JAVA SPRING BOOT REST WEB SERVICE INTEGRATION WITH JAVA ARTIFICIAL INTELLIGENCE WEKA FRAMEWORK

Željko Jovanović, Dijana Jagodić, Dejan Vujičić, Siniša Ranđić
Faculty of Technical Sciences in Čačak, University of Kragujevac, Serbia

Abstract
This paper presents the integration of Java Spring Boot framework with WEKA – Java artificial intelligence framework. For a presentation, developed REST Web service is demonstrated. As a result, it returns result of prediction for four prediction algorithms that are used over some sample weather data.

Keywords: Java Spring, Java Spring Boot, WEKA, REST WS.

INTRODUCTION

Rapid development of programming technologies demands testing of integration of popular frameworks. Java as one of the most used programming languages has variety of frameworks for the different purposes.

Probably the most popular framework is The Spring Framework [1]. Its popularity is based on large scale integrations possibility. This advantage becomes the main problem, since it could be difficult to set up application in appropriate way. This problem is mainly solved with Spring Boot [2] version which automate most of configuration processes. This allows developers to focus on application logic realization.

As for technologies, Representational state transfer (RESTfull) Web Service are probably the standard nowadays for providing data from the application business logic part. Spring Boot provides very good tools for REST WS development.

We are all witnesses of artificial intelligence usage in variety of application. This is probably the main trend nowadays in a programming world. Java provides WEKA framework [8] for artificial intelligence integration in Java based applications.

This paper present integration of this two popular Java frameworks. As a result Java Spring Boot REST WS is developed to return the accuracy of four prediction algorithms using WEKA over sample data. This paper is outlined as follow: First Java Spring framework with its Spring Boot version is presented. After that WEKA framework and its usage are shown. As a result, demonstration of this framework integration is presented.

JAVA SPRING

The Spring Framework [1] is a major open source Java/J2EE application development framework for more productive application development. It is the most popular Java framework with 30% share of usage. The Spring Framework features enable efficient development from simple Web to complex enterprise applications. Main concepts that the Spring Framework depends are:

- Inversion of Control (IoC)
- Dependency Injection (DI)
- Aspect Oriented Programming (AOP)
- Java persistence API (JPA)

IoC is the general concept where control of flow is Inverted : instead of the programmer controlling the flow of a program, the external sources (framework, services, other components) take control of it. DI, as presented in [3] is a pattern in a form of IoC, nowadays key concept in Spring and the rapidly-growing Google Guice. AOP [4] is based on the conception that improves both modularity and the structure of code. JPA [5] is in charge for the mapping of the object state to the database columns and how to issue queries across the objects.

The Spring framework consist of around 20 modules. Its architecture is presented in Fig 1.

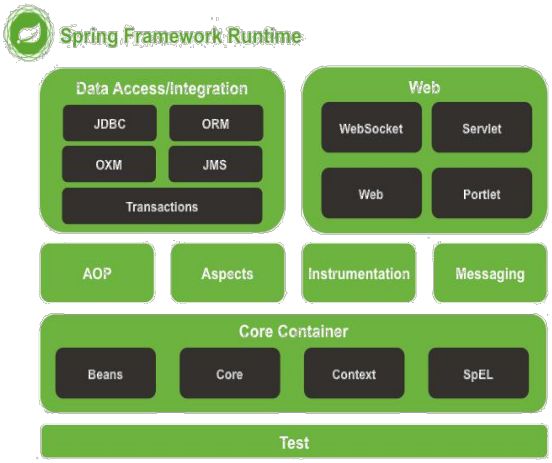

Fig. 1. *The Spring Framework arhitecture [1]*

Some of the most important modules are Core Container, Data Access/Integration, WEB, AOP, Instrumentation, Messaging, Test... For the development of Web applications, Web module is most important. It consists of Web, WebSockets, Portlet and Servlet module. Servlet module contains the definitions of the two most used concepts nowadays. First one is Spring Model-View-Controller (MVC). Second one is RestFull WebSerive integrations (REST WS).

Spring MVC [6] provides in-depth coverage of Spring MVC and Spring Web Flow, two highly customizable and powerful web frameworks which enables development of multi-layered applications. It is "Open for expansion, closed for modification". Spring MVC is built on *DispacherServlet* which pass the requests to appropriate *@Controller-s* based on the *@RequestMapping* annotations. Spring Web Flow is a project that complements Spring MVC for building reusable web controller modules that encapsulate rich page navigation rules.

