1.1.2018

Bc. Miloslav Moravec

Administer s.r.o.

iWAS – Webová aplikace

Dokumentace

# Obsah

[Obsah 1](#_Toc522457981)

[Úvod 3](#_Toc522457982)

[1 Controller 4](#_Toc522457983)

[1.1 HomeController 4](#_Toc522457984)

[1.2 LoginController 5](#_Toc522457985)

[1.3 LanguageController 6](#_Toc522457986)

[1.4 PrijemController 7](#_Toc522457987)

[1.4.1 Index() 7](#_Toc522457988)

[1.4.2 Index(string searchString, int resultCount) 8](#_Toc522457989)

[1.4.3 Details(int? id) 8](#_Toc522457990)

[1.4.4 Details(string searchString, int? id) 8](#_Toc522457991)

[1.4.5 Create() 9](#_Toc522457992)

[1.4.6 Create(prijem prijem) 9](#_Toc522457993)

[1.4.7 Edit(int? id) 9](#_Toc522457994)

[1.4.8 Edit(prijem prijem) 10](#_Toc522457995)

[1.4.9 DeleteConfirmed(int id) 10](#_Toc522457996)

[1.5 Prijem\_PolController 10](#_Toc522457997)

[1.5.1 Details(int? id) 10](#_Toc522457998)

[1.5.2 VyberZbozi(prijem\_pol model) 10](#_Toc522457999)

[1.5.3 Create(int? id) 11](#_Toc522458000)

[1.5.4 Create(prijem\_pol prijem\_pol, int? id) 11](#_Toc522458001)

[1.5.5 Edit(int? id) 11](#_Toc522458002)

[1.5.6 Edit(prijem\_pol prijem\_pol) 12](#_Toc522458003)

[1.5.7 DeleteConfirmed(int id) 12](#_Toc522458004)

[1.6 ZboziController 12](#_Toc522458005)

[1.6.1 Index() 13](#_Toc522458006)

[1.6.2 Index(string searchString, int resultCount) 13](#_Toc522458007)

[1.6.3 Details(int? id) 14](#_Toc522458008)

[1.6.4 Create() 14](#_Toc522458009)

[1.6.5 Create(zbozi zbozi) 14](#_Toc522458010)

[1.6.6 Edit(int? id) 15](#_Toc522458011)

[1.6.7 Edit(zbozi zbozi) 15](#_Toc522458012)

[1.6.8 DeleteConfirmed(int id) 16](#_Toc522458013)

[1.6.9 ExportSkladovaKarta(int id) 16](#_Toc522458014)

[1.7 InventurniSestavaController 17](#_Toc522458015)

[1.7.1 Index(string datum) 17](#_Toc522458016)

[1.8 SkladovaPrijemkaController 18](#_Toc522458017)

[1.8.1 Export(int id, ExportType exportType) 18](#_Toc522458018)

[1.8.2 ExportXLS(int id, List<prijemka\_skladova> report) 19](#_Toc522458019)

# Úvod

Následující text představuje dokumentaci k webové části aplikace „iWAS“ společnosti Administer s.r.o.

Webová aplikace je implementována pomocí frameworku ASP.NET s MVC architekturou. MVC architektura umožňuje oddělení aplikační logiky od datové a zobrazovací. Data jsou uložena v databázi MSSQL; zobrazení těchto dat zajišťuje značkovací jazyk HTML. V tomto se dále používá kaskádových stylů frameworku Bootstrap verze 3.3.7 a JavaScriptu, konkrétně jQuery verze 3.3.1 a Bootstrap verze 3.3.7 a jiných.

# Controller

Controller (dále jen kontrolér) je třída, která má na starosti tok událostí v aplikaci a aplikační logiku. Kontrolér by se měl nacházet mezi každým aktérem a případem užití. V tomto případě kontroléry odrážejí databázové objekty (tabulky, pohledy).

Aplikace obsahuje tyto kontroléry:

* HomeController,
* InventurniSestavaController,
* LanguageController,
* LoginController,
* Prijem\_PolController,
* PrijemController,
* SkladovaPrijemkaController a
* ZboziController.

## HomeController

Při prvním spuštění aplikace se volá metoda Application\_Start() ve třídě MvcApplication souboru Global.asax. Tato pak volá metodu RegisterRoutes ve třídě RouteConfig. Metoda RegisterRoutes() vytvoří směrovací tabulku. Výchozí směrovací tabulka obsahuje jednu trasu (výchozí). V tomto případě se volá kontrolér Home a jeho metoda Index().

Třída HomeController obsahuje metody Index() a VyberZakaznika(Zakaznik model).

