BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

REMOTE API WEB REFERENCE FOR JAVA ENTERPRISE APPLICATIONS

NÁZEV PRÁCE

TERM PROJECT
SEMESTRÁLNÍ PROJEKT

AUTHOR — **Bc. ONDŘEJ KRPEC**
AUTOR PRÁCE

SUPERVISOR — **Ing. RADEK KOČÍ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2017

Abstract

This thesis focuses on modification of an existing tool JRAPIDoc that allows automatically generate an API documentation from an existing source code, but currently lacks good and simple presentation of the generated API. The goal of this thesis is to design an innovate user interface with regard to clarity and simple use, aimed to developers, even in case of large remote interfaces. Ultimately, the resulting tool will provide not only presentation of a generated API, but it will also allow its testing through calling web services that are provided by the API. The theoretical part describes the principles of web services, remote interfaces and examines existing tools for automatic API code generation. Last but not least the thesis describes the technologies that will be used for development of the tool and presents the tool's first designs. In conclusion, the results of this thesis allows the development of the JRAPIDoc improvements, which allow for decrease need of funds for the development of the documentation and for the realization of the system integration.

Abstrakt

Tato práce popisuje vylepšení existujícího nástroje JRAPIDoc, který umožňuje ze zdrojového kódu automaticky vygenerovat API dokumentaci. Tento nástroj ovšem postrádá jednoduchou a kvalitní prezentaci vygenerovaného API. Cílem této práce je návrh vylepšení, které umožní zmíněnému nástroji nejen jednoduché a propracované zobrazení automaticky vytvořeného API, ale také jeho testování pomocí volání webových služeb, které dané API poskytuje. Teoretická část práce popisuje principy webových služeb, vzdálených rozhraní a zkoumá již existující nástroje pro automatické generování API. Dále jsou zde popsány technologie, které budou použity při vývoji a také designové návrhy výsledného nástroje. Výsledky této práce umožní vývoj vylepšení nástroje JRAPIDoc, který bude moci poskytnout automatické generování snadno přístupného a testovatelného API, což umožní snížení nákladů potřebných pro tvorbu dokumentace a tím i realizaci systémové integrace.

Keywords

Remote API, Web Services, API documentation tool, System integration, REST, JRAPIDoc, Angular, PatternFly, Java Enterprise Applications, Red Hat

Klíčová slova

Vzdálená API, Webové služby, Nástroj na dokumentaci API, Systémová integrace, REST, JRAPIDoc, Angular, PatternFly, Red Hat

Reference

KRPEC, Ondřej. *Remote API Web Reference for Java Enterprise Applications.* Brno, 2017. Term project. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Radek Kočí, Ph.D.

Remote API Web Reference for Java Enterprise Applications

Declaration

Hereby I declare that this master's thesis was prepared as an original author's work under the supervision of Mr. Ing. Radek Kočí, Ph.D. The supplementary information was provided by Mr. Mrg. Ivo Bek. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....
Ondřej Krpec
October 15, 2017

Acknowledgements

I would like to thank Mr. Mgr. Ivo Bek for his technical leading of this Master Thesis. At the same time, I would like to thank Mr. Ing. Radek Kočí, Ph.D., for his pedagogical leadership.

Contents

Chapter 1

Introduction

In the software industry, the great and accessible code is crucial for modern business, and the best way for us to connect and share it is through APIs. But not only there hasn't been an industry standard for designing APIs, there hasn't been an industry standard for documenting them. APIs are supposed to connect engineers and allow for the sharing of great developments. APIs let companies add value to the products and create an ecosystem of shared knowledge. But an API is only valuable if it's accessible. And for that, it needs clear, human and machine readable documentation. So developers have worked hard to find a way to present their APIs.

This thesis aim to solve the problem of presenting APIs. The goal is to design an application that provides innovative user interface with regard to clarity and simple use, aimed to developers, even in case of large remote interfaces. The application will be based on existing JRAPIDoc [2] tool that provides a way to automate API code generation, but currently lacks its good represenetation. Ultimately, the iprovements should provide not only a remote API reference, but also the functionality to test and call the listed web services.

The thesis is organized as follows. Chapter 2 gives definitions needed to follow the thesis and provides overview of an existing API Reference generators. Chapter 3 focuses on technologies that are going to be used for building the improvements. Chapter 4 describes the iterative application design that is based on live mockups and later on PatternFly Framework. The last Chapter contains an overall summary of the designed solution and final thoughts on the work done within this thesis.

