



**WIRTSCHAFTS
UNIVERSITÄT
WIEN** VIENNA
UNIVERSITY OF
ECONOMICS
AND BUSINESS

SEMINARARBEIT

Deutscher Titel der Seminararbeit

Groovy/PHP: Apache Tomcat Kochbuch (Nutshell Beispiele)

Englischer Titel der Seminararbeit:

Groovy/PHP: Apache Tomcat Cookbook (Nutshell Examples)

Verfasserin: Marion Tomes

Matrikel-Nr.: 11776345

Studienrichtung: Wirtschaftsinformatik

Kurs: 0100 Seminar aus BIS (Skiseminar)

Textsprache: Englisch

Betreuer: Prof. Mag. Dr. Rony G. Flatscher

Unternehmen: Wirtschaftsuniversität Wien

Declaration of Authorship

I assure:

- to have individually written, to not have used any other sources or tools than referenced and to not have used any other unauthorized tools for the writing of this seminar paper.
- to never have submitted this seminar paper topic to an advisor neither in this, nor in any foreign country.
- that this seminar paper matches the seminar paper reviewed by the advisor.

Date: December 17th, 2020

Signature Marion Tomes:

A handwritten signature in black ink, appearing to read 'Marion Tomes', is written on a light-colored background.

Abstract

Apache Tomcat evolves evermore to one of the most relevant open-source web servers worldwide. Therefore, the compatibility with various scripting languages, other than simply Java, is necessary and important to widen the possibilities of usage. The seminar paper assesses this issue by applying the scripting languages Apache Groovy and PHP in the web server using the BSF and JSR-223 tag libraries. After a brief theoretical introduction on the relevant topics, the appropriate installation to enable the use of both coding languages is elaborated. Further, simple nutshell examples are provided with the use of a fictive company to show the utilization of the scripts in Apache Tomcat in a realistic context. Throughout the nutshell examples the focus will also lie on the similarities and differences of Apache Groovy and PHP. To conclude, a short summary with the most relevant findings is provided as well as a future outlook focusing on possible further research possibilities in this topic field.

Table of Contents

1	Introduction.....	6
1.1	Objectives.....	6
1.2	Methodology.....	6
1.3	Structure.....	7
2	Background.....	8
2.1	Apache Tomcat.....	8
2.2	Java Server Pages (JSP).....	9
2.3	The Java Archive Tool (JAR).....	11
2.4	Apache Groovy.....	11
2.5	PHP.....	13
2.6	Tag Libraries (BSF & JSR-223).....	14
3	Installation.....	16
3.1	Apache Tomcat.....	16
3.2	Tag Libraries (BSF & JSR-223).....	18
3.3	Apache Groovy.....	18
3.4	PHP.....	20
4	Application of Nutshell examples.....	22
4.1	Nutshell context.....	22
4.2	Apache Groovy Examples.....	22
4.3	PHP Examples.....	28
5	Conclusion.....	35
5.1	Summary.....	35
5.2	Outlook.....	35
	Bibliography.....	37
	Appendix (A).....	40
	Appendix (B).....	44
	Appendix (C).....	48

List of Figures

Figure 1. Viewing a plain HTML page (Zambon, 2012).....	10
Figure 2. Viewing a JSP page (Zambon, 2012).....	11
Figure 3. Apache Tomcat Folder Structure.....	16
Figure 4. Apache Tomcat starting page.....	17
Figure 5. Tomcat Manager Role.....	17
Figure 6. BSF Tag Library Header.....	18
Figure 7. JSP-223 Tag Library Header.....	18
Figure 8. /demo Directory.....	19
Figure 9. Index of demo.war.....	20
Figure 10. Apache Tomcat Lib Directory.....	21
Figure 11. bsf.jsp.....	23
Figure 12. Groovy „Hello World“.....	23
Figure 13. Result Groovy „Hello World“.....	23
Figure 14. Groovy If-Else-If Statement.....	24
Figure 15. Result Groovy If-Else-If Statement.....	25
Figure 16. Groovy Switch Statement.....	25
Figure 17. Result Groovy Switch Statement.....	26
Figure 18. Groovy List.....	26
Figure 19. Possible result Groovy List.....	27
Figure 20. Groovy Map.....	27
Figure 21. Result Groovy Map.....	28
Figure 22. jsr223.jsp.....	28
Figure 23. PHP „Hello World“.....	29
Figure 24. Result PHP „Hello World“.....	29
Figure 25. PHP If-Else-If Statement.....	30
Figure 26. Result PHP If-Else-If Statement.....	30
Figure 27. PHP Switch Statement.....	31
Figure 28. Result PHP Switch Statement.....	31
Figure 29. PHP Sessions.....	32
Figure 30. Result PHP Session.....	33
Figure 31. PHP Cookies.....	34
Figure 32. Result PHP Cookie.....	34

1 Introduction

This seminar paper focuses on the implementation of Apache Groovy and PHP code in the Apache Tomcat web server. This approach is achieved through the use of the BSF and JSR-223 tag libraries, that enable the involvement of scripting languages, other than Java, in JSP scripts. The theoretical background covers the relevant topics and gives a more in-depth insight on the concepts of the Bean Scripting Framework (BSF) as well as the Scripting for the Java Platform (JSR-223) in combination with the corresponding tag libraries. Moreover, the seminar paper discusses similarities and differences of Apache Groovy and PHP with the use of simple nutshell examples. These examples are applied with the use of a fictive company to enable a more practical understanding of the coding instances. For Apache Groovy a simple Hello World application is provided as well as nutshell examples for an If-Else-If as well as a Switch statement. Following, two important features, lists and maps, are introduced and relevant similarities as well as differences outlined. Next, the PHP examples cover also a Hello World application, as well as an If-Else-If and Switch statement with comparison to the examples introduced in the Apache Groovy section. Lastly, sessions as well as cookies are introduced in the context of PHP scripting and nutshell examples are provided. To conclude, a brief summary with focus on the findings of this seminar paper is given as well as an outlook for future or further work is provided.

1.1 Objectives

The objectives of this work are the research of a possible implementation of the scripting languages Apache Groovy and PHP into the Tomcat web server using Java Server Pages and consequently present them as nutshell examples. Furthermore, the goal is to achieve a development environment where these nutshell examples can be applied and tested properly. Additionally, the aim is to provide a future outlook for further research in this area.

1.2 Methodology

The seminar paper consists of a first literature review that provides the theoretical fundament for the further work. Additionally, a practical installation guide is provided as

well as short nutshell examples consisting of code pieces as well as pictures of the respective outcomes. The nutshell examples provide an additional theoretical insight in certain topics through further in-depth literature.

1.3 Structure

The seminar paper is structured into four sections. First, the theoretical section covering all topics relevant for the further practical sections. Second, an installation guideline for the development environment used in the third section. Third, the practical section focuses on providing practical and realistic nutshell examples based on the knowledge gained in section one. Last, section four concludes the seminar paper by discussing the findings and providing an outlook for further research.

2 Background

This chapter covers the theoretical background for this seminar paper. First, the web server Apache Tomcat will be explained together with a description of the Java Server Pages in the context of Apache Tomcat. Second, the scripting languages Apache Groovy and PHP, that are necessary for the further chapters, are covered together with the Java Archive tool. Lastly, an introduction for the tag libraries will be given, that is used for the implementation of the nutshell examples in chapter 4.

2.1 Apache Tomcat

Apache Tomcat is a mature web server development from over twenty years ago. It has since then be maintained by the open-source community and has gained a great community support throughout the years (Davis, 2014). “The Apache Tomcat software is an open source implementation of the Java Servlet, Java Server Pages, Java Expression Language and Java WebSocket technologies (The Apache Software Foundation, 2020b).” The web server and servlet container are open source and powers numerous web applications across several ranges of organizations. It is commonly used for applications that are developed using Java Server Pages (JSP) files and servlets (Agrawal & Gupta, 2014).

For the purpose of this seminar paper the version 9.0.36 of Apache Tomcat is used.

The following sub-chapters will cover the most convincing factors and advantages of Apache Tomcat, that have a high relevancy for this seminar paper.

2.1.1 Lightweight Application

There are several reasons to decide for an Apache Tomcat web server, a main advantage is the easy deployment and the fast-loading times that can be achieved in comparison to competitors. Additionally, the functions are only basic to run the server, but therefore enable a lightweight application (Davis, 2014).

2.1.2 Open-Source

Besides being a lightweight application, the web server is open-source and is available to anyone who decides for a download. This also enables a high degree of freedom for developers to interact and change the Tomcat application according to their needs (Davis, 2014).

2.1.3 Stability

Another reason to choose Tomcat is the server stability, that is very high because the web server runs independent of the Apache installation. Therefore, any error that would cause a disfunction of the Tomcat server, would not cause any damage and the other servers continue their service normally (Davis, 2014).

2.2 Java Server Pages (JSP)

To add dynamical content to web pages, the JSP technology is the way to go. While strict HTML pages need to be adjusted by hand to update any content, JSP can add dynamic content based on various factor, e.g., user history or the browser type (Zambon, 2012). This content is enabled through script tags commonly beginning with `<%` and ending with `%>` (Leitenmüller, 2001). Following, a short explanation will cover the standard process when viewing a web page, to go into more detail on why JSP is needed in this context (Zambon, 2012).