Metoda Index() postupuje následovně:

* Kontrola, zda je v Session nastaven atribut „userID“. Čili zda je někdo přihlášen.
  + Pokud není nastaven, přesměruje se tok programu do metody Index() kontroléru Login.
* Deklarace ViewBag nazvaného id\_zakaznik, do kterého se zároveň přiřazuje seznam všech zákazníků ve formě SelectListu.

Metoda VyberZakaznika(Zakaznik model) silně navazuje na Index(). Po vybrání zákazníka z výše zmíněného SelectListu se po potvrzení formuláře tok programu dostane sem.

* Zde se ID zvoleného zákazníka přiřadí do Session jako atribut „zakaznikID“.
* Podle ID zákazníka se poté nalezne v databázi jméno daného zákazníka, které se přiřadí do Session s atributem „zakaznikJmeno“.
* Nakonec přesměrování zpět do metody Index kontroléru Home.

V metodě Index() se vyskytl pojem ViewBag. ViewBag je společně s ViewData a TempData jedna z možností, jak předávat data mezi kontrolérem a zobrazením. ViewBag disponuje následujícími vlastnostmi:

* Pomáhá udržovat data při přechodu z kontroléru do zobrazení.
* Hodnota krátkodobé hodnoty se stává null, když dojde k přesměrování.
* Nevyžaduje specifický datový typ. Lze do něj přiřadit v podstatě cokoliv.

Jinými slovy jedná se o malý objekt, do kterého lze z kontroléru uložit informaci, kterou chceme zobrazit v následujícím zobrazení. Po přepnutí na jiný kontrolér, či zobrazení se tato data mažou. S ViewBagy se v této dokumentaci ještě setkáme.

## LoginController

K tomu, aby si mohl uživatel zvolit zákazníka, se kterým chce pracovat (pokud není zákazníkem on sám), musí být v Session vyplněn atribut „userID“ a právě k tomu slouží kontrolér Login.

Tato třída obsahuje metody Index(), Authorize(uzivatel model) a Logout().

Metoda Index() pouze kontroluje, zda už uživatel není přihlášený, v takovém případě ho přesměruje do metody Index() kontroléru Home. Jinak zobrazí formulář pro přihlášení.

Po vyplnění přihlašovacího formuláře se zavolá metoda Authorize s parametrem typu uzivatel. V něm je vyplnění přihlašovací jméno a heslo, které uživatel zadal do formuláře. Podle těchto údajů se uživatel nalezne v databází. Ověří se, zda byl někdo nalezen, pokud ne – vyplní se ViewBag LoginErrorMessage zprávou o tom, že heslo, či přihlašovací jméno, bylo nesprávné a zobrazí se opět přihlašovací formulář. Pokud byl uživatel nalezen, nastaví se jeho ID jako atribut „userID“ do Session. Dále se nastaví do Session jeho jméno („userName“) a ID jeho uživatelského přístupu („userRole“). Pokud je ID uživatelského přístupu rovno 1, znamená to, že uživatel je zákazník. Pomocí ID uživatele se v tabulce uživatelů nalezne ID zákazníka, které se uloží do Session jako „zakaznikID“. Jeho jméno pak jako „zakaznikJmeno“. Následuje pak přesměrování do metody Index() kontroléru Home.

Metoda Logout() má za úkol jediné, zrušit všechny uložené objekty v Session a přesměrovat uživatele do metody Index() kontroléru Login.

## LanguageController

Tento kontrolér obsahuje jednu jedinou metodu – Change(String LanguageAbbrevation, String ControllerName, String ViewName).

Jde o změnu jazyka veškerého textu v zobrazovací části aplikace. Aplikace disponuje dvěma jazyky: češtinou a angličtinou. Texty jsou umístěné ve složce Resources, kde každý soubor odpovídá kontroléru (není to povinné, jen pro přehlednost). Každý soubor je v základu psán anglicky, přičemž k němu patří i jeho anglická verze.

Příklad:

*HomeTexts.resx*

*HomeTexts.en.resx*

Soubor je v podstatě tabulka, kde levým sloupcem definujeme název výrazu a v pravém jeho hodnotu. Důležité je, aby názvy výrazů správně seděly jak v české, tak v anglické verzi. Dále musí být modifikátor přístupu nastaven na Public, aby se k souboru dalo přistoupit odkudkoliv z aplikace.

Metoda Change má následující parametry:

* LanguageAbbrevation
  + Typu String.
  + Upřesňuje, na jakou jazykovou verzi se má aplikace přepnout.
  + Vstupní hodnoty buď „cs“ nebo „en“.
* ControllerName
  + Typu String.
  + Představuje název kontroléru, ze kterého byla metoda volána.
* ViewName
  + Typu String.
  + Představuje název zobrazení, ze kterého byla metoda volána.

