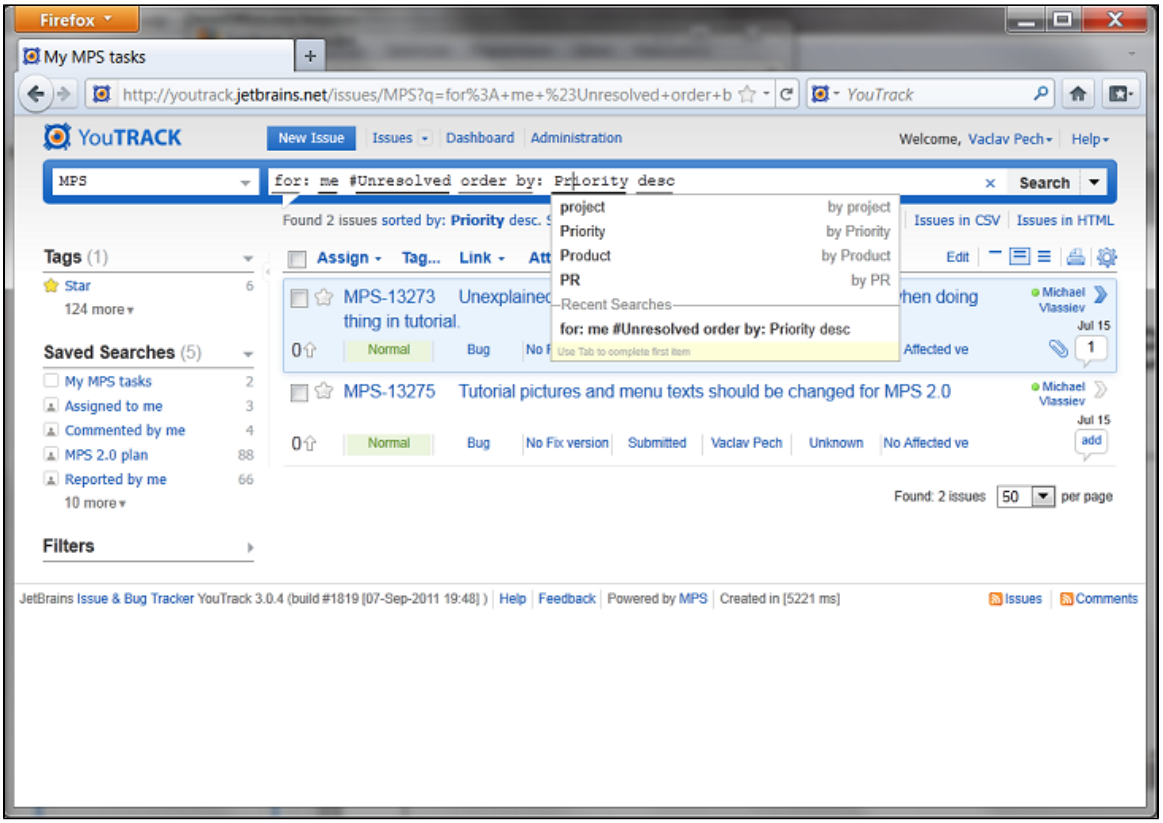
YouTrack MPS case study

A case study of JetBrains YouTrack use of MPS

Valeria Adrianova, Maxim Mazin, Václav Pech

What is YouTrack

YouTrack is an innovative, web-based, keyboard-centric issue and project tracking system by JetBrains, the creators of popular IDEs and team cooperation tools. IntelliJ IDEA, ReSharper and TeamCity are the most prominent represents of the JetBrains toolset family of products. YouTrack has been designed to help software developers and their teams to better perform their issue-handling tasks and keep track of their projects with pleasure and ease. External users, who need to report defects or suggest improvements to the team, can do that in a snap and with minimal steps through YouTrack.
YouTrack's user interface is based on two simple, keyboard-driven controls: a search box and a command window. In the YouTrack search box, users enter Google-like search queries instantly aided, as they type, by YouTrack's suggestions and highlighting. Powerful YouTrack commands help them modify multiple issues simultaneously, again solely with the use of their keyboard. YouTrack is a highly customizable and flexible issue tracking system. The set of tracked issue attributes along with their values can be completely customized to fit any project needs. Administrators can create custom workflows to define the life-cycle of the tracked issues tailored to the project's specific process needs.
Please visit www.jetbrains.com/youtrack for more details.