2.2.1 Viewing a HTML page

Before going deeper into JSP, it is necessary to have clear knowledge on what happens when a browser is asked to view a web page. The progress to view a static web page can be seen in Figure 1 and could be initiated by either typing in an URL or clicking on a hyperlink. First, the URL that was entered is resolved by the browser into the respective Internet Protocol address, by asking the DNS server about the address. Second, the browser consequently sends a HTTP request to the IP address and gets returned the content of the requested file. Continuing, the web server sends an HTTP response that consists of the plain-text HTML page. Lastly, the browser conceives the response, interprets the HTML and displays also non-textual content (Zambon, 2012).

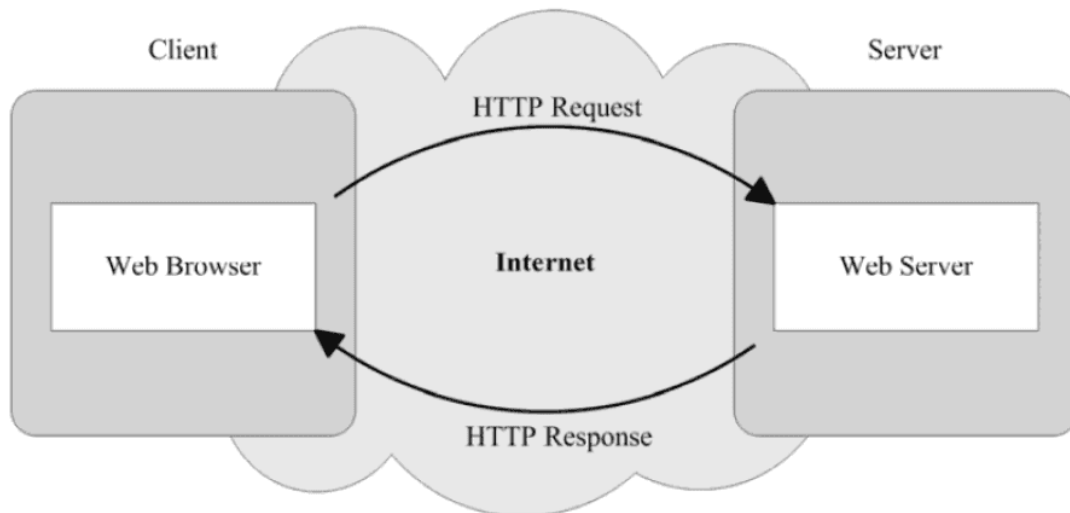


Figure 1. Viewing a plain HTML page (Zambon, 2012)

In this context Java Server Pages is a technology that supports the creation of dynamically rather than static generated pages. This happens by dynamically converting script files into Java modules. This code can then be executed by Apache Tomcat, which acts as the web server (Zambon, 2012). The process in detail is covered in the following chapter.

2.2.2 Viewing a JSP page

After introducing the standard process of viewing a simple HTML page, it is now necessary to provide some information on how it is done with JSP. Figure 2 shows the steps necessary for the web server to create a web page with JSP. First, the browser sends an HTTP request to the web server, same as for the static HTML page. Now, the server is a Java server that is able to handle Java servlets and forwards the corresponding JSP files to a JSP engine. The JSP engine is reliable for the download of the JSP page and the conversion to a Java servlet. Next, the Servlet engine compiles the servlet and sends the original request to the servlet engine that loads the servlet class and executes it. Throughout the execution, a page in an HTML format is produced by the servlet and finally forwarded to the web server with an HTTP response. This response is then passed to the browser with an HTTP response. Lastly, the web browser receives the page in form of a dynamical generated HTML page and displays it as if it were a static page like in the previous chapter (Zambon, 2012).

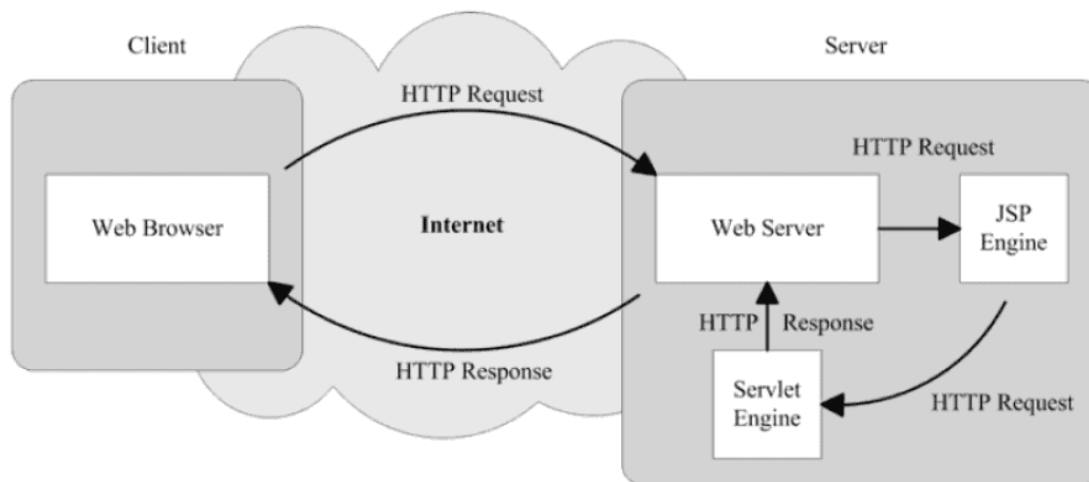


Figure 2. Viewing a JSP page (Zambon, 2012)

2.3 The Java Archive Tool (JAR)

Generally speaking, a single JAR archive file consists of multiple files. The main focus of JAR files is the packaging of java applets or applications in a simple compressed archive and therefore improving the download time. This leads to the fact that it can be downloaded in a single HTTP transaction and there is not necessity to establish a new connection for each piece. An advantage of using JAR files is the possibility to sign entries by the applet author and therefore ensure authentication (Oracle, 2020b). For the purpose of this seminar paper, five JAR files are needed and are further described in Chapter 3.4.

2.4 Apache Groovy

The scripting language Apache Groovy is an agile and dynamic language for the Java Virtual Machine. It is compatible with all existing Java classes and libraries but has additional features that were inspired by languages like Python or Ruby. Additionally, the learning effort for Java developers is very low, but enables them to use modern programming features. Besides that, Groovy can be used in every context where Java bytecode can be applied and increases productivity by reducing code effort for developers (The Apache Groovy Project, 2020d). Following, some of the main differences to Java as well as the most important frameworks using Apache Groovy are described.

2.4.1 Differences to Java

As already introduced, Groovy has a low learning effort for Java developers, as it tries to stay as natural as possible and follows the principles of Java. Still, a view differences exist and outline the advantages of Groovy in comparison to Java. First, some of the most frequently used classes are already imported and do not need to be included in the script by an explicit import statement. Second, the methods that are chosen are invoked at runtime, meaning that the method is chosen based on the arguments at runtime. Java, in comparison, works at compile time and methods are therefore chosen at compile time based on the declared types. The last difference to be outlined in this context, because there consist numerous more, are the existing extra keywords. Through this, it is possible to code even faster by using keyword such as 'def', 'as' or 'in' that are not defined in Java and condense more code-intensive variants of Java (The Apache Groovy Project, 2020b).

2.4.2 Related Frameworks

Numerous projects such as frameworks, desktop application frameworks, testing frameworks and even more evolved through Apache Groovy. A number of successful projects are going to be mentioned, that leverage Groovy at their core (The Apache Groovy Project, 2020c).

The first, and probably most successful, project is Grails which is an open-source full stack web application framework for the Java Virtual Machine. It uses Groovy to provide a productive and stream-lined development. For the approach of this seminar paper, grails would have also provided a possibility to enable Groovy in Apache Tomcat (The Apache Groovy Project, 2020c).

Another successful build automation framework based on Groovy is Gradle that enables automatic testing, building and publishing of software packages or projects (The Apache Groovy Project, 2020c).

Another approach that has been developed with Groovy is Ratpack, a simple and capable toolkit for the construction of high performing web applications (The Apache Groovy Project, 2020c).

2.5 PHP

PHP is a scripting language made to be integrated in HTML mainly for web scripting purposes. PHP simplifies scripting with HTML by enabling the possibility to embed instructions that display the relevant context. The scripting language is acting on the server-side and therefore means that the code is processed on the server where a HTML page is created and send to the client. The benefit lies in the code privacy as there is no possibility for the client to access the code and only view the content that is displayed by the server. PHP code is placed between special instructions in the beginning `<?php%` and in the end `%?>` (The PHP Group, 2020e). The following sub chapters focus on possible application areas of PHP and the database compatibility.

2.5.1 Application Areas

PHP can be used with all common operating systems and its major focus lies on the server-side scripting. The scripting language is not only restricted to be a web framework, there are several other areas where PHP is applied.

First, as already stated, PHP is suitable for server-side scripting. It is the most used and targeted field of the scripting language, because it enables the developer to access PHP output with a web browser and view the context through the server. A main advantage is, that PHP can be used by beginners as well as experts and can be used from local machines to wide firm networks (The PHP Group, 2020b).