Po zavolání metody se zkontroluje, zda parametr LanguageAbbrevation není prázdný. Pokud není, nastaví se kultura aplikace podle hodnoty LanguageAbbrevation. Dále se vytvoří Cookie s názvem „Language“, do které se jako hodnota vloží právě LanguageAbbrevation, aby i po restartu aplikace byl nastaven stále stejný jazyk. Následuje přesměrování do metody a kontroléru podle zadaných parametrů ControllerName a ViewName.

Třída MvcApplication v souboru Global.asax má kromě výše zmíněné metody Application\_Start() také metodu Application\_BeginRequest(), která vyhledá právě Cookie se jménem „Language“ a podle něj nastaví kulturu aplikace. Pokud Cookie nebyla nalezena, defaultně se nastavuje kultura aplikace na „cs“.

## PrijemController

Poté, co se už umíme do aplikace přihlásit a měnit její jazyk, si už můžeme konečně začít hrát. Kontrolér Prijem se stará o všechny operace spojené s příjemkami. Nalezneme zde následující metody:

* Index(),
* Index(string searchString, int resultCount),
* Details(int? id),
* Details(string searchString, int? id),
* Create(),
* Create(prijem prijem),
* Edit(int? id),
* Edit(prijem prijem),
* DeleteConfirmed(int id).

### Index()

Metoda, která je volána http metodou GET. Zobrazuje informace o všech příjemkách zákazníka.

* Kontrola přihlášení uživatele a případné přesměrování na přihlašovací stránku.
* Kontrola, zda byl vybrán, nebo je přihlášen, zákazník a případné přesměrování na hlavní stránku.
* Vybrání všech příjemek z databáze, které patří danému zákazníkovi.
* Seřazení vytvořeného seznamu sestupně podle ID příjemky.
* Přečtení Cookie s názvem „ResultCount“, která nese informaci kolik záznamů má být zobrazeno na stránce.
  + Pokud tato Cookie neexistuje, zobrazí se defaultně 10 výsledků.
* Zobrazení View s listem příjemek jako parametrem.

### Index(string searchString, int resultCount)

Metoda, která je volána http metodou POST při změně počtu výsledků na stránku, či vyhledání zadaného textu.

* Vybrání všech příjemek z databáze, které patří zvolenému zákazníkovi.
* Kontrola, zda hodnota searchString není prázdná.
  + Vyfiltrování seznamu příjemek podle hledaného textu nacházejícího se buď v ID příjemky nebo v dokladech.
* Seřazení seznamu sestupně podle ID příjemky.
* Kontrola, zda hodnota resultCount je větší než 0.
  + Vytvoření nové Cookie s názvem „ResultCount“ do které se uloží hodnota resultCount.
  + Nastavení platnosti Cookie na jeden den.
  + Uložení hodnoty resultCount do ViewBag resultCount.
* Pokud hodnota resultCount není větší než 0 pak:
  + Přečtení Cookie s názvem „resultCount“.
  + Pokud neexistuje, nastavení defaultní hodnoty na 10.
* Zobrazení View s listem jako parametrem a určitým počtem výsledků.

### Details(int? id)

Metoda, která je volána http metodou GET. Zobrazuje detail a seznam položek příjemky. Jediná nepracuje s tabulkou prijem, ale s pohledem prijem\_details.

* Kontrola, zda bylo zadané ID příjemky. Pokud nebylo, přesměrování do metody Index() kontroléru prijem.
* Vybrání všech výsledků z pohledu prijem\_details, kde id\_prijem odpovídá parametru id.
* Zobrazení View s listem jako parametrem.

### Details(string searchString, int? id)

Metoda, která je volána http metodou POST při vyhledání určité položky v příjemce. Pracuje podobně jako metoda Index(string searchString, int resultCount). Rozdíl je v absenci nastavení počtu výsledků a jinému přístupu k seznamu výsledků.

Řeší se zde hlavně problém, kdy podle zadaného řetězce nebyla nalezena žádná položka. Pak je seznam prázdný a není ani čím vyplnit detaily o příjemce, které musí být k dispozici vždy, i kdyby příjemka nedisponovala žádnými položkami.

Proto se ještě před vyfiltrováním seznamu hledaným řetězcem uloží první záznam. Pokud skutečně v seznamu nebylo nic nalezeno a tento je pak prázdný, z uloženého záznamu se odstraní údaje o položce a vloží se pak do seznamu pouze s informacemi o příjemce. Metoda pak zobrazí View s tímto seznamem jako parametrem.

