

- ✓ **Better forms.**
- ✓ **Better checklists.**
- ✓ **Fewer custom fields.**



# Better Form Design in Jira

by Jennifer Choban and Rachel Wright





About us	3
Form Design Matters	4
Layout and Flow: User-Friendly Forms in Jira	12
Good Form Questions Part 1: Open Questions	20
Good Form Questions Part 2: Choice and Conditional Logic	27
Things to Think About When Converting Forms to Jira	36
Efficient Jira Screens and JSD Forms	41
Tips for Creating Good Jira Forms and Screens	48



## ABOUT THINKTILT

ThinkTilt was founded to provide powerful support tools that give teams like yours power over your business' processes.

Our premiere product, **ProForma**, is an app for Jira, and is available through the Atlassian Marketplace, both as a fully featured free version, called ProForma Lite, and an unlimited version.

ProForma makes it easy for business teams to build and deploy user-friendly forms with custom forms and fields, backed by Jira's great workflow engine. Empower every team in your organization to take control of their processes and deliver first class request management. All the information you need, where you need it.



## ABOUT RACHEL WRIGHT

Rachel Wright is an entrepreneur, process engineer, and Atlassian Certified Jira Administrator.

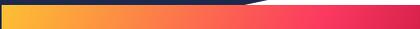
She is the owner and founder of Industry Templates, LLC, which helps companies grow, get organized, and develop their processes.

Rachel also uses Atlassian tools in her personal life for accomplishing goals and tracking tasks. Her first book, the "**Jira Strategy Admin Workbook**", was written in Confluence and progress was tracked in Jira!

*We challenge you to think of a type of team that wouldn't benefit from Jira!*



# Form Design Matters



*By Jennifer Choban*

Forms and screens do more than collect data. Learn how form design can enhance efficiency, improve customer relations and make the world better.



Let's just admit it. Forms aren't sexy. They aren't exciting or innovative. Few people would call them fun. They are, however, important and few business tools have proven to be as enduring.

The move from paper to electronic forms, and then to online forms solved a lot of problems. We no longer have to contend with illegible hand-writing and necessary fields that were left blank. But the advent of online forms also presents a challenge. How can we best use the tools we have - validation, conditional logic, auto-formatting - to truly optimize our forms so that teams get the information they need and users have a painless, possibly even pleasant, experience?

Forms and screens that serve as electronic forms (I'll be using the terms *forms* and *screens* interchangeably) are more than just vehicles for data collection. They are the first step that gets the ball rolling on most business processes. They are also one of the primary ways we interact with our internal and external customers. And they are a mechanism for influencing (whether we want to or not) human behavior. Combining forms with a powerful workflow engine like Jira is a great way to get work done. Ensure that your Jira forms are well-designed and you'll do more than get work done. You'll save resources, improve relationships and influence user actions in the right direction.



# Your Jira Forms Drive Data Quality

We create forms for collecting data, and the quality of the form will go along way towards determining the quality of the data. With the creation of every issue type and every request type, you are making decisions - which fields to include on which screens, what order to list the fields in, what justifies the creation of a new custom field - that will impact the quality of data you receive. In turn, getting better data will allow you to:

- **Provide better, faster service** - Imagine if all of your Jira Service Desk agents could provide one-touch service, receiving all of the information they need on the initial request and responding to the customer to let them know that issue has been resolved. There's no going back and forth to collect the missing pieces. No digging through comment chains, no trying to parse out multiple data points that have been lumped together in a description field - all of the needed data is there, where agents can find it. Requests move through the queue faster, pleasing customers and allowing agents to focus on higher value work.

**WHICH REQUEST WILL GET BETTER RESULTS?**

Raise this request on behalf of

---

Summary

Description (optional)

Hi,  
I have a new hire, Sherry Wilson, who will be starting on Monday, taking Diego's old position. Hope we can have everything ready!  
Thanks

Sherry Wilson

Office/cubicle number

Start date

**Hardware needed**

Order new laptop  
 Order new laptop  
 Order new desktop  
 Set up existing laptop  
 Set up existing PC

Standard software configuration  
 Yes  
 No

Any specifics

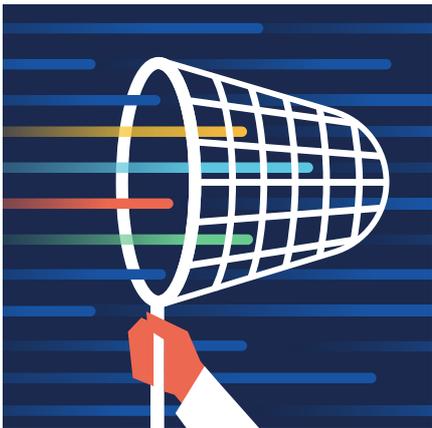
**Communications**

Set-up needed  
 Email  
 Smartphone  
 Install new landline  
 Redirect existing landline



- **Refine your processes**

We often create our processes to accommodate the limitations of what we know or what our software can do. Jira is super flexible, but configuration can become complex. Using well-designed forms that embed in Jira issues allows you to reduce the number of issue types (the work of capturing specific pieces of data is shifted from the issue to the form), simplify workflows (use checklists instead of additional statuses or subtasks) and prune back that jungle of Jira custom fields. Your Jira instance becomes less complex, and your Jira administrator less burdened. Shifting from custom fields to form fields will also improve Jira performance.



- **Improve the Bottom Line**

What is the cost of bad data? How much time do service agents, Jira users and customers lose searching for information that's hard to find, invalid or missing? Include a field to capture a document version number on a form and the agent can glean the data at a glance. Obtaining the same information through a comment can cost hours, as the request sits waiting for the customer to come back with needed data. Later, when the agent goes to retrieve the information combing through a comment chain can be like going down a rabbit hole. Better forms lead to better productivity.

- **Enhance Compliance**

Forms can serve as a record, capturing and preserving data from different sources, at different points in time, enhancing compliance and accountability. A well-designed form will make it easy to provide management with useful reports, and auditors with the information they need. Adding multiple copies of a well-designed form to a Jira issue allows you to see not only the current state of an issue, but also the discoveries that have been made along the way.

**Multiple copies of the same form documents changes over time.**

This screenshot shows a Jira issue interface. At the top, there are action buttons: Attach, Create subtask, Link issue, and a menu icon. Below this is the 'Description' field with the text 'Company website is not formatting correctly.' and a 'Description' section with a placeholder 'Add a description...'. The 'Environment' section is set to 'None'. The 'Forms' section lists several attached forms: 'Bug Checklist' (SUBMITTED), 'Bug Checklist' (SUBMITTED), 'Bug Checklist' (SUBMITTED), and 'Bug Report' (LOCKED). Each form entry has a three-dot menu icon to its right. An 'Add Form' button is visible at the bottom right of the forms list.

This block contains three overlapping screenshots of a 'Bug Checklist' form, illustrating how the form's state changes over time. The form is titled 'Bug Checklist' and has a 'Viewing Form' header. It is divided into several sections: 'Key Steps' with radio buttons for 'Initial Investigation (Support)', 'Fix Complete (Development)', and 'Release Check (Q/A)'; 'Investigation' with a 'Can you reproduce the bug?' question and 'Yes/No' radio buttons; 'Additional Information' with a text area; 'Issue Summary' with a text area; and 'Known Issues / Limitations' with a text area. The first screenshot shows the 'Initial Investigation' step selected. The second screenshot shows the 'Fix Complete' step selected. The third screenshot shows the 'Release Check' step selected and includes a 'Browsers tested' section with checkboxes for Chrome, Firefox, Safari, Edge, and Internet Explorer, where 'Chrome' is checked.

# Forms are the Basis of Relationships

A form can be the start of a beautiful friendship. Or not. When you consider how many business processes start with a user filling a form, it's amazing that more attention isn't given to form design as part of managing customer relations.

Most business processes start with a form. That means that the form is often your one and only chance to make a first impression. In some cases, government bureaucracies come to mind, a form or forms may be the only way customers interact with an organization. If the forms are bad, the relationship won't be good. Bad forms can result in users postponing, abandoning and sometimes bypassing key processes.

On the other hand, well-designed, intuitive, user-friendly forms can reduce friction, build rapport and help set appropriate expectations. The improved service that results from good data collection will inspire confidence. Your forms build your reputation and promote your brand.



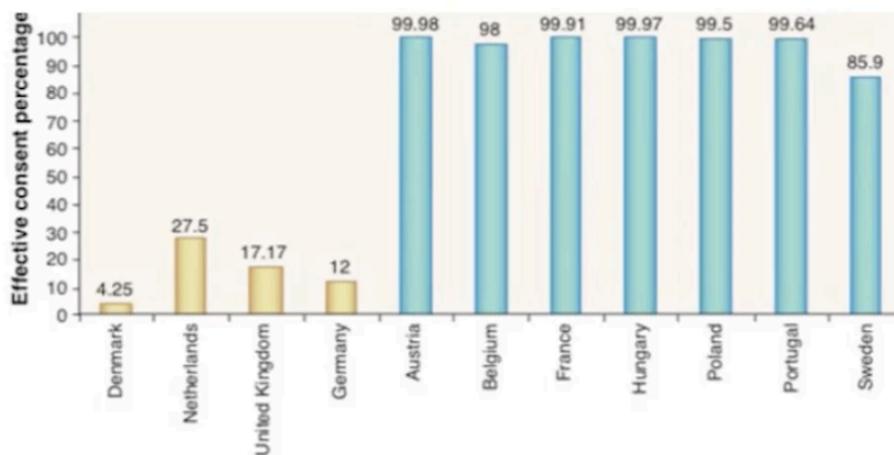
*A form is often your one and only chance to make a first impression.*



# Form Design Influences Behavior

How we design our forms – the words we choose, the presence or absence of default values, the order of choice options – influence users decisions. If that sounds far fetched consider the image below illustrating organ donation participation in Europe:

## Organ donation enrolment in Europe



The stark difference between the rates for the countries listed on the left and those listed on the right isn't due to culture or religion. It's the difference between opting in and opting out. Want to increase employee participation your retirement contribution program? Change the form? Want to see more customers opt for a higher value product? Change the form.