Second, PHP enables command line scripting without the use of server or browser. This can be done easily with the PHP parser and complements scripts using cron (Linux) or the Task Scheduler (Windows) (The PHP Group, 2012).

Lastly, the scripting language also enables the development of desktop applications. Obviously, it is not the primary source for this application area, but for experienced PHP users it is possible to create a desktop application with a graphical interface using the PHP-GTK extension of PHP. By using this extension, it is also possible to create cross-platform applications (The PHP Group, 2016).

2.5.2 Database compatibility

A main feature of PHP, as a server-side scripting language, is the support of a wide range of databases. It enables a simple possibility to use database data and extensions

for web pages. Some examples for compatible data bases are MySQL, PostgreSQL and MongoDB.

2.6 Tag Libraries (BSF & JSR-223)

As JSP enables developers to extend their functionality using the tag extensions, it is possible to create tag extension using JSP syntax or tag libraries that extend JSP pages with a tag handler. These tag libraries consists of the following components (Oracle Corporation, 2020):

- One or more handler classes
- Descriptor (TLD)
- XML document with metadata about tags

With the use of the tag libraries it is possible to simply interpret different scripting languages used in a JSP page (Oracle Corporation, 2020). This seminar paper uses the BSF and JSR-223 tag libraries that will be elaborated further. This tag libraries are able to process scripts using BSF or JSR-223. Those frameworks as well as the descriptor are covered in the following sub-chapters.

2.6.1 Tag Library Descriptors (TLD)

If tag files should be redistributed or customer tags should be implemented with tag handlers written in Java, it is necessary to declare these tags in a tag library descriptor (TLD). The file is an XML like document consisting of information about the library and the tags that are contained in the library. It is needed to include TLD, because it is used by the JSP page development tools and a web container in order to validate the tags (Oracle, 2010).

2.6.2 Bean Scripting Framework (BSF)

BSF has been developed in 1999 by IBM and is an open-source interface for scripting languages within Java applications. By using BSF it is possible for scripting languages to access Java objects and methods. Therefore, it is possible to write JSP files with other languages than Java while enabling access to the Java class library. At the moment two versions of BSF are existing. On the one hand, the 2.x releases still take the originally

developed API from IBM. On the other hand, the 3.x releases are the new version of Apache BSF and use the API that was defined as part of JSR-223 (javax.script) (The Apache Software Foundation, 2011). Important to mention is, that Apache BSF itself does not consist of language engines. They have to be downloaded independently, as for many languages separate factories do already exist. This leads to the fact that having all factories in one JAR, some major problems could arise at run-time. The reason is that if other JARs include factories that apply another version of the same language, problems are caused when trying to choose the loaded version (The Apache Software Foundation, 2011).

2.6.3 Scripting for the Java Platform (JSR-223)

JSR-223 first introduced a pluggable and scripting language-independent architecture. It was inspired by the BSF and was the base for the web scripting framework. The first approach of JSR-223 was the definition of a standard and portable way for the allowance to generate web content of programs written in scripting languages for web content. To do so, there is a necessity of having a common set of interfaces for programming used for the execution of scripts in script engines. Therefore, it was necessary to widen the specification and the specification then also included a standardized scripting API, similar to the BSF. In this API the elements of the framework for the web scripting are defined. The goals of the scripting API were the portability and backward-compatibility. A main goal was to enable developers to use different code paths to handle various behaviors at runtime. Important was, that the interface resemble the established ones to a high extent, enabling an easier adoption of the existing interfaces. Another difference to the previously introduced BSF is, that JSR-223 does not distinguish if the language is hybrid or interpreted. Also, it is a framework that enables Java applications the possibility to host script engines (Oracle Corporation, 2020).

3 Installation

The theoretical fundament has now been covered; hence it is necessary to enable a development environment to enable the application of nutshell examples. Therefore, the following sub-chapters will consist of the installation guidelines for the web server Tomcat as well as the implementation instructions for Groovy, PHP and the tag libraries. The set-up was made using MacOS Catalina version 10.15.7.

3.1 Apache Tomcat

Starting, the web server Apache Tomcat needs to be installed in order to continue with the following applications. For this purpose, the zip-package containing all relevant files needs to be downloaded from the official Apache Tomcat website (The Apache Software Foundation, 2020a). For this seminar paper the folder 'apache-tomcat-9.0.39' was then further edited in the IntelliJ IDEA environment. To do so, the Tomcat file needs to be opened as a new project and if the installation was successful, the directory should look similar to Figure 3.

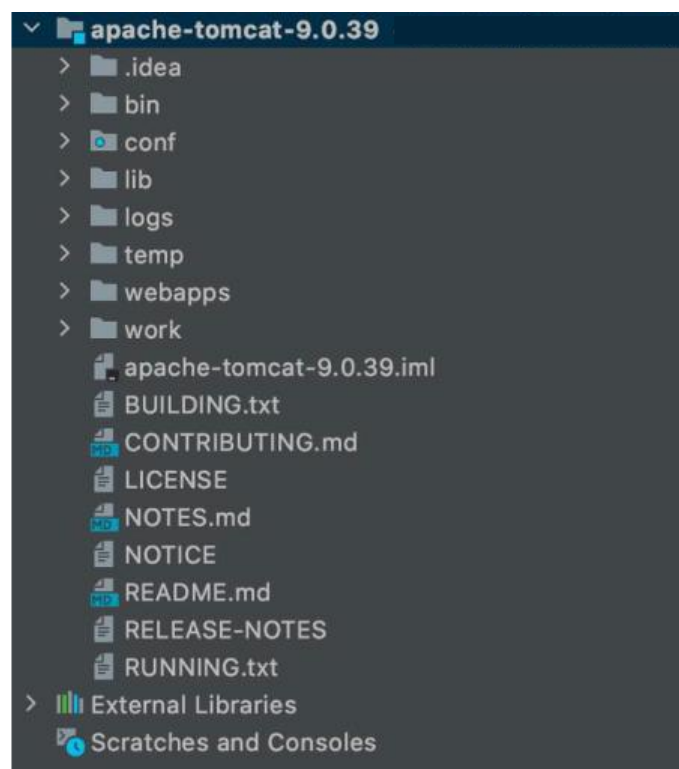


Figure 3. Apache Tomcat Folder Structure

The web server can from there on be started with the command ‘catalina start’ in the terminal. If the installation was successful, the terminal will state ‘Connected to server’ and the success of the installation can be review by directing to localhost:8080 or 127.0.0.1:8080 where the initial starting page of Apache Tomcat should be displayed like in Figure 4.

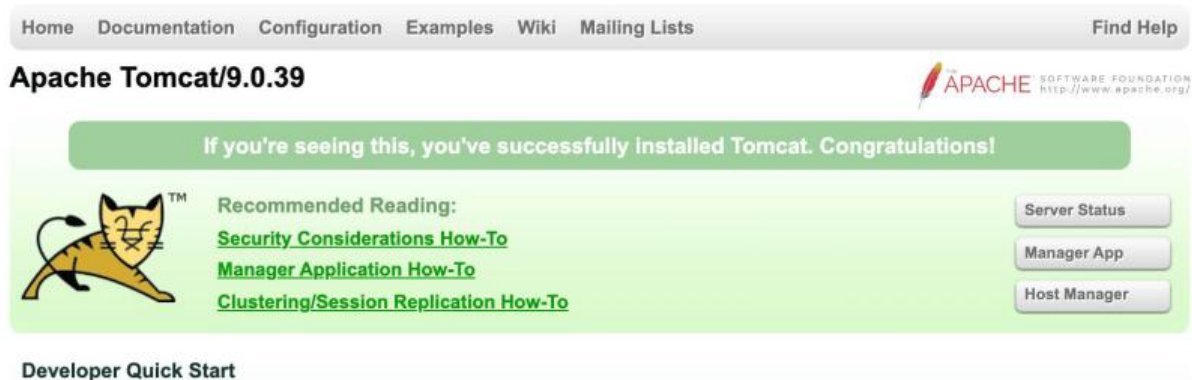


Figure 4. Apache Tomcat starting page

The next step to configure Tomcat is to set the manager roles to get access to the manager app where the applications can be configured and also started, stopped or restarted. The roles need to be set in conf/tomcat-users.xml between the <tomcat-user> tags and an example can be seen in Figure 5. To review if the settings where successful, the web server needs to be restarted using the command ‘catalina stop && catalina start’ in the terminal. After the web server is connected again, the configuration can be verified by browsing to localhost:8080/manager and inserting the chosen username and password. If the installation was successful, the manager app is visible, and the default applications are available and can be navigated to. To get to know the usage of the applications the /examples path shows some instances especially for JSP Examples. The corresponding scripts can be found in webapps/examples/jsp.

```
<user username="admin" password="secret" roles="standard,admin-gui,manager-gui"/>
```

Figure 5. Tomcat Manager Role

The starting process can also be configured in IntelliJ IDEA to make it easier. First, it is necessary to browse in the toolbar to Run > Edit Configurations, click on add and pick ‘Tomcat Server’ and ‘Local’. Second, the default Run/Debug Configuration can be

accepted, but make sure that the right application server (in this case 'Tomcat 9.0.39') is set. By clicking on 'Configure' the additional settings are available at the top right bar of IntelliJ IDEA and can be used to handle the Tomcat web server without the need of commands in the Terminal.