REST WS is architecture for providing interoperability between computer systems on the Internet. Nowadays it is one of the most used techniques for data providing to various types of data consumers. The Spring framework provides support for REST WS development. Java class that will represent REST WS need to be annotated as a @RestController. Methods that will provide functionalities need to be annotated with unique @RequestMapping anotation. Using ResponseEntity class as a method result type entire set of HTTP status code, headers, and body could be specified. It comes with several constructors to carry the information sent in HTTP Response.

SPRING BOOT

Spring Boot is designed to simplify Spring application development. In [2] main concepts are presented. Among others, most important are:

- Automatic configuration - configure applications as standard Spring applications
- Starter dependences - integrate required dependences automatically
- The command-line-interpreter - enables application control thought console
- The actuator - show information about events in application.

Spring Boot provides a radically faster and widely accessible getting started experience for all Spring development and provides a range of non-functional features that are common to large classes of projects (e.g. embedded servers, security, metrics, health checks, externalized configuration); and with absolutely no code generation nor requirements for XML configurations. Also, application testing is easier to do as described in [7]. At the URL address:

https://start.spring.io/

it is possible to set up and download start project with adjusted dependences. It is possible to chose between Maven or Gradle projects, and it is only necessary to check which dependences should be included. For developing REST WS application WEB module dependency is needed.

WEKA WORKBENCH

WEKA (Waikato Environment for Knowledge Analysis) was developed at the University of Waikato in New Zealand. It is written in Java and provides interfaces to numerous machine learning algorithms. WEKA also supports the methods for data processing and evaluation of learning results [8] .

Since its beginning in 1992, WEKA has been widely adopted by many scientists in various areas of expertise. Some of the reasons

for its popularity may be found in some of WEKA's main characteristics [9]:

- Data preprocessing: WEKA supports numerous file formats and databases; it also possesses methods for filtering the data;
- Classification: WEKA contains more than 100 classification methods (Bayesian, lazy, rule-based methods, tree learners, function-based learners, etc.);
- Clustering: several algorithms for unsupervised learning are supported in WEKA;
- Attribute selection: there are various search methods and criteria in WEKA;
- Data visualization: WEKA supports plotting data in dependence of class or other attribute values; many different types of data visualization methods are supported, such as tree viewers, Bayes network viewers, dendrogram viewers, etc.

However, despite all the characteristics it possesses, WEKA has become vastly popular because of its intuitive and easy-to-use GUI. In this way, the machine learning algorithms had become available to almost anybody who wants to classify some data, not even necessarily being a programmer [10].

WEKA integrates several GUIs for easy access to the certain functionalities. WEKA GUI Chooser window is shown in Fig. 2.

Fig. 2. *WEKA GUI Chooser window*

By clicking on the Explorer button, the WEKA Explorer window is shown, Fig. 3. This window has several tabs [10]:

- Preprocess: loading and transformation of data with filters;
- Classify: enables access to WEKA's algorithms for classification and regression;
- Cluster: enables access to WEKA's clustering algorithms;
- Associate: access to the association rule mining;
- Select attributes: selection of algorithms and evaluation criteria for determination of the most important attributes in dataset;
- Visualize: different options for data visualization.

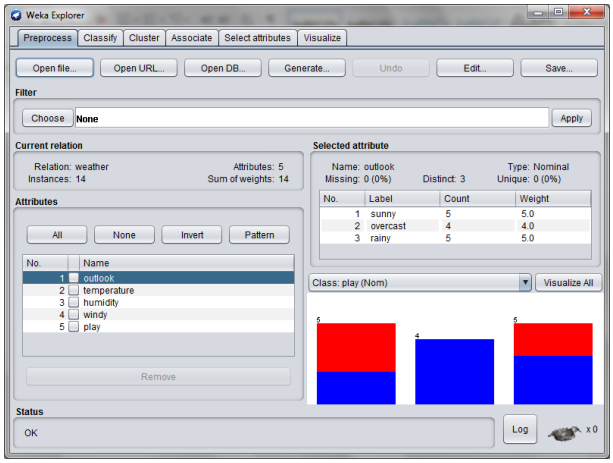

Fig. 3. *WEKA Explorer window*

The possibilities of using WEKA as a data mining tool in bioinformatics were discussed in [11]. In [12], authors used WEKA for predicting the phospholipidosis inducing potential. WEKA found its usage in traffic flow clustering [13]. Authors [14], [15] used WEKA for cancer classification from gene selection. WEKA was used as a tool for improving performance of learning from imbalanced data in [16]. Interesting study on using machine learning WEKA algorithms for uncovering spammers on social networks was conducted in [17]. Authors of [18] used machine learning methods for classifying human physical activity from on-body accelerometers. WEKA was also used for classification of IP traffic flow [19], [20] and botnet traffic [21]. In [22], machine learning WEKA algorithms were used for detection and annotation of real – life emotions. WEKA found its application in agriculture in [23].