### Create()

Metoda, která je volána http metodou GET. Zobrazuje formulář pro vytvoření nové příjemky.

Zde se pracuje hlavně s objektem ViewBag, do kterého se zadefinuje:

* id\_dodavatel
  + seznam všech dodavatelů, kteří spolupracují se zákazníkem.
* id\_zakaznik
  + Seznam všech zákazníků. Defaultně nastaven zákazník, se kterým se pracuje, nebo který je přihlášen.
* id\_palety
  + Seznam všech palet.

Následuje zobrazení View obsahující formulář k vytvoření příjemky.

### Create(prijem prijem)

Metoda, která je volána http metodou POST při vyplnění formuláře k vytvoření příjemky.

Všechny položky z formuláře se vložily do objektu prijem, který přišel jako parametr. Neobsahuje ovšem údaje o uživateli, který jej vytvořil a o datu založení. Tyto informace se doplní a objekt prijem se vloží do seznamu všech příjmů. Uloží se v databázi změny a proběhne přesměrování zpět do metody Index().

### Edit(int? id)

Metoda, která je volána http metodou GET k zobrazení předvyplněného formuláře pro úpravu stávající příjemky. Pomocí id se nalezne příjemka, která se pošle jako parametr do View. Zbytek operací ohledně ViewBagů jsou stejné jako u metody Create().

### Edit(prijem prijem)

Metoda, která je volána http metodou POST po úpravě příjemky.

* Nejprve se podle id příjemky vyhledá stávající příjemka (old) v databázi.
* Do tohoto objektu (old) se přepíšou informace z parametru prijem.
* Příjemka old s aktualizovanými atributy se vrátí zpět do databáze.
* Uloží se změny v databází.
* Proběhne přesměrování zpět do metody Index().

### DeleteConfirmed(int id)

Metoda, která je volána http metodou POST. Na rozdíl od ostatních metod k sobě nemá bratrskou GET metodu. Tato metoda je volána pomocí JavaScriptu. Soubor: deleteItem.js, funkce: confirmation(id, message, path) – bude popsáno dále.

## Prijem\_PolController

Tento kontrolér obsahuje operace pro práci s položkami dané příjemky. Obsahuje metody:

* Details(int? id),
* VyberZbozi(prijem\_pol model),
* Create(int? id),
* Create(prijem\_pol prijem\_pol, int? id),
* Edit(int? id),
* Edit(prijem\_pol prijem\_pol),
* DeleteConfirmed(int id).

### Details(int? id)

Metoda, která je volána http metodou GET. Na základě id položky se najde daná položka a pošle se jako parametr do View.

### VyberZbozi(prijem\_pol model)

Metoda, která je volána http metodou POST při vytváření nové položky. Do Session[„zbozi“] se nastaví hodnota id\_zbozi z parametru model. Tok programu se přesměruje zpět do metody Create (int? id), kde jako id se nastaví id příjemky, do které vkládáme novou položku.

### Create(int? id)

Metoda, která je volána http metodou GET. Připravuje formulář pro vytvoření nové položky příjemce.

* Kontrola, zda id není prázdné.
* Kontrola, zda je přiřazena hodnota k Session[„zbozi“].
* Nalezení zboží podle id zboží.
* Pokud bylo zboží nalezeno pak:
  + Přiřazení názvů měrných jednotek do ViewBagů.
  + Přiřazení koeficientu měrné jednotky 2 do ViewBag.koeficient2.
  + Přiřazení koeficientu měrné jednotky 3 do ViewBag.koeficient3.
  + Přiřazení seznamu zboží a vybrání vybraného zboží do ViewBag.id\_zbozi.
* Pokud nebylo zboží nalezeno pak:
  + Přiřazení seznamu zboží do ViewBag.id\_zbozi.
* Přiřazení id příjemky do ViewBag.id;
* Zobrazení View.

### Create(prijem\_pol prijem\_pol, int? id)

Metoda, která je volána http metodou POST při vyplnění formuláře pro vytvoření nové položky příjemky. Do parametru prijem\_pol se doplní informace o datu založení a uživateli, který položku vytvořil. Parametr se pak vloží do seznamu položek a uloží se změny v databázi. Proběhne přesměrování do kontroléru prijem, metody Details(int? id), kde id je náš parametr id, alias id příjemky.

Pokud nastane chyba, vypíše se do ViewBag.Create\_Error a zobrazí se opět View pro vytvoření položky.

### Edit(int? id)

Metoda, která je volána http metodou GET pro zobrazení předvyplněného formuláře pro úpravu existující položky příjemky.