What is MPS

Meta Programming System (MPS) is a brand new concept of software development environment implementing the Language Oriented Programming paradigm. It was specificaly designed to make it easy to combine general-purpose and domain-specific languages within the same codebase. On top of that existing languages can be extended and customized by developers to better match their needs on specific projects that they face. MPS shifts the level of abstraction at which developers operate by making the programming languages themselves customizable and

mutually composable.
MPS is an open-source project available under Apache 2.0 license and has already been adopted by several commercial and academic projects. Please visit http://www.jetbrains.com/mps for more details about MPS.

MPS differentiates itself from other language workbenches by avoiding the text form. Your programs are always represented by an AST. You edit the code as an AST, you save it as an AST and you compile as an AST. This approach allows you to avoid defining grammar and building a parser for your languages. Instead, you define your languages in terms of types of AST nodes and rules for their mutual relationships.
Non-dependency on parsers eliminates ambiguity and thus makes it possible to combine languages even if they have been designed independently and without any knowledge of one another.
The strength of this approach becomes particularly noticeable when you compose languages with very different notations. For example, embedding decision tables or state machines into general-purpose Java or C code.

```
public void run(string[] args) {
  map<string, Object> person = this.createPerson();

  int discount;
  discount = int Default: 0
```

	isMale(person)	*isFemale*(person)
isBaby(person)	100	100
isChild(person)	50	50
isAdult(person)	5	10
isRetired(person)	20	10

```
  ;

  System.out.println("Your name: " + person["name"]);
  System.out.println("Your discount: " + discount);
}
```

How YouTrack uses MPS

YouTrack is a typical web application running Java on the server and JavaScript on the client. The two parts of the system communicate using AJAX calls transferring JSON data across the wire. The server-side part then manipulates persisted data through a home-brew NoSQL database.

YouTrack leverages MPS in several ways. First, a web-application development language layer, called Webr, was build to provide abstraction hiding the details of the web infrastructure. Views are modelled through hierarchical builders, AJAX calls are hidden behind plain method calls, the communication gap between the client-side and the server-side code has been bridged by transparent Java and JavaScript code generation.

The key characteristics of Webr:

- A smooth mix of markup and strongly typed java in templates to specify the view aspect
- High level components are translated automatically into Html and JavaScript code
- Binding html-component to expressions that will calculate the displayed values
- Strict references to the view components in client/server-side event handlers.
- Ability to call server-side methods from the client-side, and vice versa we can execute browser javascript from the server side.
- Automatically map URLs to the root templates.

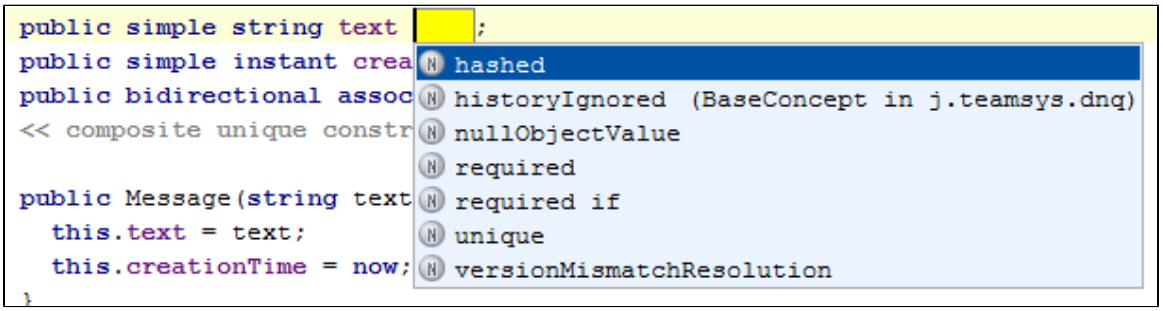
```
root html template Board(string login) id parameter: login layout Root

anonymous access true
allow access if  !("guest".equalsIgnoreCase(this.login))
stream output    predefined
```

Second, the persistence layer is modelled through a set of languages, called DNQ. These provide efficient collection-like API for intuitive, yet efficient querying. Relationships among persistent entities can be easily expressed through dedicated DSLs, just like validation constraints, triggers, versioning, transactions and other aspects of application persistence.

