Bugzilla Testopia

http://www.mozilla.org/projects/testopia

Table of Contents

Introduction

Welcome to Bugzilla Testopia, the place where all tests are happy. :-) Testopia is an extension to the Bugzilla bug tracking system designed to track testing efforts for products. Though it is designed with software testing in mind, it can be used to track testing on virtually anything in the engineering process.

As an extension to Bugzilla it integrates closely with bug tracking every step of the way. Testopia provides the ability to link your bugs with the tests that verify those bugs are addressed and vice versa.

Testopia takes a standard black box testing approach. The major objects of testopia are divided into three classes.

Test Plans

Test plans are the top level documents that lay out a testing procedure. In Testopia. plans are used to organize test cases and test runs.

Test Cases

Test cases are the meat of any testing process. These detail the steps to verify that the subject passes muster as well as the expected results.

Test Runs

Test runs are the cumulative results of running a course of test cases against a particular version or build of a product.

Each of these objects is set within the framework of Bugzilla (i.e. Products, Components, Milestones etc) and Testopia relies heavily upon them.

There are also a number of other objects in Testopia that support the big three. We will look at each in turn as we progress through the tour.

So, shall we get started?

Installation

Testopia is an extension to Bugzilla. It goes without saying then that you should have Bugzilla up and running first. Instructions for installing Bugzilla can be found at bugzilla.org as well as in the package you downloaded. Basically all you need is to unzip the package and run checksetup.pl and follow any instructions that come up.

You can run Bugzilla on either Linux or Windows operating systems, though Testopia has not been thoroughly tested on Windows yet.

Beyond the required packages for Bugzilla, in order to install Testopia you should first make sure you have the following requirements met.

Requirements

Bugzilla 2.20 (Currently the only version tested with Testopia 1.0, however this does not mean it won't work with other versions)

Text::Diff Latest version should work

As far as databases are concerned, Testopia just adds a number of tables to the existing Bugzilla database. Currently Testopia only supports Mysql as a database. By this we mean it is the only one tested as of version 1.0 although there is no real reason it shouldn't work with PostGres (the other database engine officially supported by Bugzilla). We just don't have any table creation scripts for it yet.

Testopia has been tested on Mysql 4.1 and 5.0. Bugzilla requires 4.1 as of 2.20.

Steps

1. Copy the testopia tar ball into your bugzilla's root directory.
2. Untar

   ```
   tar xzvf testopia-<release>.tar.gz
   ```

3. Run tr_install.pl

   ```
   perl tr_install.pl
   ```

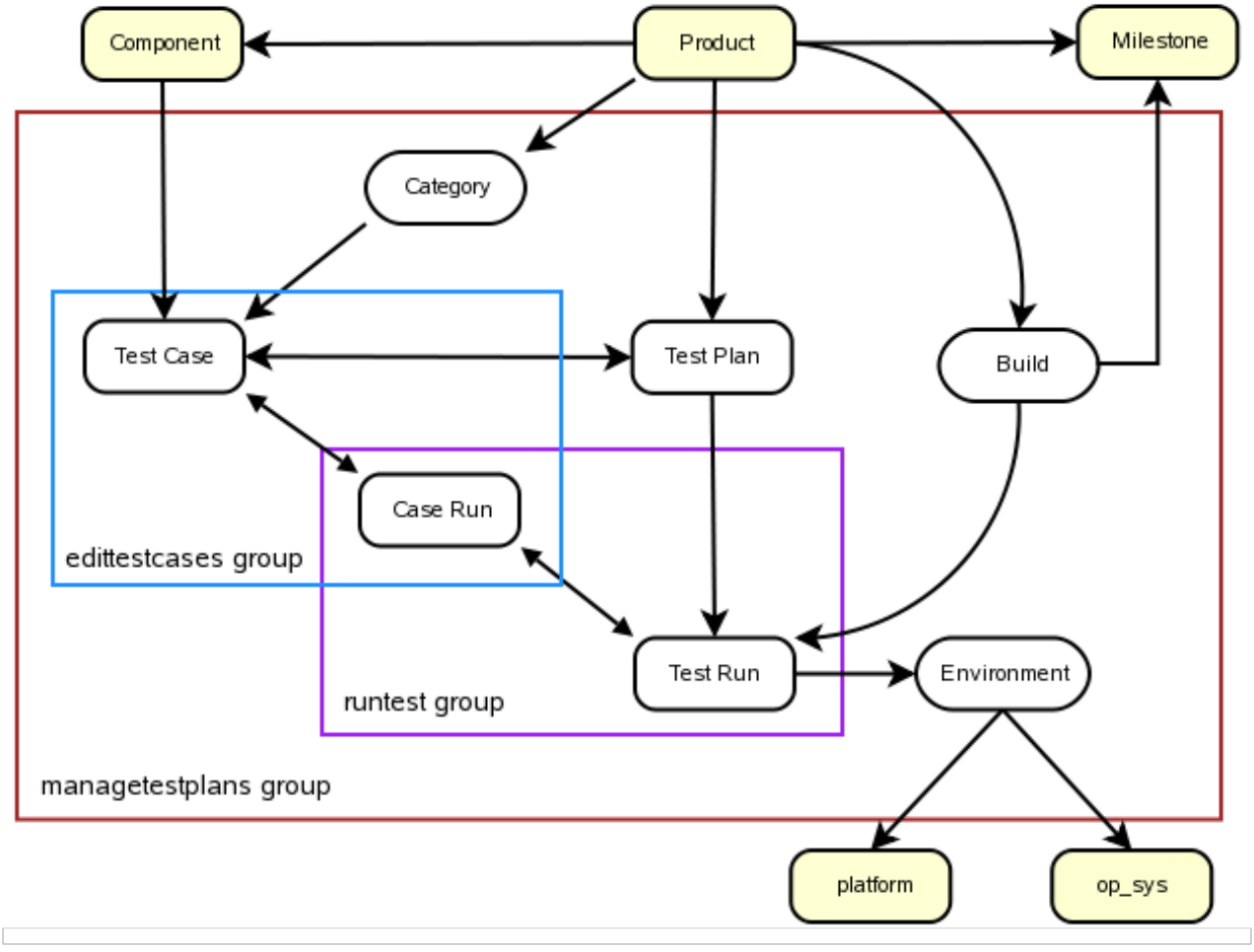
The install script sets up the database tables, patches your code, and sets up three new groups: managetestplans, edittestcases, and runtests.