3.2 Tag Libraries (BSF & JSR-223)

As the Tomcat web server has now successfully been installed, the next step is to integrate the tag libraries. The libraries can be downloaded from sourceforge.net (Slashdot Media, 2020) and contain three files that are necessary for the following installation steps. The necessary files are named javax.ScriptTagLibs.jar, script-jsr223.tld and script-bsf.tld. After the files have been downloaded the javax.ScriptTagLibs.jar needs to be added to the lib directory in the Apache Tomcat installation. This JAR file consists of the tag libraries and uses the *javax* namespace. For Tomcat 10, which is to the date of the seminar paper only released with a beta version, and later it is necessary to use jakarta.ScriptTagLibs.jar instead of javax.ScriptTagLibs.jar. The other two TLD files need to be added to the demo application and will be covered in chapter 3.3. To use the tag library for BSF in a JSP file, the code from Figure 6 needs to be added on top of the script. If JSP-223 should be used, add the code from Figure 7 to the top of the script.

```
<%@ page session="false" pageEncoding="UTF-8"  
contentType="text/html; charset=UTF-8" %>  
<%@ taglib uri="/WEB-INF/script-bsf.tld" prefix="s" %>
```

Figure 6. BSF Tag Library Header

```
<%@ page session="false" pageEncoding="UTF-8"  
contentType="text/html; charset=UTF-8" %>  
<%@ taglib uri="/WEB-INF/script-jsr223.tld" prefix="s" %>
```

Figure 7. JSP-223 Tag Library Header

3.3 Apache Groovy

To use Groovy as a scripting language with the tag libraries, some configurations need to be done. To start the implementation, the demo.war file from Gitlab (Tomes, 2020) can be downloaded and used as a starting point. The Web Archive Format (WAR) is used to distribute among other servlets, JSPs, tag libraries, JAR files, static web pages and many more packaged in a single WAR file somehow similar to ZIP files (Niemeyer &

Leuck, 2013). The WAR file in this case consists of the nutshell examples that will be handled in Chapter 4 and can be used as a starting point for the own installation and testing of the functionality. The file needs to be put in Apache Tomcats /webapps directory. After inserting the war file there, the process is started automatically, and the demo directory should be available a few seconds after putting the war file into the webapps directory. Of course, it is also possible to start with the files in webapps/examples for first tests or take the JSP files from Appendix (A) and Appendix (B). Next, the files script-jsr223.tld and script-bsf.tld need to be added to the webapps/demo/WEB-INF directory. The TLD file for JSR-223 consists of the description of the JSR-223 tag library. Similarly, the TLD file for BSF contains a description for the BSF tag library (Slashdot Media, 2020). After successfully adding the war file, the directory should look similar to Figure 8.

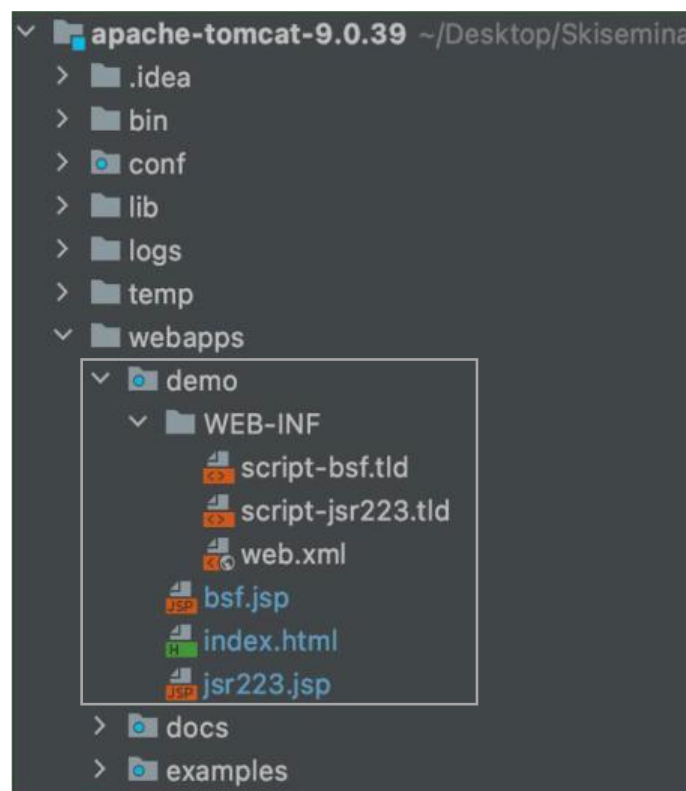


Figure 8. /demo Directory

To check the success of the implementation, it is necessary to go to localhost:8080/manager and choose /demo in the context path. If the front page looks similar to the one from Figure 9, the installation was successful. The corresponding code for the index page can be found in Appendix (.

Apache Tomcat Nutshell Examples (Groovy, PHP)

The following tag libraries (taglibs) can be applied for the same purpose:

- `script-bsf.tld`: Apache BSF (Bean Scripting Framework)
This is the original Java scripting framework that can be used for Groovy.
Important: Groovy is also supported by the newer scripting framework (see below).
- `script-jsr223.tld`: JSR-223 (Java's javax.script Framework)
This is the Java scripting framework that got introduced with Java 6 and works with Groovy and PHP.

Example Files

- [BSF Examples](#) (using Groovy via BSF)
- [JSR-223 Examples](#) (using Groovy and PHP via JSR-223)

The example files are JSP (Java server page) files that are usually created with HTML or xhtml and have usually Java "Servlet" code injected. When creating the HTML page the code gets executed and usually creates part of the resulting HTML file.

The scripting tag libraries enable the user to use any Java script language instead or in addition to Java using the Apache BSF framework or the Scripting for the java platform (JSR-223) framework.

The log files can be found in the Tomcat log directory.

Figure 9. Index of demo.war

Afterwards, it is necessary to download the suitable JAR files to connect groovy and the tag libraries. To do so, the following files need to be downloaded from the Maven directory (The Apache Groovy Project, 2020a):

- `groovy-3.0.6.jar`
- `groovy-bsf-3.0.6.jar`
- `groovy-jsr223-3.0.6.jar`

After the successful download, the files need to be located to the `/lib` directory of the tomcat web server.

3.4 PHP

Lastly, the installation of PHP is necessary to use it as a scripting language in the JSR-223 tag library. Contrary to the scripting language Groovy, PHP is not a Java-based language and therefore needs a different installation. The initial PHP installation is not necessary on a Mac, as it is already preinstalled by default. To enable a connection of PHP and the JSR-223 tag library it is necessary to install a Java Implementation that allows an integration of Java services with PHP scripts. For this purpose, the most suitable approach was Quercus and the WAR file can be downloaded from the Quercus official website (Caucho Technology, 2014). The WAR file can be deployed in the `/webapps` directory and will be unpacked after a few seconds of waiting. Afterwards take `quercus.jar` from `webapps/quercus-4.0.39/WEB-INF/lib` and copy it to tomcats `/lib` directory. The `/lib` directory with all necessary JAR files for this seminar paper can be seen in Figure 10.

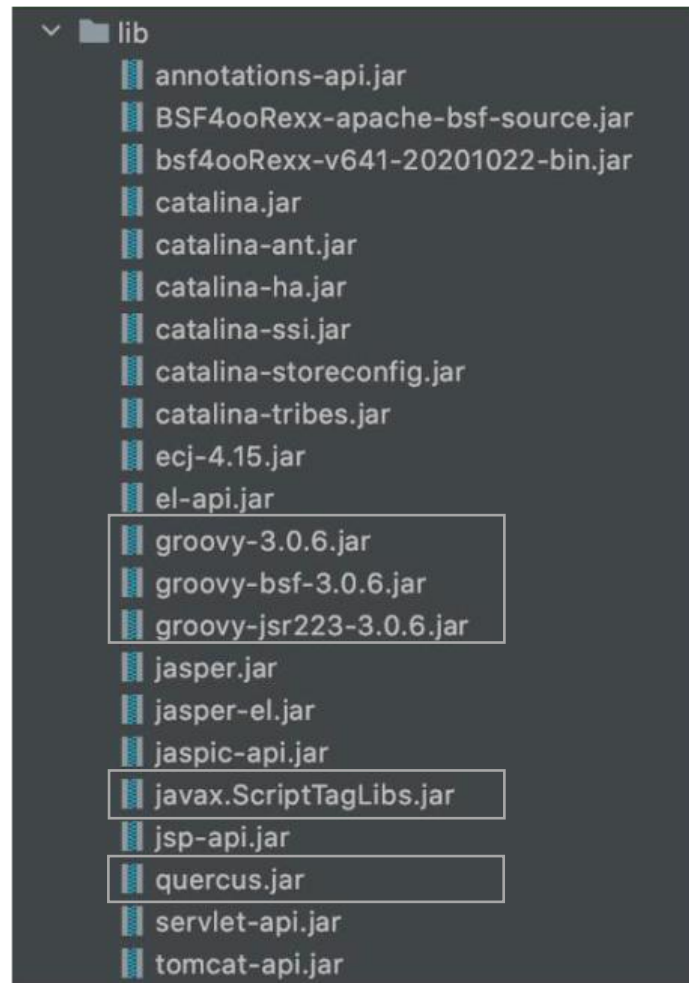


Figure 10. Apache Tomcat Lib Directory

The installation is now finished, and the success can be verified by applying PHP code in a JSP file or by clicking on the example files in the demo folder. The demo folder consists of two files, one for BSF scripting, consisting of Groovy examples, and one for JSR-223 consisting of PHP examples. The PHP implementation only works with JSR-223. On the contrary, it is possible to include the Groovy code from the BSF examples also in the JSR-223 file.