We'll be discussing how you can use wording, question framing, defaults and choice architecture to influence user choices in a later chapter. And if you're thinking, "Wait, that's manipulative. I don't want to do that," we'll also discuss how you can mitigate the influence of your questions on users choices (though it's not possible to be 100% neutral).

# Where Do We Go From Here?

Once you're convinced that form design matters, what's the next step? Start by reframing how you think about the form process.

The [Behavioral Economics Team of the Australian Government](#) recently introduced the WISER framework for form design. It lays out specific steps to go through in identifying the form's audience, providing appropriate instructions, using a clear layout, etc.

## WISER: A framework for improving government forms



	W	I	S	E	R
	WHO	INTRODUCTION	STRUCTURE	EXPRESSION	REPEAT
	Understand your clients and the process.	Focus on page 1. Offer clear instructions, highlight key info.	Structure the form simply and guide people through it.	Write for clients, not government. Use plain English.	Test and iterate.
STEPS	<ul style="list-style-type: none"> <li>Map the overall process: identify each step your clients need to undertake</li> <li>Gather data on the form and clients: eg: completion rates, demographics</li> <li>Conduct focus groups and user testing</li> <li>Complete a form audit from your client's perspective</li> <li>Identify where, when and how clients receive and lodge the form</li> <li>Identify key friction points in the process and the form</li> </ul>	<ul style="list-style-type: none"> <li>Write a title that makes sense to clients</li> <li>Clarify eligibility and purpose</li> <li>Highlight why people should complete the form</li> <li>Tell them what they need on-hand to complete the form</li> <li>Use boxes and bold sparingly to highlight key information</li> <li>Remove technical information, or consider appendices</li> <li>Put key information first</li> <li>Advise how to submit</li> <li>Use checklists to signal actions</li> <li>Personalise if possible</li> </ul>	<ul style="list-style-type: none"> <li>Order sections logically</li> <li>Make it visually attractive</li> <li>Use formatting cues to signal alternatives</li> <li>Group into common themes</li> <li>Remove duplication</li> <li>Consider defaults</li> <li>Use navigation prompts</li> <li>Ask 'Is this necessary?'</li> <li>Move legal notes to end</li> <li>Consider online best practice                             <ul style="list-style-type: none"> <li>Prefilling</li> <li>Active choice</li> <li>Sequencing</li> <li>Mobile responsiveness</li> <li>Positive error messages</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>Write in plain English</li> <li>Remove jargon and legal jargon</li> <li>Keep sentences short</li> <li>Keep tone direct, calm &amp; understated</li> <li>Consider framing</li> <li>Aim for year 7-8 reading level</li> <li>Use active voice, not passive</li> <li>Use 'you' and 'we'</li> <li>Ask single-issue questions</li> <li>Make statistics tangible</li> <li>Read it out loud to ensure it's clear</li> </ul>	<ul style="list-style-type: none"> <li>Implement the new form in a way that reduces friction</li> <li>Measure success of re-design                             <ul style="list-style-type: none"> <li>Completion rates</li> <li>Data quality</li> <li>Policy outcomes</li> <li>Focus groups and feedback</li> </ul> </li> </ul>
BI CONCEPTS	<ul style="list-style-type: none"> <li>Humans vs econs</li> <li>Friction costs</li> </ul>	<ul style="list-style-type: none"> <li>Cognitive overload</li> <li>Saliency</li> <li>Friction costs</li> </ul>	<ul style="list-style-type: none"> <li>Choice architecture</li> <li>Saliency</li> <li>Friction costs</li> </ul>	<ul style="list-style-type: none"> <li>Cognitive overload</li> <li>Friction costs</li> </ul>	<ul style="list-style-type: none"> <li>Choice architecture</li> <li>Friction costs</li> </ul>

Behavioural Economics Team of the Australian Government

behaviouraleconomics.pmc.gov.au

There are two big things you should notice about the WISER framework. First, the framework focuses on the user's point of view. Who are the users? Why should they complete the form? What information will they need? What language will they understand?

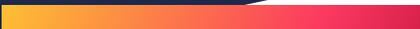
Second, the framework tells us to test and repeat. You wouldn't adopt a "set it and forget it" attitude towards developing software, so don't do that with your forms either. Form design needs to be iterative.

So now that we know that, where do we start? In the next chapter we'll discuss the layout and flow of good forms.



# Layout and Flow

## Creating User-Friendly Forms in Jira



*By Jennifer Choban*

Make your Jira forms more user-friendly by following the principles of form design.





There are two things we know for sure about forms – One: everyone fills out forms, and two: no one likes filling out forms. With that in mind, your goal in designing the layout of your forms should be to help the user get through the form quickly and easily while providing accurate information.

With multiple options for label placement, help text placement, validation, action buttons and page layout – webforms provide multiple ways to pose questions to users. For the purpose of this article however, we'll be focusing on Jira forms. We won't get into the weeds about whether or not your labels should be above or to the left of your input fields, because Jira doesn't offer options on all of those details.

Instead, we'll focus on three big principles – **minimalism, logical structure, and a clear path to completion** – that will ensure your forms (be they Jira screens or ProForma forms) are as user-friendly as possible. Rather than thinking in terms of what information we need, we will look at the problem from the user's point of view. Who are your users? What are they trying to achieve by filling out the form?

Let's get started.

# Principle 1: Minimalism

Less is more when it comes to form. Our brains are subject to cognitive overload. If too much comes at us at once, we're unable to process it. We don't have the RAM. So keeping our forms decluttered makes it easier for users to complete them.

## Reduce the number of questions you ask

The first thing you can do to make sure your forms are as minimal as possible is to prune back the number of questions you ask. To ensure you're only asking what you really need to know, either start from a blank slate, or – if you're working from a template – put every question on trial by considering:

- How is this information used?
- Is it necessary in order to provide the user with what they want?
- Do we already have this information somewhere else?
- Can we use conditional logic to ensure that the questions is only shown to the right users?
- Are details likely to have changed since the last submission? If not can they simply check a box 'Unchanged' and avoid providing the same information time and again.

If you don't need the information, don't ask for it.



*Since no one likes filling out forms, we can demonstrate respect for our users by making the forms as brief, easy to understand, and fast to fill out as possible.*



## Simplify the Language of your Question

You'll also want to be minimal in how you ask your questions. We'll be delving into the details of question types and wording in the next article. For now, suffice it to say that your question labels should be concise. Use just as many words as needed to avoid ambiguity. You'll also want to be sure that the language you use fits your audience. If the form is for customers, use plain language. If it's for technical users, you can use more technical language.

## Keep the Extras to a Minimum

Finally, keep visual extras (color, images, etc.) to a minimum. Color can be a good option for making important things stand out, but only if there's not too much of it. The same is true for icons. Other than branding, only include images if they help clarify things for the users. Reserve red text for error and validation messages.

## Make Optional Questions, optional!

You can still pose optional questions. But if it's not information you have to have to provide the service, then wait and ask it later. In *Web Form Design: Filling in the Blanks*, Luke Wroblewski cites research indicating that users are more willing to answer optional questions if they're posed after the initial form has been completed. With ProForma, you can [automatically add a supplementary form](#) when the initial form is completed. Alternatively, you can create a customer satisfaction survey in Jira to collect more information (See page 19 of Rachel Wright's *Jira Strategy Admin Workbook* for instructions on deploying a minimal customer satisfaction survey in your Jira support project.).

Atlassian has been working to reduce cognitive overload when viewing Jira issues...