If the patch fails, you may want to patch the files by hand. There are only 4 small changes to the Bugzilla 2.20 code.

You should now be able to See the testopia links in the Bugzilla footer. Make sure you update the new testopia user groups to include whomever is going to need access to test cases.

The Big Picture

We will start with the 10,000 foot view and zoom in from there. This section is called the big picture since, well, it is a big picture.



As you can see, Testopia objects are arranged in a hierarchy under products. At the top are test plans. From a test plan it is possible to create test cases and test runs. Test cases are broken up by categories which are associated with products. Test cases can also be associated with one or more components. Test runs are classified by builds which are in turn broken into target milestones. Runs also have an associated environment which can include a platform and an operating system. We will take a look at each of these in turn starting from the top.

Test Plans

Let's take it from the top.

As mentioned earlier, test plans are kind of the filing cabinet for everything in Testopia. Test plans are divided by product and act as the dashboard for everything you will do as a tester.

To create a new test plan, click the New Plan link in the Bugzilla footer. You will be taken to a page where you will be prompted to fill in a few fields, Lets take a look at those fields.

Fields

Name

The plan name is a simple text field. Plan names can consist of any combination of letters, numbers and other characters and do not have to be unique. In fact you can have two plans of the exact same name if you like. This may actually be desirable as we will see later.

Product

Each plan must have one and only one product. This comes from the Bugzilla product list. As a tester you must have access to the product (i.e be a member of the product group) to be able to see and create plans for the product.

Product Version

Once you choose a product, the product version list will show you available versions for that product (Javascript must be enabled to have the magic happen). This will be used as the default for setting up runs, which we will see later.

Type

This is where you choose which type of testing this plan should represent. the default list consists of such things as Unit, System, Integration, Acceptance, and Performance, however system administrators can customize this list to meet their needs. Each plan type will represent a different mode of testing.

Plan Document

This is a place to write up the test plan's procedure document. Depending on your organization and requirements, this can be very detailed. or not. This area shows up with a WYSIWYG editor to allow you to include formatting easily. Again, Javascript is required.

Once you hit submit, you will be taken to the plan view for your newly created plan.

At the top of the plan page is a table containing an overview of the fields contained in a plan. You will also notice some buttons.

Commands

Archive

As products move through their lifespan you will often find plans associated with earlier versions to be obsolete. Plans can therefore be archived to hide them from your basic searches.

To archive a plan, click the Archive button at the top of the plan page.

If a plan is already archived the button will Unarchive the plan instead.

Print

If you would like to see a printable version of the plan document, click the Print button.

History

You can view the history of changes to a plan, as well as view unified diff comparisons of versions of a plan's text by clicking the History Button.