The list of projects related to WEKA can be found at [24] and at the time of the writing of this paper, that number is 69.

SPRING BOOT - WEKA INTEGRATION

As explained in Spring Boot section, start version of maven project with WEB, REST, JPA automatic dependences integration is download from https://start.spring.io/. After extraction, this project is imported to Eclipse J2EE IDE as a maven project. weka jar is added as additional jar in project->properties->Java build path. After this WEKA library functionalities are available for usage. To develop REST WS using Java Spring Boot java class called WekaRestController.java is created. Project structure of created application is presented on Fig. 4.

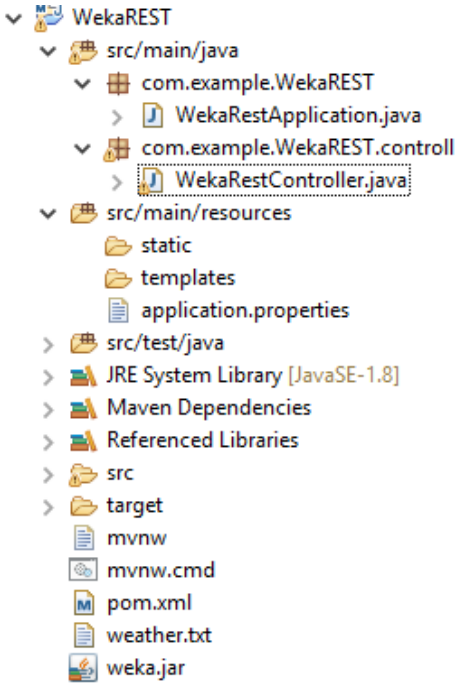

Fig. 4. *Spring Boot REST WS project structure with WEKA support*

WEKA's trees.J48 classifier algorithm gives the visualization tree of the loaded data, shown in Fig. 5.

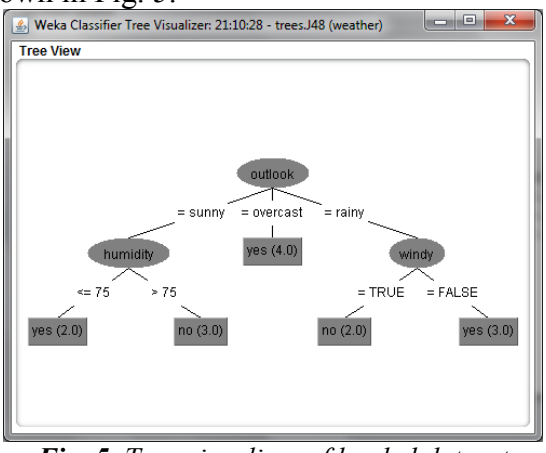

Fig. 5. *Tree visualizer of loaded dataset*

Main code parts of WekaRestController class are presented below.

```
@RestController
public class WekaRestController {

@RequestMapping(value = "/wekaTest",
method = RequestMethod.GET)

public ResponseEntity<List<String>>
predictions() {
        List<String> acc;
        // calculate prediction for rules...

        // add result to list acc

return new ResponseEntity<List<String>>
        (acc, HttpStatus.OK);
        }
}
```

As shown, class is annotated with *@RestController* anotation defined in *org.springframework.web.bind.annotation.RestControlle*. URL for WS integration is mapped as "/wekaTest" using *@RequestMapping* annotation. Java method "*prediction*()" is created for this URL mapping processing.

As a demonstration of successful Weka integration in Spring Boot REST WS application, "weather" prediction example is realized. As a result, REST WS return list of accuracies for four prediction algorithms with the same used data. Achieved prediction results and REST WS usage demonstration are presented in Fig. 6 by calling next URL:

http://localhost:8080/wekaTest

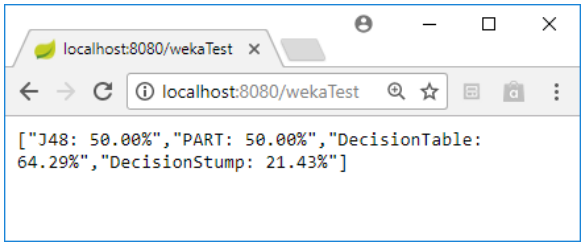

Fig. 6. *REST WS with WEKA support usage demonstration*

CONCLUSION

As shown, Spring Boot application is running on localhost on port 8080 and created REST WS is available in "/wekaTest" URL mapping.