Jira then,

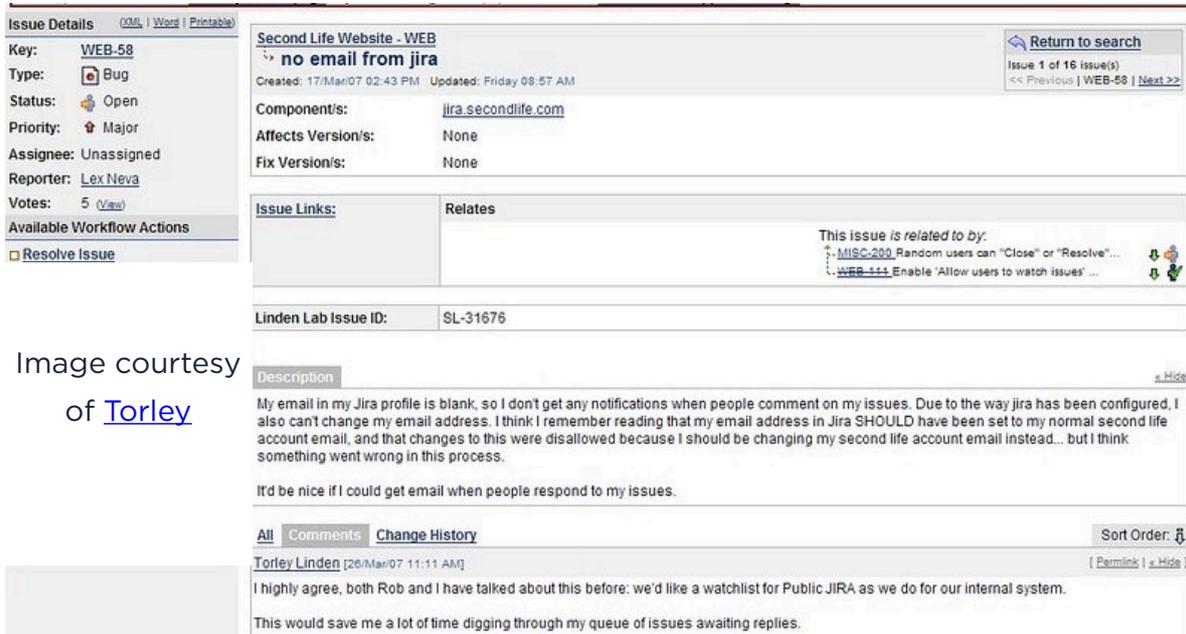
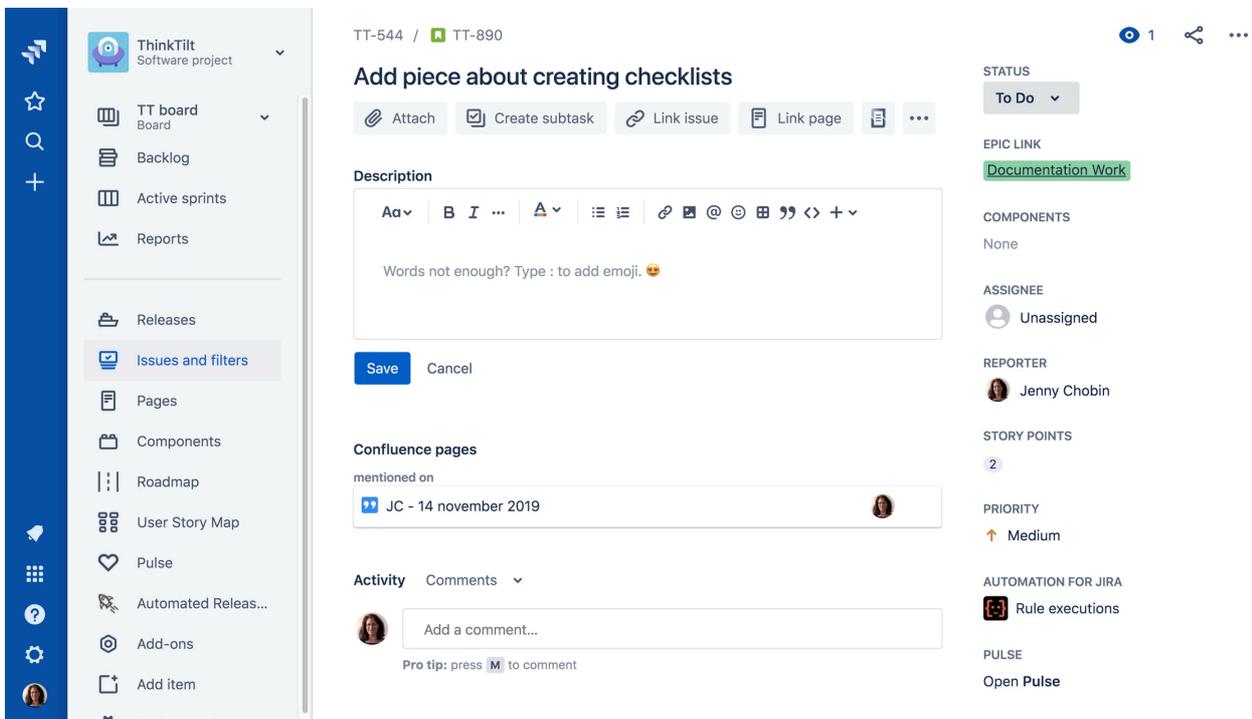


Image courtesy  
of [Torley](#)

Jira now.



## Principle 2: Logical Structure

Now that you've decided what needs to be included on your form, you have to decide how to arrange it. Your forms will be a lot more user-friendly if you think of them as one side of a conversation, rather than as a way to get info into your database.

### Group Questions into Sections

A conversation usually starts with easy, non-intrusive questions. Questions in a conversation are grouped around larger topics, and they follow each other in a natural sequence. There are also natural breaks when a conversation shifts to a new topic. You can recreate that in your forms by grouping questions into sections, and by putting those sections – and their respective questions – in a logical order. This will provide context, making it easier for the user to understand what you're asking for.

### Keep Introductory Text to a Minimum

Keep initial instructional text at the beginning of your form to a minimum. If it's a big block of text users won't read it anyway. Instead, provide general instructions in their corresponding sections, or better yet, use field-level help tips. Keeping the instructions close to the questions they refer to will eliminate confusion for the user.

There's one important exception to this rule. It's good to let people know up front what they will need, and if there are any eligibility criteria. Well designed forms frequently include a "gateway" page or instruction section at the beginning for this purpose.

### Use Pages, Tabs or Sections

Online forms often use multiple pages to organize the form into manageable parts. While we don't have this capacity in Jira, you can organize questions onto different tabs. Similarly, ProForma lets you group questions into sections, and using conditional logic you can progressively reveal more of the form to the user as they work through it. ProForma also lets you include multiple forms on an issue, making it easy to break up data collection into logical parts.

Good form practice also includes a "thank you" or acknowledgment page at the end that lets the user know the form has been successfully submitted, and advises them about the next step in the process.



## Principle 3: A Clear Path to Completion

Since we know that no one likes to fill out forms, one thing we can do to lessen resistance and procrastination is to make it clear to the user how to get to the end. There are a few strategies you can use to illustrate a clear path to completion:

### Consistency

If your team publishes a lot of different forms, use a consistent style when it comes to format, grouping questions, and the overall structure of your forms. This will make the forms feel familiar, and therefore less intimidating to your users.

### Be Upfront

If the user is going to need information that they're not likely to have in their head, let them know upfront. Having to leave a task in progress to dig up needed info is frustrating for your users. It also provides a great opportunity for the user to abandon the process all together.

### Easy to Scan

You can also help users know what to expect by putting section titles in title case. (Section titles are also an item where you may want to use color.) Drawing the eye to section titles makes it easy for users to scan, letting them know what kind of questions to expect.

### Vertical Alignment

Finally, unless you're asking for information that demands to be in a grid (e.g. budgets), or space is at a premium, keep your input fields in a straight vertical line. This makes it easy for the eye to see how to get to the end of the form. The path to completion is like a literal path – if it's straight you can see where you're going. If it's full a switch backs it's hard to know where you're going or when you'll get there.

Compare the path to completion on these two forms...

The screenshot shows the PayPal 'Send Money' interface. At the top, there's a navigation bar with 'My Account', 'Send Money', 'Request Money', 'Merchant Tools', and 'Auction Tools'. Below this is a yellow confirmation box with the text: 'Jason, please confirm this secure transaction', 'You're about to send \$37', 'To: lucky@37signals.com (a verified member)', and 'Source: \$37 from your PayPal balance (pay another way)'. Below the box, there's an 'Email' section with subject and note, and a 'Shipping Information' section with a dropdown menu for 'Ship to' and radio buttons for shipping options. At the bottom, there are three buttons: 'Send the \$37', 'Edit transaction', and 'Cancel transaction'. Red arrows point from the top of the yellow box down to the 'Send the \$37' button, indicating a clear path to completion.

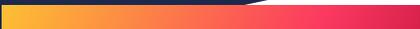
The screenshot shows the PayPal 'Check Payment Details' interface. At the top, there's a navigation bar with 'My Account', 'Send Money', 'Request Money', 'Merchant Tools', and 'Auction Tools'. Below this is a 'Check Payment Details' section with a 'Secure Transaction' icon. The 'Payment Details' section includes 'Pay To: paypal.jf@spinfree.com (a verified member)', 'Amount: \$37.00', 'Source of Funds: PayPal balance more funding options', 'Email Subject: Here's the cash I owe ya', and 'Note: Thanks for bailing me out! I also included \$7 for the cab ride. Thanks again!'. Below this is a 'Shipping Information' section with a dropdown menu for 'Ship to' and radio buttons for shipping options. At the bottom, there's a yellow box with three buttons: 'Send the \$37', 'Edit Transaction', and 'Cancel Transaction'. A red arrow points from the 'Send the \$37' button in the yellow box up to the 'Ship to' dropdown menu, indicating a path to completion.

Images courtesy of [Rosenfeld Media](#)

Creating a clear path to completion, using a logical structure and being as minimal as possible will make your forms more user-friendly and reduce the time needed to complete them. This will improve the quality of the data you receive. Of course, the real nitty-gritty detail of getting good data depends on asking good questions. We'll dive into the details of form questions in the next chapter.

# Good Form Questions Part 1

## Open Questions



*By Jennifer Choban*

Get good data in Jira by asking the right questions, the right way.



Questions, and the input fields that collect their answers, are the heart of every form. Getting your questions right, will mean getting your data right. Well designed questions also minimize frustration for the user. You'll need to decide what field types to use, what context to provide, how to word your question and how much validation to include. Think about why you're including each question. What data are you hoping to get? Then consider the user's point of view. How should you create the question in order to make it easy for them to respond with the data you're looking for?

## Open vs Closed

Questions come in two general varieties, open and closed. Open questions (text fields) let the user type in what they want to tell you. Closed questions (check boxes, combo boxes, pickers, radio buttons, and select lists) let the user choose an answer from options you provide.

They each have their advantages. Closed questions are good for users because they are easier to interpret and quick to fill out. They're good for you because data comes into your database in nice categories, with consistent formatting. Data from closed questions is easy to aggregate and report on.