Cloning a Plan

Plans can be cloned or copied. This is especially useful as you move from one version of a product to the next. You will likely need to carry forward many of the attributes and test cases that you created for an earlier version. Cloning allows you to do this in a few mouse clicks.

When you click the Clone button, you are presented with some options for cloning your plan.

The first couple are pretty straightforward. Give your copy of a plan a name. There is nothing preventing you from using the exact same name, though it might be harder to distinguish from earlier versions if you don't give it something at least a little unique.

Second, choose a product version for this plan.

Next you have the option of copying the plan document. If you select NO you will have a clean slate for your new plan.

You also have the option of pointing any tags you have set to this plan to the new plan as well. We will cover tags later.

Link vs Copy

Lastly you have the option of moving your test cases forward. You have two options here. If you choose to copy the test cases you will, in essence, be cloning those just like you are the plan. They will be exact replicas except they will not have the same ID.

The benefits of copies are that they are now unique test cases which can be edited at will without the fear of messing up other plans that might have been linked to them. However this means you now have two test cases where one existed before. This will increase the workload if you need to update test cases across versions of a product.

If you instead choose to link your test cases, what you get is the equivalent of a symbolic link in Unix. That one test case will now be pointed at more than one plan. This makes updating your test cases across plans a breeze. However if you need to branch information in a test case for a different plan, you should clone it. You will see that you can clone a case just as easily as a plan from the case page.

If you choose to copy or link the test cases as you clone your plan, you have the option of only taking those cases that match a certain category or categories.

Once you have selected all the options for your copy, click Clone to be taken to your new plan.

Categories

Before setting up a test case you need to have at least one category defined to put it in.

Categories are buckets for test cases. Each test case can belong to only one category at a time. They are here to help you organize your test cases into useful groups.

Categories are actually defined by product, meaning if you define a category in a plan for a given product, it will become available for all plans in that product.

To create a category. Click the add link under the categories list in your test plan. You can also click the Categories header and then click add on the page that comes up.

A category is very simple, It consists of Two fields, only one of which is required.

Name

A category must have a name. It can consist of any combination of letters and numbers and must be unique inside the product you are creating it for.

Description

This optional field is for you to provide a verbose description of your category if you like.

Builds

Builds are similar to categories but they apply to runs.

A build is a particular iteration of a product. In software a build is often something that represents the compiled results of all the source code committed during an interval of development. That interval can be as frequent as every hour or as little as once a week, Regardless of how often a build is done, the main thing we want to track here is which builds get tested.

Like categories, builds are product specific and will show up in any plan for that product. Since builds can happen very frequently, you may wish to automate build creation using whatever build system you have devised.

To create a new build manually click the add link under the build list in your plan, You can also click the build header and click the add new link on the page that appears.

There are three fields for builds

Name

The name can consist of any letters or numbers and special characters. It must be unique within the product.

Milestone

This is the target milestone from the product. Many builds can use the same target milestone.

Description

This area can be used to put notes or a description about the build. It is optional.

Test Cases

Test cases are the heart of Testopia. A test case is a set of steps and expected results that should be applied to a subject in order to find out if a product is sound.

All test cases need to be associated (linked) to at least one test plan. The easiest way to go about creating a test case then, is to click the Create Test Case link on the plan page.

Let's take a look at the fields of a test case.

Fields

Summary

The summary is a short description of the test case. It can contain any combination of characters and does not need to be unique. This will allow you to quickly identify test cases in a list.

Alias

Like a summary, Aliases are used to help you quickly identify a test case. Unlike summaries however, they must be unique in the system, meaning no two test cases, regardless of product, can have the same alias. Aliases can be used in place of ID's in linking up test cases. Their main purpose is as an aid for automations that would like to refer to a test case by name rather than number.

Status

The status field for test cases helps you determine which test cases are actively being used for testing versus those that are only in development or are no longer relevant. Status can be one of three options.

Proposed

Test cases with a status of proposed are currently being defined. They should not yet be used for testing as they are in flux.

Confirmed

Confirmed test cases are those that are actively being used in testing.

Disabled

Disabled test cases are either no longer relevant to a current version of a product or have been taken out of the current testing loop for whatever reason.

Priority

The test case priority is determined by how important a test case is in relation to others in a product. It can also refer to the priority of any bugs that are identified by the test case or for which the test case is designed to check. This list is identical to that of bug priority.

Category

As mentioned above, categories are buckets for subdividing test cases. They work in tandem with product components to allow you to quickly find test cases.