Chapter 2

Preliminaries And Definitions

This chapter will gradually introduce terms necessary to follow the thesis. In the first sections, we will provide explanation why is API documentation important and why it is important to automate API code generation. In the next section, we will go over the basic terminology and establish notion of remote interfaces. The last section covers the existing solutions for API code generation, their advantages and disadvantages.

2.1 What is API Documentation and Why It Matters?

In this section we explain what API is, we introduce the basic terms that relate to APIs and we focuses on why it is important to document APIs.

2.1.1 What is application programming interface?

In computer programming an application programming interface (API) is the defined interface through which interactions happen between an enterprise and users of its assets. An API can become the primary entry point for enterprise services, for its own website and applications, as well as for a partner and customer integrations. APIs are defined through a contract so that any application can use it with relative ease. An API approach is an architectural approach that resolves around providing programmable interfaces to a set of services to different applications serving different kinds of customers. It assumes that these different user groups might change or evolve over time in the way they utilize services. The API approach creates a loosely coupled architecture that allows a component service to have a wide range of future uses, and is technology agnostic. The strategy of providing APIs leads to the following benefits:

- **Automation**. With APIs, computers rather than people can manage the work. Through APIs, agencies can update work flows to make them quicker and more productive.
- **Integration**. APIs allow content to be embedded from any site or application more easily. This guarantees more fluid information delivery and an integrated user experience.
- **Personalization**. Through APIs any user or company can customize the content and services that they use the most.

- **Reduction of costs**. APIs is a cheaper way of building applications by increasing the reuse of services. Providing a usage or "analytics-based" evolutionary development platform decreases cost of development and change to services.

- **Increasing customer loyalty**. The company who releases the API allows its customers to access their conferencing services in new, more efficient ways, increasing brand recognition and customer loyalty.

Overall, APIs allow users to enhance and add services over existing products. This introduces the product to a new type of user: "the third-party" developer. Catering to the developer is tricky. They are analytical, precise, and are trying to solve important problems with your API. They want to understand how to use your API effectively, which is where API documentation comes into the picture.

2.1.2 What is API Documentation and Why We Need One?

API documentation is a technical content deliverable, containing instructions about how to effectively use and integrate with an API. It is a concise reference manual that contains all the information required to work with the API, with details about the functions, classes, return types and more, supported by tutorials and examples.

The "thid-party" developer, who is API's main consumer, is busy solving complex programming challenges. API is a means to an end for the technical user, and they want to integrate as quickly as possible to move forward in their software development, meaning they should immediately understand the value and usage of the API. The aggregate experience of the developer when discovering, learning to use, and finally integrating with the API is termed as Developer Experience (DX) [**?**]. Generally speaking, DX is essentially the sum of the experiences using the API. If many of those experiences were difficult or just annoying, users are going to have a bad experience and they won't want to use the product that the API provides. Therefore good API documentation is the key to a great DX. You can have the best, functional product, but no one will use it if they don't know how to. Documentation is the foundation for good Developer Experience. Following lists the exact reasons, why API documentation matters.

// TODO pridat poznamku pod carou, co to je network effect

- **Increased Awareness**. The network effect is the phenomenon when a service or product becomes more valuable when more people use it. Satisfied consumers will be the APIs biggest advocates. As more users adopt the API and reach critical mass, there will be a probable increase in publicity, leading to the network effect.

- **Easier maintenance**. Good documentation leads directly to good product maintenance. It helps internal teams know the details of resources, methods, and their associated requests and responses, making maintenance and updates quicker.

- **Saves time and costs**. Good documentation decreases the amount of time spent onboarding new users. In addition, if API provides the ability to try out the API before implementing it, the amount of time spent onboarding will decrease even more.

The point is that documentation is the key to a great experience when consuming the API.

2.1.3 Why automate API code generation?

Among all the phases in the API lifecycle, documentation is probably the area showing the most growth. However the developers pay little to no attention it when launching code. For a lot of developers it is because writing documentation is still tedious and time consuming task. This is especially true for teams that rely on static documentation that is manually updated with every release of new version of the API. Therefore the more we can automate around the documentation, the less work it is for the developers to iterate and make improvements.