4 Application of Nutshell examples

In this section the previously gained knowledge will be visualized with code examples. The examples for Groovy and PHP will contain code pieces that show some basic functionalities of the respective scripting language. To bring the examples into context, a fictive company will be used in order to elaborate on the nutshell examples and their functionality in a possible real-life context. The nutshell examples compare on the one hand the possibilities for the same features of both languages and present on the other hand advantages of specific functions that are achieved by using the respective coding language.

4.1 Nutshell context

For the following nutshell examples a fictive company is used to give the provided code pieces a possible real-life context. The company used is an online B2B news publisher focusing especially on five main business topics. These topics cover IT, Marketing, Finance, Economics and Mathematics. The company's focus lies on the personalization of the news web page for registered customers and guest users.

4.2 Apache Groovy Examples

The principles and application of Apache Groovy has already been discussed in Chapter 3.3 and will now be elaborated further by providing nutshell examples. The nutshell examples were tested and developed in the environment that was set up throughout Chapter 3. To see the output of the respective nutshell examples, browse to the 'BSF Examples' on the index page of the demo installation (see Chapter 3.3). The nutshell examples are stored in `webapps/demo/bsf.jsp` and should look like Figure 11. It is also possible to view the respective code on the page by clicking on the topic or the triangle.

```

BSF
▶ Hello World
Hello World from Groovy!

▶ If-Else-If
You can find the latest news from your field of interest here!

▶ Switch
See the latest news on IT

▶ Lists
[Finance - 3, Mathematics - 1, Mathematics - 4, Finance - 5, Mathematics -
2]

▶ Maps
user123 = Economics, user234 = Marketing, user657 = Economics, user986 = IT,

```

Figure 11. bsf.jsp

4.2.1 Hello World

Now, as the configuration for Groovy has been finished, the BSF configuration can be tested in webapps/demo/bsf.jsp by adding the code from Figure 12. The configuration can also be tested with JSR-223 in webapps/demo/jsp223.jsp by using the same code as for the BSF example but with another header. After the implementation the success can be checked on the applications website for the specific JSP files. If the text from Figure 13 is displayed, the installation was successful and groovy can be used in the Script-Tag of the BSF or JSR-223 tag library.

```

<s:script type="groovy" cacheSrc="false">
out.println("<p>Hello World from Groovy!</p>")
</s:script>

```

Figure 12. Groovy „Hello World“

```

Hello World from Groovy!

```

Figure 13. Result Groovy „Hello World“

4.2.2 Decision Making

Nearly every scripting language offers the possibility to have decision making statements. These statements enable the developer to show or proceed with certain actions if a condition is true. Following two of the most common decision-making statements will be covered and two nutshell examples will be elaborated that show the application of the code in the fictive company.

4.2.2.1 If-Else-If Statement

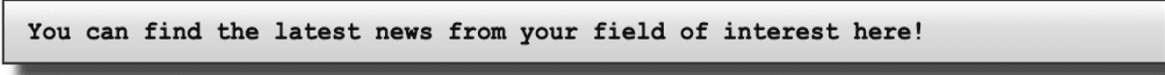
The If-Else-If Statement is an extension of the standard If-Statement, where it is possible to make decisions based on conditions throughout the code. By using the If-Else-If statement it is possible to have a more complex condition check. In the example below, three states of the user were defined and possible outcomes for these conditions designed. The states cover three possible user status namely user, premium and guest (default). The aim in this case is to have a message displayed on the front page of their website specifically for the respective user group. In the second line the user status is defined using 'def', in the case of the fictive company the data would be stored and gained from e.g., a database. To ease the process this state was already pre-defined with the user status "user".

```
<s:script type="groovy" cacheSrc="false">
  def userstatus = 'user'
  if (userstatus='user') {
    out.println("You can find the latest news from your field of
interest here!");
  } else
  if (userstatus = 'premium') {
    out.println("View your personalized news section!");
  } else {
    out.println("Have a look at the latest news!");
  }
</s:script>
```

Figure 14. Groovy If-Else-If Statement

If the script is started, the output should show the introduction headline for the user status 'user' and the outcome should look like in Figure 15. If the user status is null, the company suggest having a guest on their web site and therefore displays a generic message, whereas the premium user receives a completely personalized front page and

news section. By changing the value in line 2 to 'premium' or any other description except from 'user', it is possible to receive the other messages for the premium or guest users.



You can find the latest news from your field of interest here!

Figure 15. Result Groovy If-Else-If Statement

4.2.2.2 Switch Statement

Another possibility to personalize the user experience is to show personalized news articles for the registered users. Therefore, a Switch statement is made where the topic preference of the user is taken from e.g., a database. In this case the preference was set to IT for the simplicity of the nutshell example. The Switch Statement is divided into different cases that have individual website headlines according to the preference of the user. If no preference is stored for the respective user, the default sentence is shown.

```
<s:script type="groovy" cacheSrc="false">
  def preference = 'IT'
  switch(preferance) {
  case 'IT':
    out.println("See the latest IT news");
    break;
  case 'Marketing':
    out.println("See the latest marketing news");
    break;
  case 'Finance':
    out.println("See the latest finance news!");
    break;
  default:
    out.println("You can see some articles that you may find
interesting");
    break;
  }
</s:script>
```

Figure 16. Groovy Switch Statement

By starting the script, the headline looks like the output in Figure 17. As the user prefers to be informed about IT news, the website could suggest articles focusing primarily on the the latest IT news articles.

See the latest IT news

Figure 17. Result Groovy Switch Statement

4.2.3 Collections

Collections can often be found as fundamental components of programs (reality rehab, n.d.). Collections represent a group of objects, that are known as its elements. The Java Development Kit (JDK) does not offer implementation on this interface, rather providing implementations of more specific sub interfaces like Lists or Maps. These interfaces can be used to pass Collections around or even manipulate them (Oracle, 2020a). The following sub chapters will cover two main possibilities in Apache Groovy to handle Collections.

4.2.3.1 Lists

A list is commonly known under the term array and stores objects sequentially and is accessible through their integer indices. For the fictive company it is important to provide a changing and flexible welcome page, where different articles for various topics are shown to the guest users (i.e., that do not have any preferences stored). In this case two lists have been used to create a randomized list of articles of different topics. 'Number' is for this example representative for the latest 10 articles of the respective topic from newest to oldest. The other data type is 'topic' and is a list covering the five main topics of the website.

```
<s:script type="groovy" cacheSrc="false">
  def number = (1..10)
  def topic = ['Marketing - ', 'IT - ', 'Finance - ',
'Economics - ', 'Mathematics - ']

  def preference = [topic, number].combinations().collect
{ it.join() }
  def dashboard = preference.asImmutable()
  assert preference.size() == 50
  preference.shuffle()
  assert preference.every { pref -> dashboard.contains(pref) }
  println preference.take(5)
</s:script>
```

Figure 18. Groovy List

By applying this code to the website, it should be possible to receive a randomized list of topics and numbers that could be used in order to provide a randomized dashboard to guests or new users without preferences. The list should look similar to Figure 19, but changes for every refresh of the web page.

```
[Mathematics - 8, Finance - 1, Mathematics - 2, IT - 5, Marketing - 2]
```

Figure 19. Possible result Groovy List

4.2.3.2 Maps

In the previous sub-chapter, lists were introduced, and the example showed a possibility of a data structure that holds objects. In comparison to lists where the objects are being keyed by a linear progression of integer indices, maps use strings. In this way it is possible to attach data to a so called 'key' or 'key-value pairs' like in this case (Babal, 2015). In Figure 20 a key value pair of all users can be seen in form of a map consisting of the user, specifically all usernames, which represents the key and the preference (i.e. Economics, Marketing, ...) being the respective value of the key. Therefore, the values have been indexed by their keys. This map could be useful internally for the marketing department and could provide an overview of the importance and validity of the particular news topics. Needless to say, the usernames in this case are exemplary and would be gathered by e.g., a database in a real-life context.

```
<s:script type="groovy" cacheSrc="false">
  def user = [
    user123 : 'Economics',
    user234 : 'Marketing',
    user657 : 'Economics',
    user986 : 'IT'
  ]
  user.each { key, val ->
    println "$key = $val,"
  }
</s:script>
```

Figure 20. Groovy Map

By applying the code for the map, it is possible to review the users with the username (the key) and their corresponding preference of the news topics. This list can be used for various domains like marketing purposes or reporting issues.

```
user123 = Economics, user234 = Marketing, user657 = Economics, user986 = IT,
```

Figure 21. Result Groovy Map

4.3 PHP Examples

The theoretical background of PHP has been discussed in Chapter 3.4 and will be underlined by some nutshell examples. Furthermore, the examples will firstly cover a comparison to some of the previously introduced examples with Apache Groovy, secondly two main features of PHP for website development will be explained and code examples provided. To see the output of the respective nutshell examples, browse to 'JSR-223 Examples' on the index page of the demo installation (see Chapter 3.3). The nutshell examples are stored in `webapps/demo/jsr223.jsp` and the page should look like Figure 22.