Components

Once you have created your test case, you have the option of associating one or more Bugzilla components with it. The list of components comes from the products that this case is associated with via the plans it is linked to. This is optional as some test cases will be independent of component.

Default Tester

This person will be the default assignee for the created case-runs.

Requirement

This field is optional and is used for tracking a requirement number if one exists for this test case.

Automatic

If this test case is meant to be used with a script or automated system then choose automatic. This then allows you to set the path or URL for the script that is used. The default however is manual.

It is important to note that Testopia does not automate any tests of its own accord. This field is just to help you classify which of your test cases are automated versus which are manual.

Script

This field is only available if Automatic is chosen above. This is a place to store a path or URL to an automated script. Testopia does not run the script for you however, this is just to display where a user would look to find the script.

Arguments

If your automated script takes any arguments you can store which ones apply to this test case in this field. Again, this is only active if Automatic is set to Automatic.

Dependencies

Like bugs, test cases have a concept of dependencies. If a test case depends on another test case it would be prevented from being passed if that prerequisite case fails. Dependencies set up an order of operations for running test cases.

If test case B "depends on" test case A, then whenever test case A fails, test case B will be blocked from passing (it will receive a status of BLOCKED). So, A should run before B, and B should not be run if A fails. At some point this will be automatic, meaning failing a test case that blocks other test cases will automatically apply the BLOCKED status to those dependent test cases.
Testopia will check that there are no loops in your dependencies.

Depends On

Makes another test case a prerequisite for this test case.

Blocks

Makes this test case a prerequisite for another.

Action

The action is the list of steps to confirm that the subject under scrutiny passes inspection. The successful completion of which would earn a PASSED status in a run. This should follow general testing guidelines and be concise, checking the smallest unit possible. An example might be to "click the file menu and choose save".

In the case of performance testing this might detail the list of conditions the test was run under.

Expected Results

The Expected Results is the list of successful outcomes that the action should produce in order for a test case to be considered passed. Meeting these objectives would produce a passed result where failing to meet them would produced a failed result. To extend our example above, an expected result might be "check that the file is saved on the file system".

In performance testing this might contain the list of acceptable parameters.

Commands

History

Just as with test plans, it is possible to view the history of changes to a test case by clicking the History button. You can also take a unified diff of any two versions of the action and Expected Results on the history page.

Clone

You can clone a test case in much the same way you clone a test plan. By clicking the clone button you have several options.

First you must choose whether you are making a copy of a case or just linking an existing case to another plan.

If creating a copy, you have the option of copying just to the existing plans or a list of different plans or both. You can choose to copy the document (action and Expected Results) or leave it blank on your copy. You can copy the tags and components as well.

Overview

Once you have set up a test case you will notice at the top of the screen an overview section that details many of the fields you saw while setting up your test case as well as a few others. This will contain such information as the author of the test case and when it was created as well as links to the test plans it is linked to.

Case Run History

Included in the view of each test case is a table with the history of test runs that it has been run in. You will see the status it received and who it was assigned to as well as who actually ran it if it has been completed. There is also a link to a bug list of any bugs that were identified during the run. You can click on the run number to be taken to the test run page.

Attachments

Both test plans and test cases can support attachments. You will find the attachments table near the bottom of both of these pages. To create an attachment, simply click the browse button and locate the file you wish to attach, type a description in the Description field and click the attach button.

Attachments are treated very much the same as they are in Bugzilla. The size limit will be determined by the max-attachment-size parameter the same as it is for Bugzilla

To edit an attachment click the edit link. This takes you to a page where you can set the description, filename, and mime-type for the attachment as well as view it if it is a viewable type or download it otherwise.

Test Runs

Test runs are the work horse of Testopia. A test run is the application of a list of test cases against the subject to determine if the subject meets the standards laid out by the test case. Each test run is associated with a single plan and a list of test cases.

Creating a Run

To create a test run, click the Create a New Test Run link in the plan you are creating the run for. You are taken to a page with a list of test cases that are associated with this plan. You can select all, part, or none of the test cases to be included in this run. Later you can search for specific test cases to add to the run if you like.

For each test case associated with a run a record is created for the history in the test case for the result of this run against that case.

Fields

CC List

Once you complete the creation process you will have the option of adding and removing people from the CC list for the run. These people will eventually receive notification of changes to a test run in addition to the manager and testers. Notification is currently not working as of 1.0

Product Version

Though a plan has this information, you may wish to mix test runs of other product versions for comparison. This allows you to set the actual product version used for testing in this run.