Open questions are more work for the user. They have to interpret what you're asking, decide what they want to say, and then type it in. However, open questions ensure that the user has a chance to say what they want to say. The user may be looking for an opportunity to tell you something you didn't think to ask. While open questions don't make for tidy reports, they allow users to provide information which can lead to disruption, innovation and continuous improvement. If your forms use purely closed questions, you may force the user and a service agent to comment back and forth in order to get important details.

Finding the right balance of open and closed questions will depend on the purpose of your form. Consider including at least one open question on your form. This could be an overview question up front - like the Description field in Jira - or an optional "Comments" field at the end. You can also mitigate the restrictiveness of closed questions by including an "Other" option. (We'll dive deeper into closed questions in the next chapter.)

## Asking Good Questions

Teachers are fond of saying that there's no such thing as a stupid question. I beg to differ. Stupid questions are questions that don't ask what you want to know, or questions that try to ask too many things at once.

Whenever I call my credit card company I immediately get a follow-up email that asks:

Based on your call of (date), how likely are you to recommend this credit card to a friend or family member?

- Very likely
- Likely
- Unlikely
- Very unlikely

The first part of the question indicates that they want to know if I'm happy with the customer service I received, but the second part of the question asks something completely different. The question is either:

- Using the wrong metric to measure customer service;
- Or, trying to ask two different things at once.

Personally, I would never choose or recommend a credit card based solely on customer service. Other factors - interest rates, cash back and annual fees play a big role. Since I'm middle-aged my friends and family members have already found credit cards that work for them. Giving an honest answer to the question might reflect badly on a perfectly competent agent who took my call. The result is that, unless I received really bad service, I don't answer the question.

Resist the temptation to use double-barreled questions as a strategy for keeping your forms short. Yes, you want to be brief. That's why you're only asking questions you really need the answers to. But if you lump two questions into one, you won't know which part the user is answering. The agent and the user will have to exchange comments back and forth to clarify, and you'll end up taking up more of the user's time than you would have if you'd just included two questions in the first place.



## Language

How you word your questions matters. Choosing unambiguous wording will mean reduced frustration for your users, and fewer errors in the data you receive. When writing form questions:

- **Know your audience**

The language you use should fit your audience. If your form is for the general public, remember that not everyone has the same literacy level. Use [plain language](#) – short sentences and everyday words.

What if the general public isn't your audience? Reading through postings in the [Atlassian Community](#), I notice that many participants are highly technical, but may not be native English speakers. Writing for this audience, I might choose a longer, more formal word that has a latin root (thus recognizable to speakers of other latin-based languages) over a shorter, more common word. It's all about who you're creating the form for.

- **Use neutral words**

If you're truly trying to get the user's opinion, then strive for neutral wording. The words you choose can influence the answer, so avoid phrases that indicate what you think the correct answer should be. And if you're wondering what leading questions look like, grab a survey put out by a political party.

You'll also want to avoid using absolutes – words like *all*, *always*, *never*, etc. Including an absolute means you will find the exceptions that prove the rules. You can improve your questions by simply deleting those words.

- **Provide context**

Organizing your questions into [logical sections](#) will help provide context for the user. Depending on what you're asking, that may or may not be enough context. Remember that the user doesn't know what you know.

Field level help text is a great opportunity to provide more context. Including an example can help the user understand what kind of response you're looking for.

### Triggers

Example: Fire a Work Started event that can be processed by the listeners.

### Conditions

Example: Only users in the "Team" can execute this transition.

### Validators

Example: The parent issue of current issue must be in one of the following statuses: In Progress

### Post functions

Example: Assign to current user

[Rachel Wright](#) provides examples on her [Custom Workflow Documentation](#) form.

If you're asking a question that can seem intrusive, tell the user why you're asking and how the data will be used.

Finally, if you're asking for information the user may not know how to find, help them out. For example, tell them where to find a version number for their software or a serial number for their hardware. You can put this information directly in your field help text, or use your help text to point to another page with detailed instructions.

## Getting Text Fields Right

While text fields collect responses to “open” questions, that doesn’t mean that you don’t have any control of the data you receive. Using appropriate field types, formatting and validation will ensure you get data you can use, while still allowing users the flexibility they need.

### Field Size and Type

Text fields are not one size fits all. When choosing the size of your text fields (two options in Jira, three options in ProForma) remember that the size of the field should be proportional to the amount of text you want the user to input.

Text fields also come in many varieties. Along with various sizes (single line, multiple line/paragraph), there are also number fields, email fields, url fields, etc. These fields have been formatted and validated behind the scenes to ensure that the user can only enter certain kinds of information. Using the correct field type means better data that can be used for other functions later (perform calculations with number questions, export a spreadsheet of email addresses to make a contact list, etc.).

### Validation

Finally, deploying validation rules – such as word/character limits and value limits for number fields – not only ensures clean data, it allows teams to build their business rules into their forms. Save time and frustration for your users by letting them know, either in the question itself or in the help text, what the limitations are.

## Pattern Matching

You can also create your own text field types using ProForma's [Regex](#) feature. Regex allows you to create text patterns that the user's input must match. These can be simple or complex expressions, and are a great way for collecting properly formatted data. Regex patterns are often used for collecting:

- Account numbers
- Phone numbers
- IP addresses
- Currency amounts
- References to ISO standards
- Serial numbers

Note that you'll want to tell your users what patterns the field accepts. Use field level help (the Description property in ProForma) to provide instructions and examples.

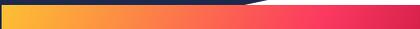
The screenshot displays the 'Edit Text Question' configuration window. At the top, a preview shows a text input field with the label 'C Number' and an example value 'C0000001'. Below this, the configuration panel includes:

- Label:** C Number
- Description:** Example: C0000001
- Style:** Narrow
- Linked Jira Field:** Do not link
- Validation:** Response required (unchecked). Fields for minimum and maximum length are present but empty.
- Format:** Must match pattern (checked).
  - Regex pattern:** ^C\d{7}\$
  - Message if invalid:** Must be the letter "C" and 7 numeric characters

Following the advice above will help make your text questions easier for your users to understand and respond to. However, chances are good that most of the fields on your forms will be choice questions, so we'll look into the best practices for creating choice questions in the next chapter.

# Good Form Questions Part 2

## Choice Questions and Conditional Logic



*By Jennifer Choban*

Choice questions provide structured data and are great for triggering automation or conditional logic.



Questions and input fields are the heart of every form. In the previous chapter, we started digging in on the do's and don'ts of creating good form questions. Now we're going to take a closer look at a particular set of field types – choice questions.

## The Power of Choice

There's a lot to be said for choice questions. Choosing from a predetermined set of options is fast for the user. They don't have to imagine what the possible answers might be, and they don't have to spend their time typing.

## Field Types for Choice Questions in Jira

There are multiple options for creating choice questions:



Determining which field type to use should be based on:

- Whether you want users to select one option only (radio buttons, dropdown, single choice select list), or more than one option (checkboxes, multiple choice select lists).
- Layout and available space on the form/screen – Radio buttons and checkboxes are fastest for users because they can see all of the choices at once, no extra clicks required. However, if you're option list is excessively long, dropdown boxes let you save space on the screen.

Since choice questions already limit the user, when choosing field types it's good practice to use the least restrictive option. Radio buttons, single select lists and dropdown boxes mean the user can only select one option. If there's any possibility that the user would want to select more than one option, use check boxes or a multiple select list.

#### Which browser do you routinely use?

- Chrome
- Edge
- Firefox
- Internet Explorer
- Safari
- Other...

New option...

*I use Chrome for work and Firefox for personal use, but the question only allows me to select one. The question should be narrowed ("Which browser do you use most often for work?") or should be a check box field type to allow the user to select more than one option.*

## Things to Consider When Creating Choice Questions

### Make Sure Everyone Can Respond

The limitation of choice questions, is that they are closed. While this may be helpful to the form owner, it can be frustrating for the user. What if what they really want to choose isn't listed as an option? One advantage of using ProForma is that it allows you to include an "Other" option, with an adjoining text field on your choice lists. Depending on how your question is phrased, it may be appropriate to offer options such as "I don't know," or "Not applicable". The important thing is to make sure that your choices offer every user the opportunity to provide an honest answer.

### Avoid Double-Barreled Questions

Since the purpose of a form is to obtain structured data, it's important that each question asks only one thing. This is especially true for choice questions. Otherwise, you'll end up with a bunch of easy-to-categorize answers and have no idea what they refer to. Customer satisfaction surveys that ask you to agree or disagree with statements like, "The agent I spoke to was friendly and knowledgeable," miss the mark. What if the agent I spoke to was kind, but clueless?



## Structure Your Rating Scales

Choice questions are often formatted as rating scales (strongly agree > strongly disagree, etc.) to measure user's opinions or satisfaction. Carefully consider if you want an even or odd number of options. An odd number of options gives the user a chance to be neutral, while an even number forces them to give an opinion. (If you're not going to make the user give you an opinion, is it really worth including the question?)

Label each point in your scale with words, so users don't have to take the extra mental step of translating numbers into another meaning. Scales should not be too large, and the increment (in meaning) between each point on a scale should be equal.

Points on a rating scale can either go from "lowest" to "highest" or vice versa. Scales are frequently structured with the most positive response listed first because users often select the first option they see. The important thing is to be consistent throughout your form.

## When a Choice Really Isn't a Choice

Sometimes we use choice questions when we're really not offering the user a choice. The user has to say yes if they want to continue the process. In these cases, a checkbox is the correct field type since we are not excluding other options. In a chapter, I said that if you want users to read your instructions, you should avoid big blocks of text. The opposite is also true: If you want your users not to read something, make it long and full of jargon. John Oliver does a great job of describing how effective this is.