Figure 22. jsr223.jsp

4.3.1 Hello World

To test the installation of PHP, it is necessary to navigate to the script `webapps/demo/jsr223.jsp` and to add the code from Figure 23. If the text from Figure 24 is displayed on the corresponding web address of the JSP file, the installation has been successful.

```
<s:script type="php" cacheSrc="false">  
    <?php echo '<p>Hello World from PHP!</p>'; ?>  
</s:script>
```

Figure 23. PHP „Hello World“



Hello World from PHP!

Figure 24. Result PHP „Hello World“

4.3.2 Decision Making

As already stated in Chapter 4.2.2, decision making enables the possibility to have statements in the code, that change a certain outcome. Equally to the Apache Groovy examples, the If-Else-If as well as the Switch statement will be elaborated and visualized with nutshell examples and the differences will be outlined.

4.3.2.1 If-Else-If Statement

Generally speaking, the decision-making syntax is very similar in both scripting languages. Still, the declaration and conditions are defined differently when using PHP. The If-Else-If statement has been designed equally to the one based on Apache Groovy in Chapter 4.2.2.1, to outline the significant differences even more. First, the declaration of the variable is different, as it is done using a dollar sign instead of the word 'def'. Second, the conditions for the various cases are different, as the variable for user status is received using the dollar sign. Also, the comparative operator works differently in PHP, then in Apache Groovy, and is defined using two equal signs.

```
<s:script type="php" cacheSrc="false">
<!DOCTYPE html>
  <?php
    $userstatus = user;

    if ($userstatus == "user") {
      echo "You can find the latest news from your field of
interest here!";
    } elseif ($userstatus == "premium") {
      echo "View your personalized news section!";
    } else {
      echo "Have a look at the latest news!";
    }
  ?>
</html>
<title>Cookies</title>
</html>
</s:script>
```

Figure 25. PHP If-Else-If Statement

By applying the code from Figure 25 the result should be similar to the example with Apache Groovy (see Figure Figure 15). Even if the scripting languages were developed and are applied for different purposes, there are still functionalities that work equally for both.



You can find the latest news from your field of interest here!

Figure 26. Result PHP If-Else-If Statement

4.3.2.2 *Switch Statement*

The switch statement can be used similarly to all forms of if statements for the same expression. It is especially practical for comparisons of the same variable with different values and executing a code piece depending on the result. The difference to the If-Else-If statement is that the condition in a switch statement is only evaluated once, and the result is compared to each of the defined cases. In an If-Else-If statement in comparison, the condition is evaluated again. Generally speaking, if the condition is more complex than a simple comparison, a switch can be faster and more effective in these cases (The PHP Group, 2020d).

In this example, the purpose is equal to the switch statement in the Apache Groovy chapter. (see Chapter 4.2.2.2). Still, the syntax has some differences in the declaration of variables and the general format of the switch statement. As in the previous switch example for Apache Groovy, the result remains the same for PHP.

```
<s:script type="php" cacheSrc="false">
<!DOCTYPE html>
  <?php
    $preference = "IT";

    switch ($preference) {
    case "IT":
    echo "See the latest IT news";
    break;
    case "Marketing":
    echo "See the latest marketing news";
    break;
    case "Finance":
    echo "See the latest finance news!";
    break;
    default:
    echo "You can see some articles that you may find
interesting";
    }
  ?>
</s:script>
<html>
<title>Switch</title>
</html>
```

Figure 27. PHP Switch Statement

Similarly, to the Apache Groovy chapter with the switch example, the users receive a personalized headline on the website according to their preference. As in the other examples, the default case would display a generic message to the user, if no preference is stored or known. In this case, the result in Figure 28 focuses on a message regarding the IT sector, as IT is stored as the preference.

See the latest IT news

Figure 28. Result PHP Switch Statement

4.3.3 Sessions

Sessions can be used in PHP to preserve certain data across subsequent accesses. This enables a more customized application and experience of the website. Visitors accessing the website get assigned a unique id, that can also be accessed as session id. It is possible to store session information in a cookie (see 4.3.4) or in the URL of the webpage. The advantage of sessions in PHP is when a visitor accesses the website, PHP is able to automatically check if a session id has been forwarded with the page request. If this is the case, it is possible to alter the website according to the information from the session (The PHP Group, 2020c).

The example in Figure 29 shows the start of a session and the setting of a session id for the specific user. Beneath an if statement is set to distinguish between two different conditions. First, the condition where no preference is set, this could happen if the user visits a website for the first time and the landing page is not specifically topic related. Therefore, no session id has been set, as it is the first time on the website, and cannot be set, because the preference is not conclusive. Second, the user could be a first-time visitor on a topic-related landing page, like in this example, or a user who has already a preference stored as a session id, the condition checks for a preference, and displays a customized pop-up especially for the stored preference.

```
<s:script type="php" cacheSrc="false">
  <!DOCTYPE html>
  <?php
    session_start();
    $_SESSION["preference"] = "IT";
    if(!isset($_SESSION)) {
      echo "Welcome to our page Guest!";
    } else {
      echo "Do you want to view the latest news from the " .
$_SESSION["preference"] . " sector?";
    }
  ?>
</s:script>
```

Figure 29. PHP Sessions

By applying the code from above, the output is customized, because a session id is set for the visitor. Consequently, the visitor could be held longer on the website, by directly offering more information on the possible preferred topic. The output for the coding example from above should look like Figure 30 and provide a topic-related message.

Do you want to view the latest news from the IT sector?

Figure 30. Result PHP Session

4.3.4 Cookies

Cookies are a mechanism to store data in the browser and track returning users. As they are a part of the HTTP header, they need to be called before any output is sent to the browser (The PHP Group, 2020a). In comparison to sessions, that have been introduced previously, cookies have the advantage that not only data can be saved in a website session, but also data can be saved after the session for a predefined time period (Reimers, 2020). The reason for this is, that cookies are saved as a file on the computer of the visitor. Still, the visitor has the possibility to prohibit a website from saving cookies, as the general data protection regulation (GDPR) has defined, that the user needs to receive the possibility to reject cookies and have an overview what kind of cookies are saved (WKO, 2019).

For the example in Figure 31, two cookies are saved on for the user. On the one hand, the user status is saved and on the other hand the name of the user. This data could be gathered from a data field on the web page or stored in a database where it can be retrieved by checking the user credentials. Afterwards, the cookies are being set and a time period for the storage of the cookies is defined.

```
<s:script type="php" cacheSrc="false">
  <!DOCTYPE html>
  <?php
    $cookie_name = "user";
    $cookie_value = "Marion";
    setcookie($cookie_name, $cookie_value, time() + (86400 * 30),
"/");

    if(!isset($cookie_name)) {
      echo "Welcome to our page Guest!";
    } else {
      echo "Welcome to our page " . $cookie_value . "!";
    }
  ?>
  <html>
  <title>Cookies</title>
  </html>
</s:script>
```

Figure 31. PHP Cookies

The above example is not only used to save cookies, but also to use these cookies. In this case two conditions are possible for the further usage of cookies. First, the cookie name is not set, e.g., no user status is available, therefore the user appears to be a guest and is welcomed with a generic truism. Second, if the user status is set to user, a personalized welcome quote is displayed where the username is displayed.



Figure 32. Result PHP Cookie

5 Conclusion

To conclude this seminar paper a brief summary is provided that gives an overview of the major findings of this thesis. Additionally, an outlook is given that can be used as an approach for future research in this area. In this context, possible topics to be covered are proposed.

5.1 Summary

In the theoretical part of the seminar paper, the two scripting languages have been outlined. In this section the application areas were discussed, and it could be said that Apache Groovy and PHP are mostly applied in different contexts. Still, as the nutshell examples showed, it is possible in various cases to have a decision between both of the coding languages. Of course, it is necessary to review the background of the necessary application to decide whether one or the other is to be preferred.

Another finding was, that it is possible to apply both scripting languages with the use of the tag libraries in JSPs. Clearly it was expectable for Apache Groovy, as it is one hundred percent compatible with Java, and therefore appropriate for Java Server pages. Still, with the use of Quercus it was also possible to also include PHP code into the scripts, even though it is not a Java-based language.

Lastly, a positive finding needs to be mentioned, as Apache Tomcat was exposed to be not only a reliable, but also to be a well-documented web server. In comparison to competing products, that are often cost-intensive, Apache Tomcat offers many services and enables its users' numerous features. Also, the configuration progress was easy, as well as the installation of an individual domain.