That is why there is great need of a framework that allows developers and teams to design, build, document and consume their APIs with primary focus on the ability to generate interactive documentation. This documentation allows anyone to visualize and interact with the API's resources without having any implementation logic in place.

The "auto-generated" documentation is a central resource that the developers and teams can customize, and build on to create a more comprehensive user manual for working with the API.

2.2 Understanding Web Services

A web service is a software system designed to support interoperable machine to machine interaction over a network. A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to interprocess communication on a single computer. This interoperability is due to the use of open standards.

// TODO link na soap In the past, web services used mostly SOAP over HTTP protocol, allowing less costly interactions over the internet than via proprietary solutions like EDI/B2B. In a 2004, the W3C extended the definition of web services about REST-compliant web services, in which the primary purpose of the web service is to manipulate XML representations of web resources using a uniform set of stateless operations.

2.2.1 "SOAP-based" Web Services

SOAP is an acronym for Simple Object Access Protocol. It's a "XML based" messaging protocol for exchanging information among computers that uses WSDL TODO poznamka pod carou wsdl is an XML-based protocol that describes how to access a web service and what operation it will perform for communication between consumer and provider of the web service. Although SOAP can be used in a variety of messaging systems and can be delivered via a variety of transport protocols, the initial focus of SOAP is remote procedure calls transported via HTTP.

A SOAP message is an ordinary XML document containing the following elements:

- **Envelope**. Defines the start and the end of the message. Envelope is a mandatory element.
- **Header**. Contains any optional attributes of the message used in processing the message, either at an intermediary point or at the ultimate "end-point". Header is an optional element.

- **Body**. Contains the XML data comprising the message being sent. Body is a mandatory element.
- **Fault**. An optional Fault element that provides information about errors that occur while processing the message.

As shown in listing 2.1, a SOAP message consists of an Envelope element, inside which a Header and a Body element can be nested. Inside the Body element a Fault element can be nested, if an error occurs in the web service. This SOAP message format can be used both to send requests from the client to the web service, and to send responses back to the client from the web service. Thus the SOAP request and response message format is the same unlike the HTTP protocol where the request and response formats are different.

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
  <soap:Header>
    ...
  </soap:Header>
  <soap:Body>
    <!-- Fault element is optional,
         used only if a fault occurs in a web service -->
    <soap:Fault>
      ...
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Listing 2.1: Example of SOAP Message Structure.

When a client and a web service communicate via SOAP, they exchange messages. In the SOAP specification **??** are described two basic message exchange patterns.

- Request - Response
- Response

Request - Response pattern

In a request - response message exchange the SOAP client sends a SOAP request message to the web service. The service responds with a SOAP response message.

A request - response message exchange pattern is necessary when the SOAP client needs to send data to the SOAP service. For instance, if the client request that data be stored by the service, the client needs to send the data to be stored to the service.

// TODO obrazek example

Response pattern

In a response message exchange the SOAP client connects to the service, but does not send a SOAP message through. It just sends a simple HTTP request for instance. The service responds with a SOAP message. This pattern is similar to how browser communicates with a web server. The browser sends an HTTP request telling what page it wants to access, but the HTTP request does not carry any additional data. The web server responds with an HTTP response carrying the requested page.

// TODO obrazek example

2.2.2 RESTful Web Services

2.3 Existing solutions for API code generation

2.3.1 JRAPIDoc

2.3.2 Swagger

2.3.3 Apiary

2.3.4 Spring REST Docs

2.3.5 Postman

https://nordicapis.com/ultimate-guide-to-30-api-documentation-solutions/

Chapter 3

Technologic overview

3.1 Angular

3.2 TypeScript

3.3 PatternFly

Chapter 4

Application Design

4.1 The initial state

4.2 Building live mockups

4.3 Improving the mockups with PatternFly

Chapter 5

Conclusion

Bibliography

[1] Rábová, Z.; Hanáček, P.; Peringer, P.; et al.: *Užitečné rady pro psaní odborného textu.* FIT VUT v Brně. November 2008. [Online; navštíveno 12.05.2015].
Retrieved from: http://www.fit.vutbr.cz/info/statnice/psani_textu.html

[2] TODO: *Pravidla českého pravopisu.* Academia. 2005. ISBN 80-200-1327-X.

