajú s prefixom _.

```
<!-- Single statement block -->
@{ var myMessage = "Hello World"; }

<!-- Inline expression or variable -->
<p>The value of myMessage is: @myMessage</p>

<!-- Multi-statement block -->
@{
var greeting = "Welcome to our site!";
var weekDay = DateTime.Now.DayOfWeek;
var greetingMessage = greeting + " Here in Huston it is: " + weekDay;
}
<p>The greeting is: @greetingMessage</p>
```

Obr. 4 Ukážka .cshtml

Formátovanie .cshtml kódu budeme vykonávať rovnako ako tomu bolo pri C# kóde. Všeobecné odporúčanie je, aby cshtml neobsahoval žiaden kód obsahujúci nejakú logiku, ak tomu tak je tak nech ho je čo najmenej. Komentáre je tu možné písať dvoma spôsobmi a to pomocou @**@, alebo štandardným spôsobom pre html pomocou <!-- -->.