* Kontrola, zda id není prázdné.
* Nalezení položky podle id.
* Přiřazení seznamu zboží aktuálního zákazníka do ViewBag.id\_zbozi.
* Přiřazení množství v měrných jednotkách do ViewBagů.
* Zobrazení View s nalezenou položkou jako parametrem.

### Edit(prijem\_pol prijem\_pol)

Metoda, která je volána http metodou POST po vyplnění formuláře pro úpravu stávající položky příjemky.

* Nalezení položky (old) v databázi podle id parametru prijem\_pol.
* Přepsání atributů ve staré položce (old) novými daty z parametru prijem\_pol.
* Vložení položky (old) zpět do databáze.
* Uložení změn v databázi.
* Přesměrování do kontroléru prijem, metody Details s parametrem id\_prijem.
* Pokud nastala chyba, vypíše se do ViewBag.Edit\_Error.
* Zobrazení View s parametrem prijem\_pol.

### DeleteConfirmed(int id)

Metoda, která je volána http metodou POST.

* Nalezení položky v databázi podle id.
* Odstranění položky z databáze.
* Uložení změn v databází.
* Přesměrování do kontroléru prijem, metody Details s parametrem id\_prijem.

## ZboziController

Kontrolér obsahující metody pro práci se zbožím daného zákazníka. Tyto metody jsou:

* Index(),
* Index(string searchString, int resultCount),
* Details(int? id),
* Create(),
* Create(zbozi zbozi),
* Edit (int? id),
* Edit(zbozi zbozi),
* DeleteConfirmed(int id),
* ExportSkladovaKarta(int id),
* GetName(int id),
* RemoveSpecialCharacters(string str).

### Index()

Metoda, která je volána http metodou GET. Zobrazuje tabulku zboží zákazníka.

* Kontrola přihlášeného uživatele.
* Kontrola přiřazeného zákazníka.
* Vyfiltrování seznamu zboží podle id zákazníka.
* Seřazení zboží sestupně podle id zboží.
* Přečtení Cookie s názvem „ZboziCount“, která nese informaci kolik záznamů má být zobrazeno na stránce.
  + Pokud tato Cookie neexistuje, zobrazí se defaultně 30 výsledků.
* Přiřazení počtu výsledků do ViewBag.ResultCount.
* Vybrání prvních záznamů ze seznamu podle definovaného počtu výsledků.
* Zobrazení View se seznamem zboží jako parametrem.

### Index(string searchString, int resultCount)

Metoda, která je volána http metodou POST při změně počtu výsledků na stránku, či hledání zadaného textu.

* Vyfiltrování zboží, které patří danému zákazníkovi.
* Kontrola, zda hodnota searchString není prázdná.
  + Vyfiltrování seznamu zboží podle hledaného textu nacházejícího se buď v kódech zboží či jeho názvu.
* Seřazení seznamu sestupně podle ID zboží.
* Kontrola, zda hodnota resultCount je větší než 0.
  + Vytvoření nové Cookie s názvem „ZboziCount“ do které se uloží hodnota resultCount.
  + Nastavení platnosti Cookie na jeden den.
  + Uložení hodnoty resultCount do ViewBag resultCount.
* Pokud hodnota resultCount není větší než 0 pak:
  + Přečtení Cookie s názvem „ZboziCount“.
  + Pokud neexistuje, nastavení defaultní hodnoty na 30.
* Zobrazení View s listem jako parametrem a určitým počtem výsledků.

### Details(int? id)

Metoda, která je volána http metodou GET pro zobrazení všech informací o zboží.

* Kontrola přihlášeného uživatele.
* Kontrola vybraného zákazníka.
* Nalezení zboží podle parametru id.
* Nalezení názvu kategorie zboží podle id zboží kategorie.
* Přiřazení názvu kategorie do ViewBag.Zbozi\_Kategorie.
* Zobrazení View se zbožím jako parametrem.

### Create()

Metoda, která je volána http metodou GET pro zobrazení formuláře k vytvoření nového zboží.

* Kontrola přihlášeného uživatele.
* Vytvoření SelectListu obsahujícího seznam měrných jednotek.
* Přiřazení SelectListu do ViewBagů.
* Přiřazení SelectListu kategorií zboží do ViewBag.kategorie\_zbozi.
* Zobrazení View.

### Create(zbozi zbozi)

Metoda, která je volána http metodou POST po vyplnění formuláře pro vytvoření nového zboží.