Appendix A

Jak pracovat s touto šablonou

V této kapitole je uveden popis jednotlivých částí šablony, po kterém následuje stručný návod, jak s touto šablonou pracovat.

Jedná se o přechodnou verzi šablony. Nová verze bude zveřejněna do konce roku 2017 a bude navíc obsahovat nové pokyny ke správnému využití šablony, závazné pokyny k vypracování bakalářských a diplomových prací (rekapitulace pokynů, které jsou dostupné na webu) a nezávazná doporučení od vybraných vedoucích, která již teď najdete na webu (viz odkazy v souboru s literaturou). Jediné soubory, které se v nové verzi změní, budou `projekt-01-kapitoly-chapters.tex` a `projekt-30-prilohy-appendices.tex`, jejichž obsah každý student vymaže a nahradí vlastním. Šablonu lze tedy bez problémů využít i v současné verzi.

Popis částí šablony

Po rozbalení šablony naleznete následující soubory a adresáře:

bib-styles Styly literatury (viz níže).

obrazky-figures Adresář pro Vaše obrázky. Nyní obsahuje placeholder.pdf (tzv. TODO obrázek, který lze použít jako pomůcku při tvorbě technické zprávy), který se s prací neodevzdává. Název adresáře je vhodné zkrátit, aby byl jen ve zvoleném jazyce.

template-fig Obrázky šablony (znak VUT).

fitthesis.cls Šablona (definice vzhledu).

Makefile Makefile pro překlad, počítání normostran, sbalení apod. (viz níže).

projekt-01-kapitoly-chapters.tex Soubor pro Váš text (obsah nahraďte).

projekt-20-literatura-bibliography.bib Seznam literatury (viz níže).

projekt-30-prilohy-appendices.tex Soubor pro přílohy (obsah nahraďte).

projekt.tex Hlavní soubor práce – definice formálních částí.

Výchozí styl literatury (czechiso) je od Ing. Martínka, přičemž anglická verze (englishiso) je jeho překladem s drobnými modifikacemi. Oproti normě jsou v něm určité

odlišnosti, ale na FIT je dlouhodobě akceptován. Alternativně můžete využít styl od Ing. Radima Loskota nebo od Ing. Radka Pyšného[1]. Alternativní styly obsahují určitá vylepšení, ale zatím nebyly řádně otestovány větším množstvím uživatelů. Lze je považovat za beta verze pro zájemce, kteří svoji práci chtějí mít dokonalou do detailů a neváhají si nastudovat detaily správného formátování citací, aby si mohli ověřit, že je vysázený výsledek v pořádku.

Makefile kromě překladu do PDF nabízí i další funkce:

- přejmenování souborů (viz níže),
- počítání normostran,
- spuštění vlny pro doplnění nezlomitelných mezer,
- sbalení výsledku pro odeslání vedoucímu ke kontrole (zkontrolujte, zda sbalí všechny Vámi přidané soubory, a případně doplňte).

Nezapomeňte, že vlna neřeší všechny nezlomitelné mezery. Vždy je třeba manuální kontrola, zda na konci řádku nezůstalo něco nevhodného – viz Internetová jazyková příručka[2].

Pozor na číslování stránek! Pokud má obsah 2 strany a na 2. jsou jen "Přílohy" a "Seznam příloh" (ale žádná příloha tam není), z nějakého důvodu se posune číslování stránek o 1 (obsah "nesedí"). Stejný efekt má, když je na 2. či 3. stránce obsahu jen "Literatura" a je možné, že tohoto problému lze dosáhnout i jinak. Řešení je několik (od úpravy obsahu, přes nastavení počítadla až po sofistikovanější metody). **Před odevzdáním proto vždy překontrolujte číslování stran!**

Doporučený postup práce se šablonou

1. **Zkontrolujte, zda máte aktuální verzi šablony.** Máte-li šablonu z předchozího roku, na stránkách fakulty již může být novější verze šablony s aktualizovanými informacemi, opravenými chybami apod.

2. **Zvolte si jazyk**, ve kterém budete psát svoji technickou zprávu (česky, slovensky nebo anglicky) a svoji volbu konzultujte s vedoucím práce (nebyla-li dohodnuta předem). Pokud Vámi zvoleným jazykem technické zprávy není čeština, nastavte příslušný parametr šablony v souboru projekt.tex (např.: `documentclass[english]{fitthesis}` a přeložte prohlášení a poděkování do angličtiny či slovenštiny.