# Choice Architecture

As with all form questions, you'll want to use concise and unambiguous language. But creating good choice questions requires additional considerations. How you frame your questions can impact how users respond. This is called choice architecture. You may want to use it your advantage, or you may want to try to limit your influence to get a more "pure" response from your users.

## The Influence of Defaults

One of the first decisions you'll need to make is whether or not to include defaults. Defaults save time for the user. They also nudge the user in a certain direction. Users may perceive the default options as being recommended (after all, that's what everyone else is choosing). In some cases, the default option may even be labeled as recommended. Even when logic doesn't tell us to leave the default, inertia often does. We're happy to avoid making a decision.

The most frequently cited examples of this mechanism at work are the differences in organ donation rates (cited in the first chapter) and the use of "opting in" to get employees to contribute to retirement savings account. Parting with your organs, or your money, is a serious decision. Yet the evidence shows that many of us are happy to simply follow the default's lead.

If you feel really confident that you know what the vast majority of users would want, use a default. If you want to encourage your users to make a certain choice, use a default. If you want to get more impartial data from your users, don't use a default. Be cautious about offering a default for a required field. If the field is important enough to be required, you probably want your users to consider their answers, which they are less likely to do if a default is provided.

## Choice Order Changes Results

Even if you don't use a default, some bias will be built into your choice questions simply because you have to list the options in some kind of order. Inundated with information, our brains have learned to scan, which means we'll pay more attention to the first and the last item in a list. In most cases, it's sufficient to just put your choice options in some kind of logical order (lowest to highest, alphabetical, etc.). This makes it easy for the user to find the option they're looking for. However, if you want to truly minimize the influence of the first/last bias, you need to randomize your choice options.



## How Framing & Decoy Affects Impact Responses

How your question is perceived will be impacted by the words you choose. Here's an example that frequently comes up when buying an airline ticket:

**Do you want to insure your \$789.00 ticket for \$19.99?**

Yes! Insure my flight.

No, I'll risk it.

The words have been chosen carefully. Including the price of the flight reminds the user of the investment they've already made and makes the price of the insurance seem insignificant by comparison. The word "risk" also pushes the users towards buying the insurance.

In addition to wording, visual cues can be used to nudge a user to a certain choice. Here's an example from Atlassian.

**Plan, Track, & Support**

 **Jira Software**

Plan, track, and release world-class software with the #1 software development tool used by agile teams.

[Try cloud](#) [Try server](#)

The Cloud option has an eye-catching button, while Server is a simple text link. Cloud is also the option on the left, where we usually find buttons like Save and Publish, as opposed to the right where we often see Cancel.

Another common practice is to make an item seem more appealing in comparison with the other choices on the list. This strategy is commonly seen on eCommerce sites. [Personaliti™](#) offers this example from [BestBuy.com](#). Compared to the \$1,000 camera, the \$45 memory card seems cheap. However, if the user were to search the site for memory cards, they'd find equivalent options at a much lower price.

### Complete Your Purchase

	Item you are currently viewing <b>\$1049.99</b>	+		<input checked="" type="radio"/> Sony - 55-210mm f/4.5-6.3 Telephoto Lens for Most Sony Alpha E-Mount Cameras - Black <a href="#">See details</a> <del>\$349.99</del> <b>\$149.99</b> With bundled savings
				<input type="radio"/> SanDisk - Extreme PLUS 64GB SDXC UHS-I Memory Card - Black/Gold <a href="#">See details</a> <b>\$44.99</b>
				<input type="radio"/> Adobe Creative Cloud Photography Plan (1-User) (1-Year Subscription) - Mac Windows iOS <a href="#">See details</a> <b>\$119.99</b>

It should be noted that, as with the examples of organ donation and employee retirement contributions, these kinds of strategies can be used for the user's benefit, or – as with the examples from Best Buy and Atlassian – to push a product. The important thing is to be aware that when you create choice questions, you can (and frequently are) building in a bias. Therefore, you need to structure and word your choice questions carefully.

## Conditional Logic

Along with being fast and easy for the user, and producing nice clean data for you, choice questions allow you to use conditional logic. Conditional logic allows you to dynamically show or hide form questions based on a user's response to a previous question. This allows you to:

- Reduce cognitive overload for the user – the form looks less cluttered, less overwhelming.
- Shorten the user's path to completion – the user only sees the parts of the form that are relevant to them. There's no confusion about which parts they are supposed to fill out.
- Use one form for multiple, similar use cases.
- Accommodate edge cases.



## Let's look at an example

An HR department is using Jira Service Desk to manage Personnel Action Notifications.

### When the form first loads it looks like this:

Person making request  
Name  
Select...

Employee whose information is to be changed  
Name

Job title

Employee information change  
Item(s) to be changed:  
 Name  
 Address  
 Contact information  
 Exemption status  
 Status  
 Pay rate  
 Expense code  
 Job title

Other  
Comments

### Other questions appear depending on what items need to be changed:

Person making request  
Name  
Demo Project Admin

Employee whose information is to be changed  
Name  
Jane Doe

Job title  
CRM Specialist

Employee information change  
Item(s) to be changed:  
 Name  
 Address  
 Contact information  
 Exemption status  
 Status  
 Pay rate  
 Expense code  
 Job title

Pay status change  
Old status  
 Full-time  
 Part-time  
 Seasonal  
 Temporary  
 On call

New status  
 Full-time  
 Part-time  
 Seasonal  
 Temporary  
 On call

Effective date  
e.g. 01/01/2019

Expense code (acct #) change  
Old expense code

New expense code

Effective date  
e.g. 01/01/2019

Job title  
Old job title

New job title

Effective date  
e.g. 01/01/2019

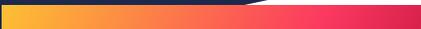
Other  
Comments

Conditional logic is one of the biggest advantages electronic forms have over paper forms. It's also one of the best things you can do to make your forms more user-friendly – especially customer-facing forms on the JSD portal, where the ability to create forms in Jira, without dozens of custom fields or complex configurations, makes Jira and JSD a viable options for non-tech teams. In our next chapter, we'll look at things you should consider when helping those teams (or any team) bring their existing forms into Jira.



# Conversion

## Things to Think About When Converting Forms to Jira



*By Jennifer Choban*

Use these best practices when converting paper and electronic forms to Jira forms and screens.



The first time I converted a paper form to an online form, I made my online form a perfect replica of its predecessor. I figured the team I was building the form for would appreciate my being faithful to their original and I wanted them to feel reassured that transition to digital did not mean they had to abandon their processes.

Later I learned that there are several reasons you shouldn't do that:



- The form may not have been well designed in the first place.
- Processes may have changed since the form was last updated.
- Online forms allow us to use functionality that isn't available for paper or PDF forms. We can make forms that are better for both the team that owns the form, and their customers/users.

# Where to Start When Converting to Jira Forms/Screens

Instead of copying, gather the stakeholders and start from a blank slate. There are several things you should consider before creating your form:

- **Define the user**

Who is the customer/user who will be filling out the form? What are they trying to achieve? Is there information that you'll be asking for that the customer may not know how to find? Or information that they may not want to provide? Are there parts of the current form that customers frequently get wrong? What do you know about your users language level (Are you creating the form in their native language? Are you using technical terms or jargon they may not be familiar with?)

- **Decide what information you need to collect**

List the data points that need to be collected. Be willing to challenge the status quo. Since one of the principles of well-designed forms is minimalism, put every element (section, question, instruction) on trial for its life. If no one can justify why the data is needed, then ditch the question.

- **Note which data points need to be collected from which users**

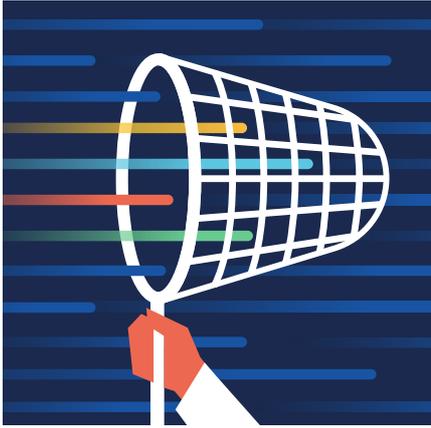
There's a good chance that you don't need to ask every user every question. Note which questions are associated with certain scenarios. You'll use this information later to make some questions conditional, thereby making the form shorter and easier for users.

- **Decide when and where to place instructions**

Large blocks of text are annoying and often ignored by users. Move as much instruction as possible to the field level.

- **Identify validation and formatting needs**

Get clarity on what's required and what's optional. Find out if there are any business rules (choice options, spending limits, etc.) that should be built into the form. Also, find out if there are preferred formats for text fields. Consider whether any of the fields should have a default value.



### **Isolate the data points that will be queried/reported on**

While you're still with the form's stakeholders, find out which of the information points identified in step 2 will be used in queries and reports. Many data points are collected in order to provide a service, but are never queried. Sorting your data points into to camps will help you select the right field types and keep from creating unnecessary Jira custom fields.



### **Locate the form in the business process**

Where does the form fit in the workflow. Are there circumstance where the form should be automatically added to the issue? What should happen once the form is submitted? Is the form part of a compliance strategy?

# Create Your Form/Screen in Jira

Now you're ready to create your screen/form. Remember to:

- Follow the rules of layout and flow to create a user-friendly form.
- **Avoid creating new Jira custom fields unless absolutely necessary.**  
Be judicious about creating new Jira custom fields. An [over-abundance of custom fields](#) makes your Jira configuration complex and will eventually degrade Jira performance. A new custom field should not be created unless it:
  - Can be used by multiple projects
  - Will be queried and reported on
  - Does not duplicate Jira functionality or an existing custom field

Try to use [other options](#) such as a ProForma field, a field configuration, or a [field context](#).