5.2 Outlook

This seminar paper was an approach to show some basic functionalities of Apache Groovy and PHP with Apache Tomcat and the tag libraries. Positively, the implementations for both languages were working accurately and had no problem being implemented. Further work could focus on other scripting languages, that are either Java based or not and test if the functionalities can be implemented in the JSPs using tag libraries.

For this seminar paper, the aim was to provide short and simple nutshell examples that show the functionality from different perspectives. Another interesting approach for future work would be the development of more complex examples for both scripting languages and if they would have an effect on the performance of the scripts. This could also be combined with the first approach, as it would be interesting to review not only other scripting languages, but in this case also more complex examples.

Bibliography

- Agrawal, S., & Gupta, R. D. (2014). Development and Comparison of Open Source based Web GIS Frameworks on WAMP and Apache Tomcat Web Servers. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL(4), 1–5. Publicly Available Content Database (1757059819). <https://doi.org/10.5194/isprsarchives-XL-4-1-2014>
- Babal, H. (2015, August 27). Groovy Map Example | Examples Java Code Geeks—2020. Retrieved December 10, 2020, from Examples Java Code Geeks website: <https://examples.javacodegeeks.com/jvm-languages/groovy/groovy-map-example/>
- Caucho Technology. (2014). Caucho Resin: Reliable, Open-Source Application Server. Retrieved November 7, 2020, from <http://quercus.caucho.com/>
- Davis, M. (2014, June 17). Five Reasons You Should Use Tomcat (Updated for 2020). Retrieved November 9, 2020, from Future Hosting website: <https://www.futurehosting.com/blog/five-reasons-you-should-use-tomcat/>
- Leitenmüller, H. (2001). JSP- Java Server Pages. Retrieved November 7, 2020, from <http://www.ssw.uni-linz.ac.at/Teaching/Lectures/Sem/2000/Leitenmueller/>
- Niemeyer, P., & Leuck, D. (2013). WAR Files and Deployment—Learning Java, 4th Edition [Book]. Retrieved December 17, 2020, from <https://www.oreilly.com/library/view/learning-java-4th/9781449372477/ch15s03.html>
- Oracle. (2010). Tag Library Descriptors—The Java EE 5 Tutorial. Retrieved December 17, 2020, from <https://docs.oracle.com/javaee/5/tutorial/doc/bnamu.html>
- Oracle. (2020a). Collection (Java Platform SE 7). Retrieved December 11, 2020, from <https://docs.oracle.com/javase/7/docs/api/java/util/Collection.html>
- Oracle. (2020b). Jar-The Java Archive Tool. Retrieved December 17, 2020, from <https://docs.oracle.com/javase/7/docs/technotes/tools/windows/jar.html#description>
- Oracle Corporation. (2020). The Java Community Process(SM) Program—Communityprocess—Final. Retrieved November 7, 2020, from

- <https://jcp.org/aboutJava/communityprocess/final/jsr245/index.html>
reality rehab. (n.d.). Groovy for Beginners. Retrieved November 7, 2020, from
<https://reality.rehab/groovy>
- Reimers, N. (2020). Cookies – PHP lernen. Retrieved December 12, 2020, from
Cookies – PHP lernen website:
<https://www.php-einfach.de/php-tutorial/cookies/>
- Slashdot Media. (2020). BSF4ooRexx—Browse /Sandbox/rgf/taglibs/beta at
SourceForge.net. Retrieved November 7, 2020, from
<https://sourceforge.net/projects/bsf4oorexx/files/Sandbox/rgf/taglibs/beta/>
- The Apache Groovy Project. (2020a). Central Repository: Org/codehaus/groovy.
Retrieved November 4, 2020, from
<https://repo1.maven.org/maven2/org/codehaus/groovy/>
- The Apache Groovy Project. (2020b). The Apache Groovy programming language—
Differences with Java. Retrieved December 12, 2020, from [https://groovy-
lang.org/differences.html](https://groovy-lang.org/differences.html)
- The Apache Groovy Project. (2020c). The Apache Groovy programming language—
Ecosystem. Retrieved December 12, 2020, from
<https://groovy-lang.org/ecosystem.html>
- The Apache Groovy Project. (2020d). The Apache Groovy programming language—
Groovy reference documentation. Retrieved November 7, 2020, from
<http://groovy-lang.org/single-page-documentation.html>
- The Apache Software Foundation. (2011). Apache Commons BSF™—Bean Scripting
Framework. Retrieved November 7, 2020, from
<http://commons.apache.org/proper/commons-bsf/>
- The Apache Software Foundation. (2020a). Apache Tomcat®—Apache Tomcat 9
Software Downloads. Retrieved November 7, 2020, from
<https://tomcat.apache.org/download-90.cgi>
- The Apache Software Foundation. (2020b). Apache Tomcat®—Welcome! Retrieved
November 7, 2020, from <http://tomcat.apache.org/>
- The PHP Group. (2012). PHP: Command line usage—Manual. Retrieved November
7, 2020, from <https://www.php.net/manual/en/features.commandline.php>
- The PHP Group. (2016). PHP-GTK. Retrieved November 7, 2020, from
<http://gtk.php.net/>

- The PHP Group. (2020a). PHP: Cookies—Manual. Retrieved December 12, 2020, from <https://www.php.net/manual/en/features.cookies.php>
- The PHP Group. (2020b). PHP: Installation and Configuration—Manual. Retrieved November 7, 2020, from <https://www.php.net/manual/en/install.php>
- The PHP Group. (2020c). PHP: Introduction—Manual. Retrieved December 12, 2020, from <https://www.php.net/manual/en/intro.session.php>
- The PHP Group. (2020d). PHP: switch—Manual. Retrieved December 12, 2020, from <https://www.php.net/manual/en/control-structures.switch.php>
- The PHP Group. (2020e). PHP: Was ist PHP? - Manual. Retrieved November 7, 2020, from <https://www.php.net/manual/de/intro-what-is.php>
- Tomes, M. (2020). Demo War for Nutshell Examples. Retrieved December 15, 2020, from GitLab website:
<https://gitlab.com/seminarpaperwu/tomcatnutshellexamples>
- WKO. (2019). Checkliste für Cookies und Web-Analyse im Webshop. Retrieved December 12, 2020, from <https://www.wko.at/service/wirtschaftsrecht-gewerberecht/checkliste-cookies-webanalyse-webshop.html>
- Zambon, G. (2012). *Beginning JSP, JSF and Tomcat: Java Web Development*. Apress.

Appendix (A)

```
<%@ page session="false" pageEncoding="UTF-8" contentType="text/html;
charset=UTF-8" %>
<%@ taglib uri="/WEB-INF/script-bsf.tld" prefix="s" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <style>
    .general {
      font-family: "Monaco", "Courier New", monospace;
      font-size: 13px;
    }

    h1 {
      background-color: #c27974;
      width: 630px;
    }

    p {
      font-size: 14px;
      font-weight: bold;
      color: black;
    }

    summary {
      font-size: 14px;
      font-weight: bold;
      color: #c27974;
    }

    .result {
      width: 600px;
      border: 1px solid #333;
      box-shadow: 8px 8px 5px #444;
      padding: 8px 12px;
      background-image: linear-gradient(180deg, #fff, #dddddd 40%,
#ccc);
      font-family: "Courier New", Courier, "Lucida Sans Typewriter",
"Lucida Typewriter", monospace;
      font-weight: bold;
    }
  </style>
  <title>Groovy Application</title>
</head>
<body>
<div class="general">
  <h1>BSF</h1>
  <details>
    <summary>Hello World</summary>
    <p>out.println("Hello World from Groovy!")</p>
  </details>
  <br>
  <div class="result">
    <s:script type="groovy" cacheSrc="false">
      out.println("Hello World from Groovy!")
    </s:script>
  </div>
</body>
</html>
```



```

</div>

<br>
<br>

<details>
  <summary>If-Else-If</summary>
  <p>def userstatus = 'user'<br>
    if (userstatus='user') {<br>
      out.println("You can find the latest news from your field of
interest here!");<br>
    } else<br>
    if (userstatus = 'premium') {<br>
      out.println("View your personalized news section!");<br>
    } else {<br>
      out.println("Have a look at the latest news!");<br>
    }</p>
</details>
<br>
<div class="result">
  <s:script type="groovy" cacheSrc="false">
    def userstatus = 'user'
    if (userstatus='user') {
      out.println("You can find the latest news from your field of
interest here!");
    } else
    if (userstatus = 'premium') {
      out.println("View your personalized news section!");
    } else {
      out.println("Have a look at the latest news!");
    }
  </s:script>
</div>

<br>
<br>

<details>
  <summary>Switch</summary>
  <p>def preference = 'IT'<br>
    switch(preference) {<br>
      case 'IT':<br>
        out.println("See the latest news on IT");<br>
        break;<br>
      case 'Marketing':<br>
        out.println("See the latest marketing news");<br>
        break;<br>
      case 'Finance':<br>
        out.println("See the latest finance news!");<br>
        break;<br>
      default:<br>
        out.println("You can see some articles that you may find
interesting");<br>
        break;<br>
    }</p>
</details>
<br>
<div class="result">
  <s:script type="groovy" cacheSrc="false">
    def preference = 'IT'
  