3. **Přejmenujte soubory.** Po rozbalení je v šabloně soubor `projekt.tex`. Pokud jej přeložíte, vznikne PDF s technickou zprávou pojmenované `projekt.pdf`. Když vedoucímu více studentů pošle `projekt.pdf` ke kontrole, musí je pracně přejmenovávat. Proto je vždy vhodné tento soubor přejmenovat tak, aby obsahoval Váš login a (případně zkrácené) téma práce. Vyhněte se však použití mezer, diakritiky a speciálních znaků. Vhodný název může být např.: "`xlogin00-Cisteni-a-extrakce-textu.tex`". K přejmenování můžete využít i přiložený Makefile:

   ```
   make rename NAME=xlogin00-Cisteni-a-extrakce-textu
   ```

[1] BP Ing. Radka Pyšného http://www.fit.vutbr.cz/study/DP/BP.php?id=7848
[2] Internetová jazyková příručka http://prirucka.ujc.cas.cz/?id=880

4. Vyplňte požadované položky v souboru, který byl původně pojmenován `projekt.tex`, tedy typ, rok (odevzdání), název práce, svoje jméno, ústav (dle zadání), tituly a jméno vedoucího, abstrakt, klíčová slova a další formální náležitosti.

5. Rozšířený abstrakt v češtině lze v šabloně povolit odkomentováním příslušných 2 bloků v souboru `fitthesis.cls`.

6. Nahraďte obsah souborů s kapitolami práce, literaturou a přílohami obsahem svojí technické zprávy. Jednotlivé přílohy či kapitoly práce může být výhodné uložit do samostatných souborů – rozhodnete-li se pro toto řešení, je doporučeno zachovat konvenci pro názvy souborů, přičemž za číslem bude následovat název kapitoly.

7. Nepotřebujete-li přílohy, zakomentujte příslušnou část v `projekt.tex` a příslušný soubor vyprázdněte či smažte. Nesnažte se prosím vymyslet nějakou neúčelnou přílohu jen proto, aby daný soubor bylo čím naplnit. Vhodnou přílohou může být obsah přiloženého paměťového média.

8. Nascanované zadání uložte do souboru `zadani.pdf` a povolte jeho vložení do práce parametrem šablony v projekt.tex (`documentclass[zadani]{fitthesis}`).

9. Nechcete-li odkazy tisknout barevně (tedy červený obsah – bez konzultace s vedoucím nedoporučuji), budete pro tisk vytvářet druhé PDF s tím, že nastavíte parametr šablony pro tisk: (`documentclass[zadani,print]{fitthesis}`). Barevné logo se nesmí tisknout černobíle!

10. Vzor desek, do kterých bude práce vyvázána, si vygenerujte v informačním systému fakulty u zadání. Pro disertační práci lze zapnout parametrem v šabloně (více naleznete v souboru fitthesis.cls).

11. Nezapomeňte, že zdrojové soubory i (obě verze) PDF musíte odevzdat na CD či jiném médiu přiloženém k technické zprávě.

Obsah práce se generuje standardním příkazem `\tableofcontents` (zahrnut v šabloně). Přílohy jsou v něm uvedeny úmyslně.

Pokyny pro oboustranný tisk

- **Oboustranný tisk je doporučeno konzultovat s vedoucím práce.**
- Je-li práce tištěna oboustranně a její tloušťka je menší než tloušťka desek, nevypadá to dobře.
- Zapíná se parametrem šablony: `\documentclass[twoside]{fitthesis}`
- Po vytištění oboustranného listu zkontrolujte, zda je při prosvícení sazební obrazec na obou stranách na stejné pozici. Méně kvalitní tiskárny s duplexní jednotkou mají často posun o 1–3 mm. Toto může být u některých tiskáren řešitelné tak, že vytisknete nejprve liché stránky, pak je dáte do stejného zásobníku a vytisknete sudé.
- Za titulním listem, obsahem, literaturou, úvodním listem příloh, seznamem příloh a případnými dalšími seznamy je třeba nechat volnou stránku, aby následující část začínala na liché stránce (\cleardoublepage).
- Konečný výsledek je nutné pečlivě překontrolovat.