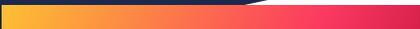
## **Make the most of the functionality offered by online forms.**

- Use [conditional logic](#) to ensure that users only see relevant questions
- Build in [validation](#) and [formatting](#) to ensure clean, complete data
- Use automation (either [JSD's](#) or [ProForma's](#)) to integrate the form into the larger workflow

By engaging in a careful discovery process and employing the rules of good form design, you can help teams maximize usability and efficiency as they bring their processes into Jira.

# Efficient Screens

## and Jira Service Desk Request Forms



*By Rachel Wright*

Screens, screen schemes, request forms...learn the best practices for collecting information in Jira.



When I became a Jira administrator, the most confusing part of project administration was how screens, screen schemes, and issue type screen schemes worked together. Huh? All I wanted to do was to change a few fields around and instead, I found myself lost in a confusing combination of settings that didn't make any sense to me. Shouldn't it be easier? Once I understood the relationship however, I saw how powerful these settings are when they work together. Let's start out with some simple definitions.

## Screens

Screens define which fields are present and their display order. Jira Server and Jira Cloud "Classic" projects have four types of screens. They are:

1. **Create:** A screen for creating a new issue
  - This screen collects the initial information from the Reporter. It often contains just a few of the most important and required fields.
2. **Edit:** A screen for editing an existing issue
  - This screen contains all the fields a user is able to complete or update.
3. **View:** A screen for viewing an issue's details
  - This screen contains all the fields a user is able to view.
  - **Note:** Jira Server and Jira Cloud "Classic" projects only display fields that have data. For example, if the "Due Date" field is empty, you won't see it on an issue's view screen.
4. **Transition:** A screen that is displayed during a workflow transition
  - This screen is often used to collect or update data at different points in an issue's lifecycle. For example, the "Resolution" field value is collected before an issue reaches its final workflow status.
  - **Tip:** Distinguish your transition screens from other screens by naming them with a "(T)". Example screen name: Assignment (T). See screenshot.

Name	Screen schemes	Workflows	Actions
Asset Screen	<ul style="list-style-type: none"><li>• Asset Screen Scheme</li></ul>		<a href="#">Configure</a> <a href="#">Edit</a> <a href="#">Copy</a>
<b>Assignment (T)</b> <small>Contains only Assignee field</small>		<ul style="list-style-type: none"><li>• Approval Workflow (Request Approval)</li><li>• Approval Workflow (Info Needed)</li></ul>	<a href="#">Configure</a> <a href="#">Edit</a> <a href="#">Copy</a>
Default Screen <small>Allows to update all system fields.</small>	<ul style="list-style-type: none"><li>• Default Screen Scheme</li></ul>		<a href="#">Configure</a> <a href="#">Edit</a> <a href="#">Copy</a>

*Image: A transition screen's name is signified with "(T)"*

You can have one screen, or one set of screens, for all issues in your project. Or you can have different screens for each issue type. We'll talk more about that in the "Issue Type Screen Scheme" section below.

Jira Cloud "Next-gen" projects work differently however. There's just one screen per project or per issue type and no distinction between the create, edit, and view operations. "Next-gen" projects treat empty fields differently as well. An empty field displays with the word "None" below it, as pictured.

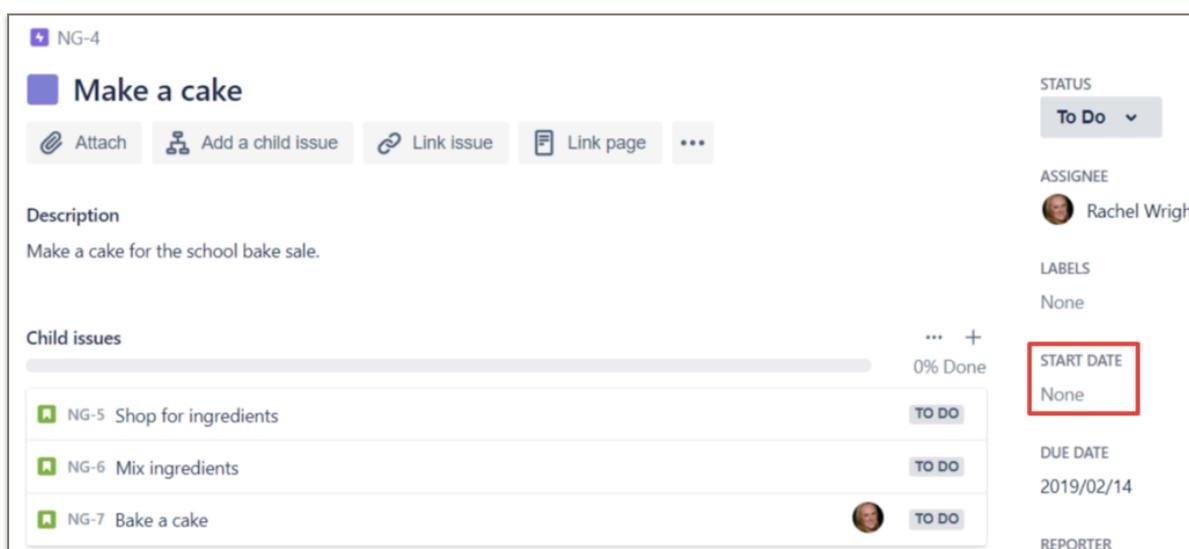


Image: The "Start Date" field is empty, but displayed in a Jira Cloud "Next-gen" project

## Fields and Ordering

In all versions of Jira, screens display both standard and custom fields. Some fields can be ordered as desired by rearranging them on the admin view of the screen. Other fields are automatically placed and grouped together. For example, all user-picker fields ("Assignee", "Reporter", etc) appear together on the right side of an issue's screen. All date fields ("Due Date", "Created Date", "Updated Date") also appear together on the right.

# Screen Schemes

Jira Server and Jira Cloud “Classic” projects have Screen Schemes.

Remember the “create”, “edit”, and “view” operations above? This scheme associates one or more screens with an operation.

In this simple example, there’s one screen for each operation.

## ▼ Epic Screen Scheme

This issue type...	...uses this screen scheme								
 Epic	<table border="1"><thead><tr><th>Operation</th><th>Screen</th></tr></thead><tbody><tr><td>Create Issue</td><td> Epic: Create, Edit and View</td></tr><tr><td>Edit Issue</td><td> Epic: Create, Edit and View</td></tr><tr><td>View Issue</td><td> Epic: Create, Edit and View</td></tr></tbody></table>	Operation	Screen	Create Issue	 Epic: Create, Edit and View	Edit Issue	 Epic: Create, Edit and View	View Issue	 Epic: Create, Edit and View
Operation	Screen								
Create Issue	 Epic: Create, Edit and View								
Edit Issue	 Epic: Create, Edit and View								
View Issue	 Epic: Create, Edit and View								

*Image: The “Epic Screen Scheme” uses the screen called “Epic: Create, Edit and View” for all operations*

In this more complex example, there is one screen for the “create” operation and another screen for the “edit” and “view” operations.

## ▼ Bug Screen Scheme

This issue type...	...uses this screen scheme								
 Bug	<table border="1"><thead><tr><th>Operation</th><th>Screen</th></tr></thead><tbody><tr><td>Create Issue</td><td> Bug: Create</td></tr><tr><td>Edit Issue</td><td> Bug: Edit and View</td></tr><tr><td>View Issue</td><td> Bug: Edit and View</td></tr></tbody></table>	Operation	Screen	Create Issue	 Bug: Create	Edit Issue	 Bug: Edit and View	View Issue	 Bug: Edit and View
Operation	Screen								
Create Issue	 Bug: Create								
Edit Issue	 Bug: Edit and View								
View Issue	 Bug: Edit and View								

*Image: The “Bug Screen Scheme” used the “Bug: Create” screen for the create operation and the “Bug: Edit and View” screen for the other operations.*

A Screen Scheme can have as little as one screen shared by all operations or as many as three screens, with one screen for each operation.

## Why Multiple Screens?

I recommend starting with one screen shared by the “create”, “edit”, and “view” operations in your project. If that screen becomes cluttered with too many fields, or if information needs to be collected during different stages of the workflow, then consider using multiple screens.

## Issue Type Screen Schemes

Jira Server and Jira Cloud “classic” projects also have one final setting called an Issue Type Screen Scheme. This scheme associates screens with different issue types. Just like you can have different screens for different operations, you can have one set of screens for your Bugs, one set for your Stories, and another set for your Tasks.

This Issue Type Screen Scheme has two Screen Schemes. The Bug issue type uses the “Bug Screen Scheme” which has two screens. The Epic issue type uses the “Epic Screen Scheme” which has one screen.

Bug Screen Scheme									
This issue type...	...uses this screen scheme								
Bug	<table border="1"><thead><tr><th>Operation</th><th>Screen</th></tr></thead><tbody><tr><td>Create Issue</td><td>Bug: Create</td></tr><tr><td>Edit Issue</td><td>Bug: Edit and View</td></tr><tr><td>View Issue</td><td>Bug: Edit and View</td></tr></tbody></table>	Operation	Screen	Create Issue	Bug: Create	Edit Issue	Bug: Edit and View	View Issue	Bug: Edit and View
Operation	Screen								
Create Issue	Bug: Create								
Edit Issue	Bug: Edit and View								
View Issue	Bug: Edit and View								

Epic Screen Scheme									
This issue type...	...uses this screen scheme								
Epic	<table border="1"><thead><tr><th>Operation</th><th>Screen</th></tr></thead><tbody><tr><td>Create Issue</td><td>Epic: Create, Edit and View</td></tr><tr><td>Edit Issue</td><td>Epic: Create, Edit and View</td></tr><tr><td>View Issue</td><td>Epic: Create, Edit and View</td></tr></tbody></table>	Operation	Screen	Create Issue	Epic: Create, Edit and View	Edit Issue	Epic: Create, Edit and View	View Issue	Epic: Create, Edit and View
Operation	Screen								
Create Issue	Epic: Create, Edit and View								
Edit Issue	Epic: Create, Edit and View								
View Issue	Epic: Create, Edit and View								

*Image: The Bug issue type has three bug-specific screens. The Epic issue type has only one epic-specific screen.*

## Tying it Together

Screens, Screen Schemes, and Issue Type Screen Schemes work together to power your project. Atlassian explains this relationship in [this diagram](#).

