vaadin}>

Migration Assessment

Annotated Table of Contents

The Migration Assessment consists of three main parts. The first part covers the current situation of the existing application, the second part examines the anticipated situation of the migrated application and the constraints imposed, and the third part proposes the transformation solution.

OVERVIEW

Part I: Technical findings of the existing code

In this part we review the physical state of the existing code, analyze dependencies by their occurrence in almost all syntactic constituents of the Java language, and consider components, libraries, and cross-cutting concerns.

1 Physical state of the inventory

This section reviews the source files and their size, their organization into projects, and in which of these projects the dependencies exist. Here we can find out how many dependencies each project contains, validate assumptions about certain packages being unaffected by the migration, and see what portion of the inventory we can completely ignore. We also review the binaries that are used by the code and we can rank them by frequency of use. Pom files give us an indication of which dependencies might be present but whether the JARs are used heavily, barely, or not at all we can only tell from the code. The Vaadin static tooling will show us the logical/physical mappings so we can analyze the frequency of dependencies on physical JARs by their related logical Java namespaces.

2 Unreachable code

Just because code is there, doesn't mean that it gets executed. Migrating code that never gets executed often serves no purpose. Understanding how much of the code can be left out of consideration for the migration due to being unreachable or dead, is key to interpreting the dependency data. To get an idea of how reliable the data is we make a count of the dependencies occurring inside unreachable code blocks.

3 Dependencies by syntactic constituent

Syntactic constituents are a set of chapters aimed at understanding all the places where dependencies occur, grouped by Java node type. Typically, by far the majority of the dependencies on the legacy APIs stem from method invocations. The analysis goes further than this however to see also where types from the legacy libraries were used in type arguments, cast expressions, type hierarchies of class declarations, return types of expressions, instantiations and injections, catch clauses, qualified name constants and variables accessed.

4 Libraries and extensions

This section has all the information related to custom components found in the application. These might be third-party components (in the case of Vaadin applications these could be hosted in the Vaadin Directory), or components created by the customer but kept in a separate project. The section takes a micro view on the options for each component identified. We also look at JARs that aren't related to visual components, like JSoup, Selenium, or Spring, of commercial components for which the substitution might have licensing consequences, and sources that might be in other languages.

5 Cross-cutting concerns

Here we consider the concerns that transcend classes in isolation. We look at the UI and Unit tests with an eye on reusing or transforming in order to test the new migrated application and validate feature parity with the existing one. We look at styling, selectors in CSS, and programmatically applied class names, navigation, drag and drop, localization, or other concerns that would be relevant.

Part II: Migration goals

In this part we answer the "where do you want to be" question, focus on what the migration target is, and consider constraints.

1 Leverage code from the existing application

Here we review each category of artefact (Java, HTML, CSS, Unit Tests etc) and review the reusability of each in the target architecture.

2 Interview results

Here we include the answers the customers provided in the interview, extended with notes taken during the interview by our experts

3 Impact of migrating to Vaadin on licensing

Here we review the impact on licensing of moving to the target stack

4 Impact on building, troubleshooting and debugging

This section reviews key changes impacting developers and the way they build their application. These may relate to build engines, required components and the way they debug.

5 Legacy browsers

Any information related to legacy browsers that are non-compliant with the latest W3C standards (mostly IE11 and EdgeHTML) will be included here.

6 Customer questions

Here we provide answers to any open questions the customer has raised to the assessment team, typically asked during the interview.

Part III: Process

In this final part we answer the question of "how will you get there?" We define the migration target and key mappings, develop a technical plan and estimate the effort of completing the migration successfully.

1 The case for code reuse

In this section we evaluate how, based on lines of code, empirical frameworks like COCOMO can be used to predict how much it would cost to redevelop the application from scratch.

2 Considerations for migration scenarios

Here we look at any considerations - options the expert team have come up with during the assessment and that can help shorten time to market, reduce costs or risks, or accelerate the early delivery of value to the application's users.

3 Considerations for a phased migration

This section considers phased approaches and the tooling options available that will make the phased approach possible. For Vaadin applications these would be the MPR tool, for applications on other stacks, other tools have to be found.

4 Testing and acceptance

Here we review unit and UI tests and evaluate how they can be reused to test the new application, what is known of the testing procedures and how they will be adapted to test a potentially large amount of new code to make it production-ready.

5 Notes on data binding

While the bulk of the migration may concern UI components, the migration will inevitably affect how the application gets data into and out of the components and how the data is converted between different formats and validated. These topics are explored in this section on data binding.

6 Technical risks

Here we list the known technical risks linked to the migration of this application. The risks are not limited to any specific area and can include runtime conditions that cannot be known from the code, or other side effects.

7 Migration of components

In this section we list all classes that can be considered components, including their containers. The information is structured in a mapping table that categorizes the components into groups, shows the frequency of use, and what the replacement strategy is for each.

8 Migration of theming

Here we look at recommendations for theming, including exploring whether it would be useful for transforming or starting over with a blank slate.

9 Replacement estimates

In this section we review the estimates, as a continuation of the tables presented in Chapter 7. We then aggregate the estimates into a work breakdown structure that gives an impression of how a project plan, with actual resources and milestones, might look.

10 Leveraging Vaadin to boost project success

In this last section we explore possibilities to boost the productivity of your team with external experts. Here we look for parts of the migration that are more purely technical, can be isolated from the rest of the work, and do not require domain knowledge.

Want to learn more about our migration assessments services?

CONTACT US