Styl odstavců

Odstavce se zarovnávají do bloku a pro jejich formátování existuje více metod. U papírové literatury je častá metoda s použitím odstavcové zarážky, kdy se u jednotlivých odstavců textu odsazuje první řádek odstavce asi o jeden až dva čtverčíky (vždy o stejnou, předem zvolenou hodnotu), tedy přibližně o dvě šířky velkého písmene M základního textu. Poslední řádek předchozího odstavce a první řádek následujícího odstavce se v takovém případě neoddělují svislou mezerou. Proklad mezi těmito řádky je stejný jako proklad mezi řádky uvnitř odstavce. [1] Další metodou je odsazení odstavců, které je časté u elektronické sazby textů. První řádek odstavce se při této metodě neodsazuje a mezi odstavce se vkládá vertikální mezera o velikosti 1/2 řádku. Obě metody lze v kvalifikační práci použít, nicméně často je vhodnější druhá z uvedených metod. Metody není vhodné kombinovat.

Jeden z výše uvedených způsobů je v šabloně nastaven jako výchozí, druhý můžete zvolit parametrem šablony "`odsaz`".

Užitečné nástroje

Následující seznam není výčtem všech využitelných nástrojů. Máte-li vyzkoušený osvědčený nástroj, neváhejte jej využít. Pokud však nevíte, který nástroj si zvolit, můžete zvážit některý z následujících:

MikTeX LaTeX pro Windows – distribuce s jednoduchou instalací a vynikající automatizací stahování balíčků.

TeXstudio Přenositelné opensource GUI pro LaTeX. Ctrl+klik umožňuje přepínat mezi zdrojovým textem a PDF. Má integrovanou kontrolu pravopisu, zvýraznění syntaxe apod. Pro jeho využití je nejprve potřeba nainstalovat MikTeX.

WinEdt Ve Windows je dobrá kombinace WinEdt + MiKTeX. WinEdt je GUI pro Windows, pro jehož využití je nejprve potřeba nainstalovat MikTeX či TeX Live.

Kile Editor pro desktopové prostředí KDE (Linux). Umožňuje živé zobrazení náhledu. Pro jeho využití je potřeba mít nainstalovaný TeX Live a Okular.

JabRef Pěkný a jednoduchý program v Javě pro správu souborů s bibliografií (literaturou). Není potřeba se nic učit – poskytuje jednoduché okno a formulář pro editaci položek.

InkScape Přenositelný opensource editor vektorové grafiky (SVG i PDF). Vynikající nástroj pro tvorbu obrázků do odborného textu. Jeho ovládnutí je obtížnější, ale výsledky stojí za to.

GIT Vynikající pro týmovou spolupráci na projektech, ale může výrazně pomoci i jednomu autorovi. Umožňuje jednoduché verzování, zálohování a přenášení mezi více počítači.

Overleaf Online nástroj pro LaTeX. Přímo zobrazuje náhled a umožňuje jednoduchou spolupráci (vedoucí může průběžně sledovat psaní práce), vyhledávání ve zdrojovém textu kliknutím do PDF, kontrolu pravopisu apod. Zdarma jej však lze využít pouze s určitými omezeními (někomu stačí na disertaci, jiný na ně může narazit i při psaní bakalářské práce) a pro dlouhé texty je pomalejší.

Pozn.: Overleaf nepoužívá Makefile v šabloně – aby překlad fungoval, je nutné kliknout pravým tlačítkem na `projekt.tex` a zvolit "Set as Main File".

Užitečné balíčky pro LaTeX

Studenti při sazbě textu často řeší stejné problémy. Některé z nich lze vyřešit následujícími balíčky pro LaTeX:

- `amsmath` – rozšířené možnosti sazby rovnic,
- `float, afterpage, placeins` – úprava umístění obrázků,
- `fancyvrb, alltt` – úpravy vlastností prostředí Verbatim,
- `makecell` – rozšíření možností tabulek,
- `pdflscape, rotating` – natočení stránky o 90 stupňů (pro obrázek či tabulku),
- `hyphenat` – úpravy dělení slov,
- `picture, epic, eepic` – přímé kreslení obrázků.

Některé balíčky jsou využity přímo v šabloně (v dolní části souboru fitthesis.cls). Nahlédnutí do jejich dokumentace může být rovněž užitečné.

Sloupec tabulky zarovnaný vlevo s pevnou šířkou je v šabloně definovaný "L" (používá se jako "p").