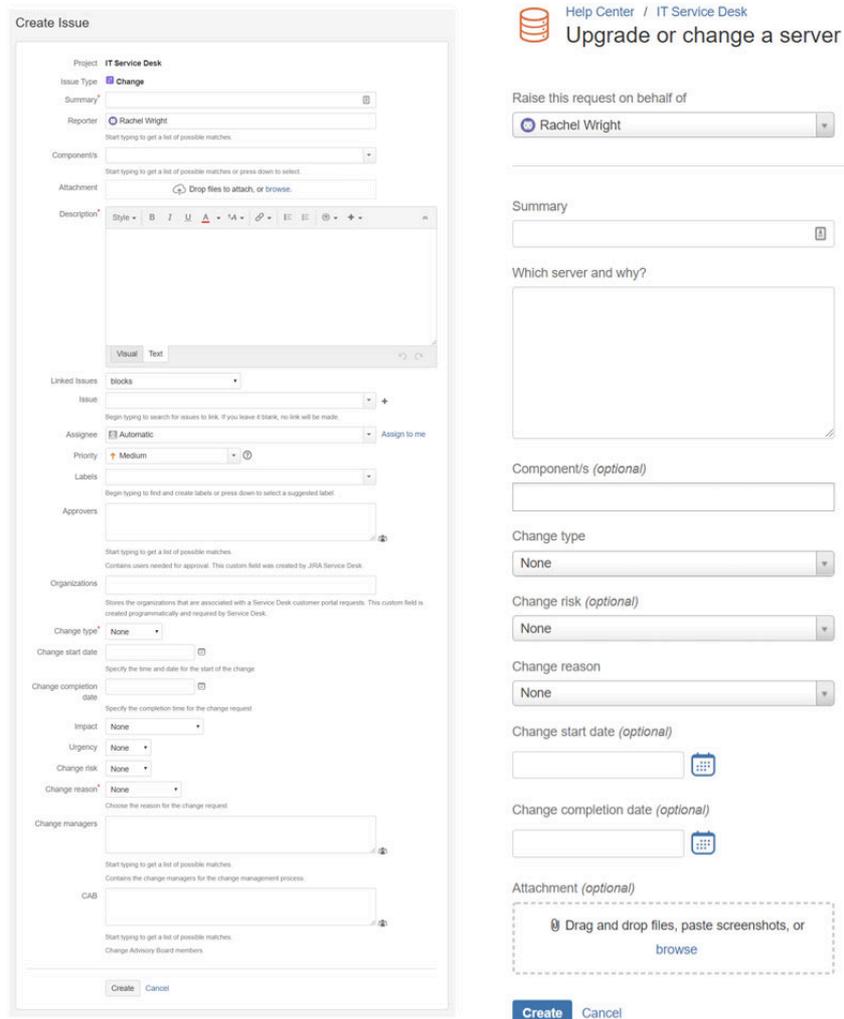
It took me a long time to understand these concepts. I recommend you re-read this article and experiment in your own Jira test environment, until the relationship between these settings is clear.

# Jira Service Desk Request Forms

If you have Jira Service Desk, there's another type of "screen" to be aware of. When Service Desk *Agents* login to Jira, they see the typical Jira screens described above. When Service Desk *Customers* login to the Customer Portal however, they see request forms.

Request forms provide a simpler and streamlined issue view, which is great for less technical audiences. Customers need no Jira knowledge to use the portal to submit their request.

In the example below, the left image shows a default Jira create screen, which contains 21 fields. The right image shows a default Jira Service Desk change request form, which contains only 10 fields. Which one looks easier to complete?



*Image: A Jira change request create screen (left) and a Jira Service Desk change request form (right)*

# Best Practices

Make your screens and schemes as easy, efficient, and reusable as possible. Here are some recommendations:

## As With all Forms

- Don't collect data you won't query on or actually use
- List fields in the order a user would likely supply the information
- Order fields consistently between issues types in a project and between projects. Users expect and appreciate a standard.
  - Example: The "Summary" field is always first, the "Description" field is always second, etc.

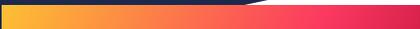
## For Jira Server and Jira Cloud "Classic" Projects

- Use a single screen for all operations ("create", "edit", "view") until there's a real need for additional screens.
  - Consider additional screens when there are too many fields or if information needs to be collected during different stages of the workflow.
- On the "create" screen:
  - Only include the most important and required fields. Too many fields overwhelm users. Too many fields also impacts loading and performance.
  - Only include fields relevant to the Reporter. For example, if a business team member is reporting a Bug, they can't provide an effort estimate and won't know which code version is impacted. Don't show the "Story Points", "Original Estimate" or "Affects Version" fields. Instead, add these fields to your "edit" and "view" screens. You can also prompt a development team member for that information, later in the workflow, using a "transition" screen.
- Create a single screen and a single screen scheme, for all issue types, until more are needed.
  - Example: You want the custom fields "Steps to Reproduce" and "Expected Result" on a Bug's "create" screen, but not on a Story's "create" screen.
  - Example: Create one standard for all development projects and another standard for support projects, not one custom configuration per Jira project.
- Create generic screens and schemes so they can be shared between projects.



# Tips for Creating Good Jira Forms and Screens

Jira Governance for Teams



*By Rachel Wright*

Use these tips to incorporate good form design into Jira screens and Jira Service Desk request forms.



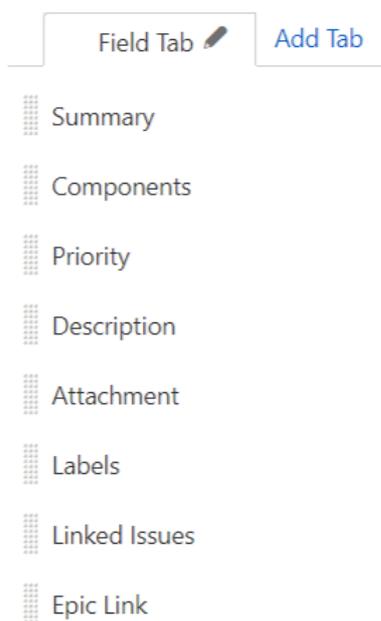
Now that you know why good form design is important and how to ask good questions, here are some quick ways to improve Jira screens and Jira Service Desk request forms.

## Jira

### 1. Limit fields on the Create screen

When you create a project, Jira automatically creates screens and schemes for it. A "Kanban Default Issue Screen" includes 14 fields! By the time you've added additional custom fields, screens are often long and cumbersome. Just because info is needed, doesn't mean it's needed at the same time the issue is created. Group your fields into the following categories:

- information needed immediately (Ex: Description and Requested date),
- information needed later in the workflow (Ex: Estimate and Due date),
- and information needed before an issue is completed (Ex: Time tracking and Root Cause).



*Fields for a Simple Create Screen*

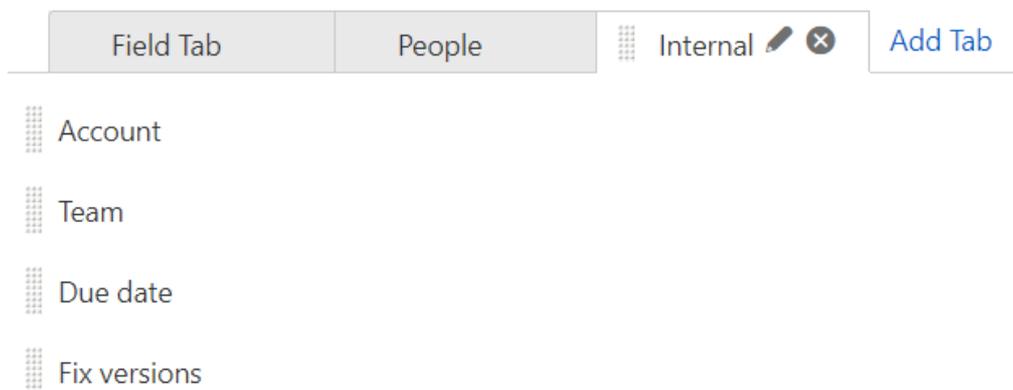
Only show fields in the first category on the "Create" screen. Fewer fields make issues easier to create, especially for non-technical users.

Also only ask for information the creator can immediately provide. For example, if the creator isn't the person who calculates the estimate or determines the release date, omit those fields. You can collect that information, during a scheduling process, later in the workflow.

If you have "Edit" and "View" screens, include all the relevant fields, so info is easy to update at any time. Usually these actions can share the same screen but sometimes they are different. Example: A field has a value but editing it is not desired. In this case, the "View" screen shows the field but the "Edit" screen does not. As a reminder, for Jira Cloud Next-gen projects, there's just one screen per project or per issue type and no distinction between the create, edit, and view operations.

## 2. Use tabs to group similar fields

If there are many fields, use the "tabs" feature to group them. In the screenshot, all user picker fields are together in the "People" tab and all date and version fields are in the "Internal" tab.