* Nastavení atributu id\_uzivatel do parametru zbozi.
* Nastavení atributu id\_zakaznik do parametru zbozi.
* Nastaveni atributu datum\_cas\_zalozeni do parametru zbozi.
* Vložení objektu zbozi do seznamu zboží.
* Uložení změn v databázi.
* Přesměrování do metody Index().
* Pokud nastala nějaká chyba, pak vypsání chybové zprávy do ViewBag.Create\_Error.
* Vytvoření SelectListu obsahujícího seznam měrných jednotek.
* Přiřazení SelectListu do ViewBagů.
* Přiřazení SelectListu kategorií zboží do ViewBag.kategorie\_zbozi.
* Zobrazení View s formulářem.

### Edit(int? id)

Metoda, která je volána http metodou GET zobrazující předvyplněný formulář pro editaci existujícího zboží.

* Kontrola přihlášeného uživatele.
* Kontrola nastaveného parametru id.
* Nalezení zboží podle id.
* Kontrola, že zboží bylo nalezeno.
* Kontrola, zda se zboží vyskytuje jako položka v nějaké příjemce.
  + Pokud ano, nastavení ViewBag.Disabled na true;
  + Pokud ne, nastavení ViewBag.Disabled na false;
  + Tento ViewBag.Disabled slouží k povolení úprav důležitých atributů zboží, jako jsou měrné jednotky a jejich koeficienty.
* Vytvoření SelectListu obsahujícího seznam měrných jednotek, s přednastavenou měrnou jednotkou 1.
* Vytvoření SelectListu obsahujícího seznam měrných jednotek, s přednastavenou měrnou jednotkou 2.
* Vytvoření SelectListu obsahujícího seznam měrných jednotek, s přednastavenou měrnou jednotkou 3.
* Přiřazení SelectListů do ViewBagů.
* Přiřazení SelectListu kategorie zboží s přednastavenou kategorií zboží do ViewBag.kategorie\_zbozi.
* Zobrazení View se zbožím jako parametrem.

### Edit(zbozi zbozi)

Metoda, která je volána http metodou POST po vyplnění formuláře pro editaci zboží.

* Nalezení stávajícího zboží (old) podle id parametru zbozi.
* Přepsání atributů zbozi (old) atributy nového parametru zbozi.
* Přepsání zboží v databázi novým (old) objektem.
* Uložení změn v databází.
* Přesměrování do metody Index().
* V případě chyby příprava formuláře pro opětovnou editaci a zobrazení View.

### DeleteConfirmed(int id)

Metoda, která je volána http metodou POST pro odstranění zboží podle jeho id.

* Nalezení zboží podle id.
* Odstranění zboží z databáze.
* Uložení změn v databází.
* Přesměrování do metody Index().

### ExportSkladovaKarta(int id)

Metoda, která je volána http metodou GET. Má za úkol vytvořit XLS soubor obsahující informace o zboží a jeho životu ve skladě. Metoda využívá knihovnu OfficeOpenXml a databázovou proceduru „SP\_report\_skladova\_karta(int? idZbozi)“.

* Vytvoření listu z procedury podle id zboží.
* Vytvoření názvu souboru pomocí privátní metody GetName(int id).
* Vytvoření objektu ExcelPackage.
* Vytvoření objektu ExcelWorksheet pomocí objektu ExcelPackage a názvu souboru.
* Nastavení prvního řádku XLS souboru na tučné písmo (pro hlavičku tabulky).
* Vytvoření hlavičky tabulky pomocí přiřazení názvů sloupců do daných buněk.
* Inicializace proměnné rowStart na 2. Jelikož první řádek je použit jako hlavička tabulky, její plnění začne na řádku číslo 2.
* Cyklické procházení listu a plnění buněk hodnotami z listu. Při každém průchodu se provede inkrementace proměnné rowStart.
* Nastavení šířky sloupců jako AutoFit, aby se šířka řídila velikostí hodnoty, která se v ní nachází.
* Vyčištění objektu Response.
* Nastavení obsahu a hlavičky souboru.
* Zapsání do objektu Response bytové pole objektu ExcelPackage.
  + V tuto chvíli by se měl začít stahovat souboru.
* Ukončení objektu Response.
* Přesměrování do metody Index().

#### GetName(int id)

Pomocná metoda k předchozí. Sestaví název souboru podle názvu zboží a kódu zboží. Upraví název pomocí privátní metody RemoveSpecialCharacters(String str). Zkontroluje, zda název zboží nepřesahuje 30 znaků. V takovém případě název ořízne a vrátí zpět.

#### RemoveSpecialCharacters(String str)

Pomocná metoda k předchozí. Využívá třídy StringBuilder. Projde vstupní řetězec str a zkontroluje každý znak, zda odpovídá buď číslům 0 – 9, nebo písmenům A – Ž nebo a – ž nebo podtržítku. Pak znak vloží do objektu třídy StringBuilder. Pokud je znak jiného původu nepoužije se. Metoda vrací vyčištěný řetězec o speciální znaky.