Presented results demonstrate successful integration of WEKA library to Spring Boot application. Developed platform is suitable for usage in various type of application that need to have some artificial inelegance involved. Realization in form of a REST WS has many advantages, since it could be used by various types of clients.

ACKNOWLEDGMENT

The work presented in this paper was funded by grant TR32043 for the period 2011-2017, by the Ministry of Education, Science, and Technological Development of the Republic of Serbia.

REFERENCE

[1] R. Johnson, J. Hoeller, A. Arendsen, T. Risberg, and D. Kopylenko, *Professional Java development with the Spring Framework*, vol. 2005. 2005.

[2] F. Gutierrez, "Spring Boot, Simplifying Everything," in *Introducing Spring Framework*, 2014, pp. 263–176.

[3] D. R. Prasanna, *Dependency Injection: Design Pattern using Spring and Guice*. 2009.

[4] F. Steimann, "The paradoxical success of aspect-oriented programming," *ACM SIGPLAN Not.*, vol. 41, no. 10, p. 481, 2006.

[5] M. Keith and M. Schnicariol, *Pro JPA 2: Mastering the Java™ persistence API*. 2010.

[6] M. Deinum and K. Serneels, *Pro spring MVC: With web flow*, vol. 9781430241560. 2012.

[7] F. Gutierrez, *Testing with Spring Boot*. 2016.

[8] I. H. Witten, E. Frank, and M. a Hall, *Data Mining: Practical Machine Learning Tools and Techniques (Google eBook)*. 2011.

[9] R. R. Bouckaert, E. Frank, M. a. Hall, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "WEKA---Experiences with a Java Open-Source Project," *J. Mach. Learn. Res.*, vol. 11, pp. 2533–2541, 2010.

[10] M. A. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *SIGKDD Explor.*, vol. 11, no. 1, pp. 10–18, 2009.

[11] E. Frank, M. Hall, L. Trigg, G. Holmes, and I. H. Witten, "Data mining in bioinformatics using Weka," *Bioinformatics*, vol. 20, no. 15, pp. 2479–2481, 2004.

[12] O. Ivanciuc, "Weka machine learning for predicting the phospholipidosis inducing potential.," *Curr. Top. Med. Chem.*, vol. 8, no. 18, pp. 1691–1709, 2008.

[13] A. McGregor, M. Hall, P. Lorier, and J. Brunskill, "Flow clustering using machine learning techniques," *Passiv. Act. Netw. Meas.*, pp. 205–214, 2004.

[14] Y. Wang, I. V. Tetko, M. A. Hall, E. Frank, A. Facius, K. F. X. Mayer, and H. W. Mewes, "Gene selection from microarray data for cancer classification - A machine learning approach," *Comput. Biol. Chem.*, vol. 29, no. 1, pp. 37–46, 2005.

[15] A. C. Tan and D. Gilbert, "Ensemble Machine Learnign on gene expression data for cancer classification," *Appl. Bioinformatics*, vol. 2, pp. 1–10, 2003.

[16] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proceedings of the 24th international conference on Machine learning - ICML '07*, 2007, pp. 935–942.

[17] K. Lee, J. Caverlee, and S. Webb, "Uncovering social spammers," in *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval - SIGIR '10*, 2010, p. 435.

[18] A. Mannini and A. M. Sabatini, "Machine learning methods for classifying human physical activity from on-body accelerometers," *Sensors*, vol. 10, no. 2, pp. 1154–1175, 2010.

[19] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, p. 5, 2006.

[20] A. A. Freitas, "A survey of evolutionary algorithms for data mining and knowledge discovery," *Adv. Evol. Comput.*, pp. 819–845, 2003.

[21] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Using machine learning techniques to identify botnet traffic," in *Proceedings - Conference on Local Computer Networks, LCN*, 2006, pp. 967–974.

[22] L. Devillers, L. Vidrascu, and L. Lamel, "Challenges in real-life emotion annotation and machine learning based detection," *Neural Networks*, vol. 18, no. 4, pp. 407–422, 2005.

[23] R. J. McQueen, D. L. Neal, R. E. DeWar, S. R. Garner, and C. G. Nevill-Manning, "The WEKA Machine Learning Workbench : Its Application to a Real World Agricultural Database," *Proc Can. Mach. Learn. Work.*, 1994.

[24] "WekaWiki Related Projects." [Online]. Available: http://weka.wikispaces.com/Related+Projects.