*Two Custom Tabs on a Screen*

## 3. Collect additional information during the workflow

Determine when in the workflow other fields should be completed. For example, fields like "Assignee", "Due date", and "Original Estimate" should be filled before an issue reaches the "In Progress" status. Use a workflow transition screen, and validators, to require entry. If you're using [ProForma](#), you can create separate forms to collect information at different times in the workflow.

## 4. Order fields strategically

List fields in the order the user is likely to supply the information. Place more important fields at the top.

Always place the "Priority" field before a "Requested" date field. It may help set realistic expectations to ask for the importance *before* the date.

## 5. Order fields consistently

Use a consistent field order for all issue types and projects. Users expect and appreciate a standard.

## 6. Only create fields that are reported on

Don't show unnecessary fields, collect information you won't use, or create custom fields that aren't queried. Instead, use the standard "Description" and "Comment" fields and train users what information to provide.

## 7. Utilize best practices and standard web form conventions

When creating screens, be aware of the web and application standard conventions that users expect. Here are some tips for effective and useful web forms.

- **Don't ask too many questions**

Only ask for information you'll use. For example, if you plan to respond to issues via email, only ask for an email address (not an email address, a phone number, and a mailing address.) If you already have the reporter's email address on file, don't ask them to type it. Short web forms are more likely to be completed. Users dislike providing many ways for you to contact (aka spam, annoy) them.

- **Ask specific questions**

Use field descriptions to ask the user for specific information or to provide formatting instructions. Asking a specific question gives you better information than a blank or "Enter your message here" description. Examples: "What software do you need installed?" or "What is the expected result of the defect?"

- **If a field has validation requirements, tell the user exactly what to enter**

Give clear and easy to understand directions. Don't wait for a user to enter data incorrectly before providing them with formatting instructions. For example, tell the user to enter their phone number in the format: ###-###-#### rather than provide the vague error "*Please enter a valid phone number.*"

- **Confirm successful submissions**

After a user clicks the submit button, there should be a confirmation that the message was received or an error message if there were any problems. Jira handles this functionality by default.

- **Post and adhere to your privacy policy**

Any time you collect user information, you should have an easily accessible privacy statement that addresses what you collect, how you use it, and under what circumstances, if any, you disclose it. If completing a form means you'll add their email address to your newsletter system, for example, that needs to be clear. This is important for public instances and when you use Jira for customer support.

- **Consider your audience**

As with everything web related, create forms with the end user and their specific goals in mind. You may need separate forms for existing customers, new prospects, or different situations. Don't try to serve all users and all conditions with the same form.



# Jira Service Desk

With Jira Service Desk, you have a different audience to consider. In Jira, the create form should be as short as possible. But in Jira Service Desk, it's important to collect all the important details up front, to avoid multiple rounds of follow-up questions. This is especially important when working with external customers in different time zones.

Use the Jira tips above and these additional tips for JSD.

## 1. Use "Introduction text" to provide portal instructions

Enter a custom message to help users understand support options and share additional help resources. The intro message is especially important when there are multiple Service Desk portals. Intro message space is available in addition to the temporary announcement banner. (Both are pictured below.) Visit *Project Settings > Portal settings* to enter introduction text.

The screenshot displays a Jira Service Desk portal interface. At the top, there is a light blue banner for "Scheduled Weekend Maintenance" with details about system downtime. Below this is the "Help Center" section for "IT Service Desk". A red rectangular box highlights the "Introduction text" area, which contains instructions on how to search for help, request assistance, or provide feedback. Below the introduction is a search bar labeled "What do you need help with?". A sidebar on the left lists "Common Requests" such as "Logins and Accounts", "Computers", "Applications", and "Servers and Infrastructure". The main content area lists several help options with icons: "Get IT help", "Set up VPN to the office", "Request a new account", "Desktop/Laptop support", "Request a desk phone", and "Report a system problem". At the bottom, it says "Powered by Jira Service Desk".

*Sample Portal Introduction Message*

## 2. Use the "Description" field to help users select the correct form

Add a short description for each request form, so users can determine the best selection for their request.

Help Center  
**IT Service Desk**

Need help? Start by searching the [Confluence knowledgebase](#) for answers to common questions. Next, choose an option below to request help, report a problem, or provide feedback. Alternatively, email the help desk at [help@company.com](mailto:help@company.com) or call 800-555-1234 for assistance.

What do you need help with?

Common Requests

Logins and Accounts

**Computers**

- Desktop/Laptop support**  
Report your computer problems
- Get IT help**  
Request other assistance or ask questions

*Sample Form Description*

Always provide a selection for "all other requests". In the screenshot above, there's a generic form titled "Get IT help."

## 3. Use the "Help and instructions" field to set request expectations

Enter custom instructions for each request form so users know what information is needed and how long it usually takes to receive a response. In the screenshot below, the user can expect help within 2 hours for this type of support request.

Help Center / IT Service Desk  
**Desktop/Laptop support**

Please provide us with as many specific details as possible so we can effectively troubleshoot the problem. Please include any error messages or screenshots to help us understand what you are experiencing.

Average response time: 2 hours

Raise this request on behalf of  
Rachel Wright

Summarize the problem  
e.g. My laptop shuts down unexpectedly

*Sample Request Message*

## 4. Customize field labels and add field descriptions

In JSD you can customize a Jira field's label. For example, I often change the default "Summary" label to the more descriptive "Summarize the problem."

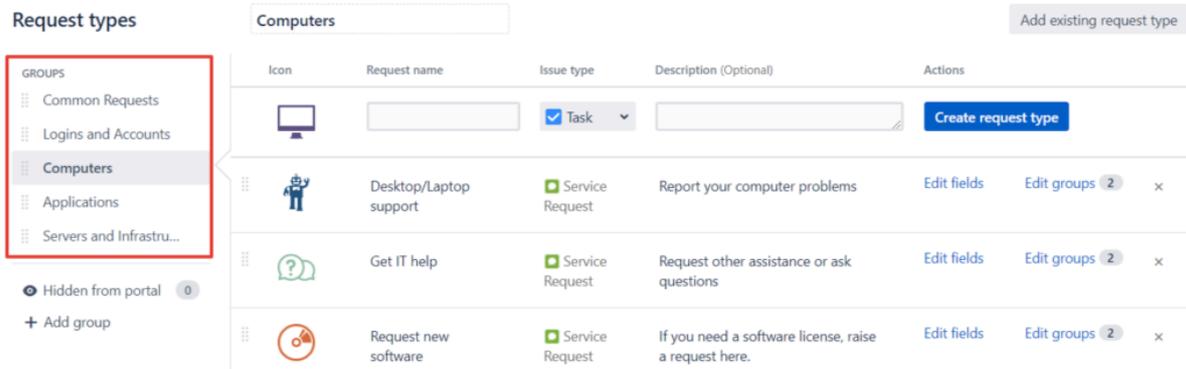
Similarly, you can also customize field descriptions. Use the Jira field description for Jira users and tailor language in the Portal to that audience.

The screenshot shows a Jira Service Desk form for 'Desktop/Laptop support'. At the top, there is a robot icon and the text 'Help Center / IT Service Desk'. Below this is the title 'Desktop/Laptop support'. A paragraph of text asks the user to provide specific details for troubleshooting. Below that, it states 'Average response time: 2 hours' and 'Raise this request on behalf of' with a dropdown menu showing 'Rachel Wright'. A red box highlights the 'Summarize the problem' label for the summary field. Below the summary field is an example: 'e.g. My laptop shuts down unexpectedly'. A 'Details' section follows with a large text area. A red box highlights the custom description for the details field: 'Please provide details or information about your computer. What operating system do you have? Are there any error messages?'.

*Custom Field Labels and Descriptions*

## 5. Group forms by request type

In my former role as a web developer, I always considered a user's [capacity for processing information](#). Too many form choices can overwhelm a user. If you have more than 5 request forms, use the JSD "groups" feature to categorize the list.



*Five Sample Form Categories*

## 6. Use unique form icons

Each request form has an icon. Make each unique and choose icons that visually communicate what each request form is for. If you can't find the right icon, you can make your own. Atlassian recommends a 20px grid with 24px padding.

Finally, and most importantly, make it easy, intuitive, and painless to complete Jira screens and Jira Service Desk request forms. The process should be simple for all users.



### **Simplify Jira with ProForma's custom forms and fields**

Extend the power of your Jira Instance with ProForma.

ThinkTilt's premiere product allows your teams to take control of your processes, without endless customizations, screen schemes and complex configurations. Add the fields you need to a form embedded in a Jira issue.

ProForma, a powerful form builder for Jira, allows you to easily create fields that capture the information you need.

For more information, visit

[thinktilt.com/proforma](https://thinktilt.com/proforma)



The Jira Strategy Admin Workbook will save you loads of time and frustration. Atlassian Certified Jira Administrator, Rachel Wright, shares the lessons she's learned from years of cleaning up messy Jira configurations.

Showing you how to set-up a well-planned implementation that will simplify Jira administration, the workbook contains:

- **152 recommendations for setting up, cleaning up and maintaining Jira**
- **50 companion worksheets**
- **33 real examples of problems to avoid**
- **Templates, code snippets, wording samples and content not available anywhere else.**

Get more information at

[jirastrategy.com/store](https://jirastrategy.com/store)

## Other resources

- Effective Jira Administration | [Click Here](#)
- Battling Jira Custom Field Bloat | [Click Here](#)
- From ITSM to ESM: Enterprise Service Management with Jira Service Desk | [Click Here](#)

# Try the Form Builder

<https://proformademo.thinktilt.net>