## InventurniSestavaController

Tento kontrolér má pouze jedinou odpovědnost. Vytvořit inventurní sestavu zboží daného zákazníka a uložit ji jako XLS soubor. Nachází se zde tedy pouze metody:

* Index(string datum) a
* GetReport(DateTime? date).

### Index(string datum)

Metoda, která je volána http metodou POST. Využívá knihovny OfficeOpenXml pro export do XLS souboru a databázového pohledu „report\_inventurni\_sestava\_2“. Inventurní sestava se zhotovuje pro všechny zboží do určitého data. Datum je vstupní parametr typu string, který se pomocí metody Split() rozdělí do pole podle tečky. Tím dostaneme v jednom prvku pole den, v druhém prvku měsíc a ve třetím prvku rok. Z těchto prvků pak vytvoříme plnohodnotné datum typu DateTime.

Tento způsob je aplikován proto, protože přechody mezi českou a anglickou verzí aplikace (viz LanguageController) měnily jazykový charakter datumu. V české verzi je sled prvků klasický: den, měsíc, rok. Kdežto v anglické verzi je sled: měsíc, den, rok. Z tohoto důvodu se datum předává přímo ze vstupu formuláře jako prostý string.

* Vytvoření seznamu záznamů pomocnou metodou GetReport(DateTime? date).
* Vytvoření objektu ExcelPackage.
* Sestavení názvu souboru spojením řetězce „Inventurní sestava“ a zvoleného datumu.
* Vytvoření objektu ExcelWorksheet pomocí objektu ExcelPackage a názvu souboru.
* Nastavení prvního řádku XLS souboru na tučné písmo (pro hlavičku tabulky).
* Vytvoření hlavičky tabulky pomocí přiřazení názvů sloupců do daných buněk.
* Inicializace proměnné rowStart na 2. Jelikož první řádek je použit jako hlavička tabulky, její plnění začne na řádku číslo 2.
* Cyklické procházení listu a plnění buněk hodnotami z listu. Při každém průchodu se provede inkrementace proměnné rowStart.
* Nastavení šířky sloupců jako AutoFit, aby se šířka řídila velikostí hodnoty, která se v ní nachází.
* Vyčištění objektu Response.
* Nastavení obsahu a hlavičky souboru.
* Zapsání do objektu Response bytové pole objektu ExcelPackage.
  + V tuto chvíli by se měl začít stahovat souboru.
* Ukončení objektu Response.
* Zobrazení View.

#### GetReport(DateTime? date)

Pomocná metoda k předchozí. Na základě datumu jako vstupního parametru a id zákazníka vyfiltruje pohled „report\_inventurni\_sestava\_2 a vrátí seznam záznamů.

## SkladovaPrijemkaController

Tento kontrolér slouží k exportování informací o dané příjemce. Obsahuje též výčtový typ ExportType, který má dva atributy: PDF, Excel. Tento určuje, jaký výstup se má provést. Nacházejí se zde následující metody:

* Export(int id, ExportType exportType) a
* ExportXLS (int id, List<prijemka\_skladova> report).

### Export(int id, ExportType exportType)

Metoda, která je volána http metodou GET. Vstupními parametry jsou id příjemky a typ exportu. Využívá databázového pohledu „prijemka\_skladova“ a knihovny Microsoft.Reporting.WebForms.

* Vytvoření objektu ReportViewer.
* Vytvoření objektu LocalReport.
* Vyčištění atributu DataSources objektu LocalReport.
* Přiřazení cesty k RDLC souboru do atributu ReportPath objektu LocalReport.
  + RDLC soubor je model vytvořený v Microsoft Rdlc Report Designeru, ve kterém se připraví vzhled reportu.
* Vytvoření listu výsledků pohledu vyfiltrovaného podle id příjemky.
* Vytvoření objektu ReportDataSource.
* Přiřazení názvu DataSetu do atributu Name objektu ReportDataSource.
* Přiřazení listu do atributu Value objektu ReportDataSource.
* Vložení objektu ReportDataSource do atributu DataSources objektu LocalReport.
* Deklarace řetězců: mimeType, encoding, fileNameExtension, reportType, které budou sloužit k vytvoření výsledného souboru.
* Kontrola hodnoty ExportType.
  + Pokud je ExportType typu Excel -> přesměrování do metody ExportXLS(int id, List<prijemka\_skladova> report).
* Deklarace pole řetězců streams.
* Deklarace pole typu Warnings.
* Deklarace pole bytů renderedBytes.
  + Do tohoto pole se pomocí metody Render() objektu LocalReport vyrenderuje výsledný soubor.
* Příprava hlavičky souboru.
* Metoda vrací soubor společně s polem bytů a koncovkou souboru jako parametry.