The key characteristics of DNQ:

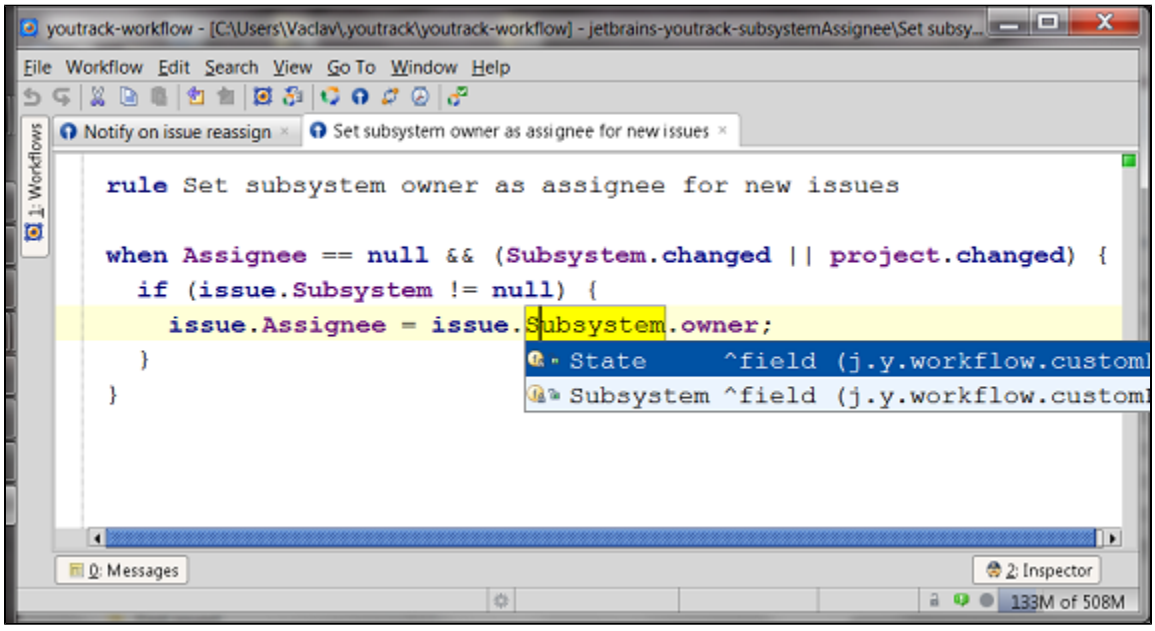
- Persistent classes inherit their behavior and capabilities from ordinary Java classes and impose no limitations on their use in the application.
- Properties of persistent classes can be constrained through annotations.
- Transactions in Java code are demarkated through a dedicated "transactional" code block
- Generation of weakly typed target code from strongly typed input code. As a result we get less footprint in PermGenSpace and easier hot deploys.
- Generator overriding for collection language operations. We generate optimal db-code when it is possible, instead of applying the expensive Java-collections operations.

- Partially typed JavaScript with explicit classes to ease development of the client-side code.
- Automatic JavaScript dependencies calculation.
- Language injection allows the developers to type structured html/jquery code inside string literals.
- Variables in css files allow developers to represent repeating values and thus avoid unnecessary duplication.
- API wrapping - a DSL was created on top of Jersey (a library for building RESTful APIs) to provide syntax that is more natural for the persistence domain than Java with annotations.
- Similarly, the JAXB API was wrapped with an intuitive DSL to allow for XML notation to be used for instructions specifying Java to xml/json translation.



The workflow definition language

Third, YouTrack comes with a visual administrator console that allows the server maintainer to manage workflow rules applied to various events. The administrator defines the workflow rules through a dedicated declarative DSL. These rules then get generated into JavaScript functions that get stored on the server and executed when the triggering conditions are met.



Key benefits of choosing MPS

Using DSLs instead of frameworks embedded into a general-purpose language comes with many benefits. A lot of the features in YouTrack would be much more difficult to achieve with a traditional approach based on a general-purpose language extended through frameworks and libraries.