```

```

switch(preference) {
  case 'IT':
    out.println("See the latest news on IT");
    break;
  case 'Marketing':
    out.println("See the latest marketing news");
    break;
  case 'Finance':
    out.println("See the latest finance news!");
    break;
  default:
    out.println("You can see some articles that you may find
interesting");
    break;
}
</s:script>
</div>

<br>
<br>

<details>
  <summary>Lists</summary>
  <p>def number = (1..10)<br>
    def topic = ['Marketing - ', 'IT - ', 'Finance - ', 'Economics
- ', 'Mathematics - ']<br>
    def preference = [topic, number].combinations().collect
{ it.join() }<br>
    def dashboard = preference.asImmutable()<br>
    assert preference.size() == 50<br>
    preference.shuffle()<br>
    assert preference.every { pref ->
dashboard.contains(pref) }<br>
    println preference.take(5)</p>
</details>
<br>
<div class="result">
  <s:script type="groovy" cacheSrc="false">
    def number = (1..10)
    def topic = ['Marketing - ', 'IT - ', 'Finance - ', 'Economics
- ', 'Mathematics - ']

    def preference = [topic, number].combinations().collect
{ it.join() }
    def dashboard = preference.asImmutable()
    assert preference.size() == 50
    preference.shuffle()
    assert preference.every { pref -> dashboard.contains(pref) }
    println preference.take(5)
  </s:script>
</div>

<br>
<br>

<details>
  <summary>Maps</summary>
  <p>def user = [<br>
    user123 : 'Economics',<br>
    user234 : 'Marketing',<br>

```

```
        user657 : 'Economics',<br>
        user986 : 'IT'<br>
    ]<br>
    user.each { key, val -><br>
        println "$key = $val,"<br>
    }
</details>
<br>
<div class="result">
    <s:script type="groovy" cacheSrc="false">
        def user = [
            user123 : 'Economics',
            user234 : 'Marketing',
            user657 : 'Economics',
            user986 : 'IT'
        ]
        user.each { key, val ->
            println "$key = $val,"
        }
    </s:script>
</div>
</div>
</body>
</html>
```

Appendix (B)

```
<%@ page session="false" pageEncoding="UTF-8" contentType="text/html;
charset=UTF-8" %>
<%@ taglib uri="/WEB-INF/script-jsr223.tld" prefix="s" %>
<!DOCTYPE>
<html>
<head>
  <style>
    .general {
      font-family: "Monaco", "Courier New", monospace;
      font-size: 13px;
    }

    h1 {
      background-color: #c27974;
      width: 630px;
    }

    p {
      font-size: 14px;
      font-weight: bold;
      color: black;
    }

    summary {
      font-size: 14px;
      font-weight: bold;
      color: #c27974;
    }

    .result {
      width: 600px;
      border: 1px solid #333;
      box-shadow: 8px 8px 5px #444;
      padding: 8px 12px;
      background-image: linear-gradient(180deg, #fff, #dddddd 40%,
#ccc);
      font-family: "Courier New", Courier, "Lucida Sans Typewriter",
"Lucida Typewriter", monospace;
      font-weight: bold;
    }
  </style>
</head>
<body>
<div class="general">
  <h1>JSR-223</h1>
  <br>
  <details>
    <summary>Hello World</summary>
    <p><span><</span><span>?</span><span>php </span> <span>echo 'Hello
World from PHP!'; ?</span>
  </details>
  <br>
  <div class="result">
    <s:script type="php" cacheSrc="false">
      <?php echo 'Hello World from PHP!'; ?>
    </s:script>
  </div>
</div>
</body>
</html>
```

```

</s:script>
</div>

<br>

<br>
<details>
  <summary>If-Else-If</summary>
  <p><span><</span><span>?</span><span>php </span> <span>$userstatus
= user;<br>

  if ($userstatus == "user") {<br>
    echo "You can find the latest news from your field of interest
here!";<br>
  } elseif ($userstatus == "premium") {<br>
    echo "View your personalized news section!";<br>
  } else {<br>
    echo "Have a look at the latest news!";<br>
  }<br>
?></span>
</details>
<br>
<div class="result">
<s:script type="php" cacheSrc="false">
  <!DOCTYPE html>
  <?php
    $userstatus = user;

    if ($userstatus == "user") {
      echo "You can find the latest news from your field of interest
here!";
    } elseif ($userstatus == "premium") {
      echo "View your personalized news section!";
    } else {
      echo "Have a look at the latest news!";
    }
  ?>
</s:script>
</div>

<br>
<br>
<details>
  <summary>Switch</summary>
  <p><span><</span><span>?</span><span>php </span> <span>$preference
= "IT";<br>

  switch ($preference) {<br>
    case "IT":<br>
      echo "See the latest news on IT";<br>
      break;<br>
    case "Marketing":<br>
      echo "See the latest marketing news";<br>
      break;<br>
    case "Finance":<br>
      echo "See the latest finance news!";<br>
      break;<br>
    default:<br>
      echo "You can see some articles that you may find interesting";<br>
    }<br>
  }<br>

```

```

    ?></span>
</details>
<br>
<div class="result">
<s:script type="php" cacheSrc="false">
    <!DOCTYPE html>
    <?php
        $preference = "IT";

        switch ($preference) {
            case "IT":
                echo "See the latest news on IT";
                break;
            case "Marketing":
                echo "See the latest marketing news";
                break;
            case "Finance":
                echo "See the latest finance news!";
                break;
            default:
                echo "You can see some articles that you may find interesting";
        }
    ?>
</s:script>
</div>

<br>
<br>
<details>
    <summary>Cookies</summary>
    <p><span><</span><span>?</span><span>php </span> <span>$cookie_name
= "user";<br>
    $cookie_value = "Marion";<br>
    setcookie($cookie_name, $cookie_value, time() + (86400 * 30),
"/");<br>

    if(!isset($cookie_name)) {<br>
        echo "Welcome to our page Guest!";<br>
    } else {<br>
        echo "Welcome to our page " . $cookie_value . "!";<br>
    }<br>
    ?></span>
</details>
<br>
<div class="result">
<s:script type="php" cacheSrc="false">
    <!DOCTYPE html>
    <?php
        $cookie_name = "user";
        $cookie_value = "Marion";
        setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");

        if(!isset($cookie_name)) {
            echo "Welcome to our page Guest!";
        } else {
            echo "Welcome to our page " . $cookie_value . "!";
        }
    ?>
</s:script>
</div>

```

```

<br>
<br>
<details>
  <summary>Sessions</summary>
  <p><span><</span><span>?</span><span>php </span>
<span>session_start();<br>
  $_SESSION["username"] = "marion";<br>
  $_SESSION["preference"] = "IT";<br>
  echo "Do you want to view the latest news from the " .
$_SESSION["preference"] . " sector?";<br>
  ?></span>
</details>
<br>
<div class="result">
<s:script type="php" cacheSrc="false">
  <!DOCTYPE html>
  <?php
  session_start();
  $_SESSION["preference"] = "IT";
  if(!isset($_SESSION)) {
  echo "Welcome to our page Guest!";
  } else {
  echo "Do you want to view the latest news from the " .
$_SESSION["preference"] . " sector?";
  }
  ?>
</s:script>
</div>
</div>
</body>
</html>

```

Appendix (C)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <title>Apache Tomcat Nutshell Examples (Groovy, PHP)</title>
  <style>
    .general {
      font-family: "Monaco", "Courier New", monospace;
      font-size: 13px;
    }

    .rightText {
      width: 500px;
      height: 150px;
      display: inline-block;
      float: left;
    }

    .leftText {
      width: 600px;
      height: 300px;
      display: inline-block;
      float: left;
    }
  </style>
</head>
<body>
<div class="general">
  <h1>Apache Tomcat Nutshell Examples (Groovy, PHP)</h1>
  <div class="leftText">

    <p>The following tag libraries (taglibs) can be applied for the
same
    purpose:
    <ul>
      <li><b>script-bsf.tld: </b>Apache BSF (Bean Scripting
Framework)
```


`
`This is the original Java scripting framework that can be used for Groovy.

`
<u>`Important:`</u>` Groovy is also supported by the newer

scripting framework (see below). `
`

`<p></p>`

`script-jsr223.tld: `JSR-223 (Java's javax.script Framework)

`
`This is the Java scripting framework that got introduced with Java 6 and works with Groovy and PHP.

``

`<p><u>`Example Files`</u></p>`

``

``BSF Examples`` (using Groovy via BSF)

``

`<p>`

``JSR-223 Examples`` (using Groovy and PHP via JSR-223)``

``

`<p></p>
`

`</div>`

`<div class="rightText">`

`<p>`The example files are JSP (Java server page) files that are usually created with

HTML or xhtml and have usually Java `"Servlet"` code injected. When creating

the HTML page the code gets executed and usually creates part of the resulting HTML file.

`<p>`The scripting tag libraries enable the user to use any Java script language instead or in

addition to Java using the Apache BSF framework or the Scripting for the java platform (JSR-223) framework.

`<p>`The log files can be found in the Tomcat log directory.`</p>`

`</div>`

`</div>`

</body>

</html>