### ExportXLS(int id, List<prijemka\_skladova> report)

Pomocná metoda k předchozí. Na vstupu je id příjemky a připravený list k exportu do souboru XLS. Princip je stejný jako u exportu skladové karty či inventurní sestavy.

# JavaScript

Jak již bylo zmíněno výše, aplikace využívá primárně knihovnu jQuery a Bootstrap. Není to ale samozřejmě všechno. Využívá i spoustu jiných skriptů, ať už stažených, online nebo vlastnoručních. Mezi stažené a online skripty patří:

* jquery-ui.js,
  + Odnož knihovny jQuery zaměřená na uživatelské rozhraní. Stará se o různé efekty a v případě této aplikace se využívá pro zobrazení okénka s výběrem datumu.
* datepicker-cs.js,
  + JavaScript starající se o obsahovou část okénka s výběrem datumu.
* chosen.jquery.js.
  + V této aplikaci využíván pro výběr zboží při vytváření nové položky příjemce. Díky tomuto speciálnímu výběru lze už v něm filtrovat podle jména.

Další skripty jsou ty, které byly napsány pro potřeby aplikace autorem. Jsou to následující:

* beforeExit.js
  + Skript kontroluje, zda byla ve formuláři provedena změna a v případě opuštění formuláře bez jeho uložení, zobrazí informační zprávu o neuložených změnách.
  + Konkrétně:
    - Inicializace proměnné somethingChanged na hodnotu false.
    - Kontrola pomocí jQuery změna vstupních formulářů. Pokud někdo změnil vstup typu text nebo number, změní se hodnota proměnné somethingChanged na true.
    - V případě opuštění stránky (windows.onbeforeunload) se zkontroluje hodnota proměnné somethingChanged a podle toho se vypíše nebo ne upozorňující zpráva.
    - Pokud uživatel ale stiskne tlačítko pro uložení, tak se zpráva zobrazit nesmí, i když je hodnota naší proměnné na true.
    - Řešení spočívá v tom, že všechna tlačítka v aplikaci, která mají odeslat formulář, nemají nastaven typ na submit. Naopak ale mají své id nastaveno na „submit“. Stejně tak každý formulář má nastaveno své id na „form“. Skript pak kontroluje, jestli byl stisknut element, který má id „submit“. Pokud ano, nastaví naši proměnnou na false a formulář, ve kterém stisknutí nastalo, sám odešle.
    - Čtenáře může překvapit použití více submitů v kódu skriptu. To se týká formuláře pro vytvoření nového zboží. Formulář je rozdělen na tři záložky. Každá záložka má své vlastní tlačítko na odeslání formuláře. Tři tlačítka na jedné stránce ale nesmí mít stejné id, proto se využívá submitů 1, 2 a 3.
* deleteItem.js
  + Odstranění položky (příjem, zboží, položka příjmu) probíhá vždy stejně. Tlačítko pro odstranění zavolá metodu confirmation, která se nachází právě v tomto skriptu.
  + Metoda confirmation má tři parametry: id položky, text zprávy do dialogového okna a cestu k metodě, která odebrání provede.
    - Konkrétně:
      * Nejprve se inicializuje proměnná answer typu boolean. Představuje odpověď na vyskakovací okno. Vyskakovací okno se spouští zavoláním metody confirm(). Tato metoda nelze použít bez knihovny jQuery. Do metody confirm se vloží text zprávy.
      * Pokud je hodnota proměnné answer rovna true, pak se pomocí ajaxu zavolá metoda pro odebrání položky.
* loading.js
  + Jednoduchý skript pro zobrazení loadingu (otáčecí kolečko) při dlouhém načítání stránky.
* tabs.js
  + Tento skript slouží ke „skákání“ mezi záložkami ve formuláři pro vytvoření nového zboží.

Tohle byly JavaScripty, které se používají v aplikaci jako celé soubory. Aplikace ovšem obsahuje i JavaScripty, které jsou přímo psané do html kódu, protože třeba přímo používají C# kódu a nemohli tak mít vlastní skript.

Na konci souboru \_Layout.cshtml nalezneme kus kódu, který se stará o odhlášení uživatele po delší nečinnosti.

Soubory Create.cshtml a Edit.cshtml ve složce prijem\_pol obsahují metodu calculate(). Ta je volána při přepsání množství zboží v položce příjemky. Má za úkol přepočítat celkové množství na základě již zmíněného množství a koeficientů zboží.