- With the view-to-model bindings we can bind an html-component to a left-value expression. When the html-component is being rendered, the value of the expression becomes the initial value for the html-component. We then generate a closure that assigns the value that comes from the client side to the same expression. Such manipulations with expressions would be close to impossible without code generation.
- The ability to call server-side methods from client-side code greatly simplifies development. We can call Java methods that sit on the server directly from the JavaScript code running inside the browser. All the RPC communication is generated.
- Last but not least, the DSL approach allows YouTrack to benefit from declarative URL mapping definitions. URLs can be mapped to the root templates, the name of which becomes a path of the URL by which this template is accessible. Parameters of the template are then automatically mapped to the query parameters of the URL.

Challenges along the way

Language design

Although it is technically very easy to create new language constructs in MPS, comparable, for example, to creating a new class in Java, the actual design of new languages brings about a lot of challenges. Development of a good language is no easier than building a good framework. The whole domain is rather new and still lacks codification of good practices and patterns. So most of the time we had to learn from our own mistakes. The good think is, however, that thanks to the MPS tooling support we can easily change languages and build automated refactoring scripts, whenever we discover a flaw or a deficiency in the language design. Our languages can evolve gradually to meet our needs as we explore the domain and adapt to the changes that come along the way.

Learning curve

MPS introduces several new concepts, which require some time for the new developers to become familiar with them. In our experience it took about two to three weeks for the newcomers to gain speed with MPS. After that period they were able to fully benefit from the advantages of language-oriented programming and deliver value to the project.

Future directions of YouTrack and how will MPS support them

Stateless webr

In YouTrack we experienced several scalability problems related to maintaining state in the web layer. We decided to build a stateless variant around our Jersey-based REST language, which will handle and map http requests. For this to happen, we will need a new strictly typed template languages for html markup, strictly typed JavaScript for client side behavior and the REST language to codify communication between the client and the server.

From our previous experience we've learned three lessons:

- Keep generators simple: the language constructs should be translated to the target code in as straightforward a way as possible. Smart algorithms do not belong into the generator.
- Follow the high cohesion and low coupling design pattern for languages. It's wise to avoid building all-including monster languages as well as wiring a large set of tiny languages with complex dependencies between them. We now follow these guidelines when designing our new language set.
- Do not add new constructs when you can extend existing ones. For example, we have components in stateful webr, which are essentially html elements with additional behavior. We created an additional concept that references an html element declaration and provides some options to configure it. This, however, gives us two choices for every html element in the completion menu and it is not clear in which cases we should use one or the other. In the new stateless webr that we are building, the new options are instead added to the existing html tags directly.
  Also, when you write html code you mix it with css definitions (e.g. by referencing to a css class). In stateful webr we have a special attribute, named *class* that allows you to reference a particular css class. What we've done in stateless webr instead, is we've added a concept that extends the text of an attribute. This concept references a css class and can be used only for the *class* attributes. So the user can choose between using plain text or a reference to a css class as the value for the *class* attribute.

Workflow editor

The workflow editor is an MPS based tool for YouTrack that allows system administrators to manage rules for issue workflow. Using a dedicated standalone tool they can define workflows in a high-level MPS-based DSL. We expect to improve the editing experience of the tool in cooperation with the MPS team to come closer the natural feel of plain-text editors. Additionally, we would like to distribute the workflow editor as an IntelliJ IDEA plugin.

Releasing to the public

At some point the set of languages we used to develop YouTrack will be released to the public, because we believe there are many teams and projects out there who may greatly benefit from our innovatove approach to web application development.

Summary and general recommendations

Ruby on Rails paved the way towards DSL-based web application development. The idea of using languages tailored to the specific needs of different aspects of a web application quickly spread across the industry. Frameworks like Grails, Play, Django, Lift and others appeared, which brought the RoR principles to their respective communities. YouTrack's Webr-DNQ is a web framework built around the very same principles. Being built in MPS, the languages in Webr-DNQ can each model their particular domain without much syntactical limitations, yet all together they form a coherent, well co-operating group of languages. The code generation aspect of MPS enables boilerplate code or repetitive functionality to

be added automatically during compilation, leaving the original source code short and focused. All these capabilities come without loosing compile-time type-safety, IDE assistance nor code analysis support during development. These can be considered to be the crucial benefits of the chosen technology stack.