Plan Version

It is possible to set which version of the plan document you mean to run this test against. The default is the latest version

Manager

The run manager will usually be the person responsible for running the test cases.

Status

There are two options here: Running and Stopped. If a test run is stopped, none of the test cases in the run can be passed or failed.

The default is running.

Build

This allows you to set the default build for the case-runs. This list comes from the builds defined for the product on the test plan pages.

Environment

Choose from the list of defined environments the one used for this run. Environments will be explained in detail later.

Summary

This is a short description of the run. Much the same as that of a case summary.

Notes

This is an area for you to make any notes about this run. This field is fully editable meaning that if you remove text it will be deleted.

Overview

Once you have created your run, you will see an overview section detailing such things as, which plan this run belongs to, when it was started, which product and product version as well as who the manager and default tester is.

Below the overview section you will notice a table of numbers detailing how many test cases have been completed and what their total outcomes have been. There is also a percentage bar showing the completion rate.

Commands

History

Just as with plans and cases, you can view a history of changes to the run. This does not include changes to the notes however.

Clone

You can clone a run the same as with other objects. When you clone a run you can choose to copy over the test cases from the run that meet the criteria you specify.

Running a Run

A test run is the area that most testers will spend most of their time in. This is where you will pass and fail test cases and log bugs against them.

When you set up the run, you selected which test cases will be included. You will see one entry per test case in the Test Case Run Table in your run. By clicking on the black triangle next to the case ID you can expand the case run form.

The first step in running a test case is to set the build if it is different from the default build. Testopia is designed to be flexible in that it allows you to define your process. If you have many builds and they come very frequently, you may not wish to create a single run per build but would rather have a parent run and set the build on each test case that you tested against. Once the build is set you can begin your testing for that test case. Upon completion you click the appropriate button to either pass or fail the test case. In the case of performance tests you can choose to set the status to RUNNING while you perform the test and PAUSE it at anytime during the process.

The status of blocked is used when a test case has dependencies and the prerequisite case failed.

You have the option of making specific notes for each test case as you run it in the notes field. You can also log and attach bugs to a test case. Clicking the NEW button will open the bug creation screen in a new window. When you submit the bug, it will automatically be associated with this test case. You can also reassign the test case.

There are a number of filters available to allow you to display only the case-runs you are interested in. You can also sort on virtually any field in the table.

If you wish, you can update multiple test case-runs at one time by clicking the link at the bottom of the table.

This format uses AJAX to make a more dynamic user experience. If you prefer, you can use the Classic interface by clicking the link here. You are presented with the same options but in a more static view.

Tags

Tags are a novel approach to categorizing objects. They have become very popular with numerous web tools and have met with much success.

The idea behind tags is that each user can categorize each item to his or her own liking without destroying other users categorizations. Each of the big three (Cases, Runs, Plans) in Testopia can have tags associated with them.

To tag an object simply enter a string in the tag field and click add. You can search and sort by tags by clicking the Tags header and then clicking the appropriate object of interest in the list.

Environments

Environments are definitions of how a test run was run. In software testing this might include such things as which Operating system was used or what hardware platform. Environments can be as broad or narrow as you define them. The basic environment consists of an OS and platform chosen from Bugzilla's lists of these objects. However it can be much more complicated such as a suite of applications and other products or a browser. It could be a temperature range or a certain list of physical characteristics that your test must be conducted in.

Creating an environment in Testopia requires two steps. The first involves defining a set of variables to be used in your environment. The second is to create the environment from the set of possible elements.

Environment Administration

When you first install Testopia, you must first define the set of environment variables that will be used to construct your environments. To access these, click the *Environment Variables* link in the footer. The environment variables are arranged in a hierarchy of objects that is represented as a tree. There are four major levels: Category, Element, Property, and Property Values.

Categories

Environment Categories are similar to test case categories in that they provide a sorting mechanism for your environment elements. Each category is associated with a single product or in the special bucket labeled *–ALL--* . The –All-- denotes all products, meaning it holds categories of elements that are not specific to any product.

When you first install Testopia you will see that the –ALL-- bucket contains two Categories, OS and Platform. Expanding these you will see that there are elements representing each of the OS and Platform values defined in Bugzilla. This is list is generated at the time you first install Testopia and is maintained separately from the Bugzilla lists thereafter.

To create a category, right click on the product or –ALL-- and choose *Add Category*. You can then click on the newly create category in the tree which will pop up a form that allows you to edit the category name or change the product it is associated with.

Elements

Once you have a category defined for your product or the –ALL-- bucket, you can add elements to that category. Elements are the crux

of what makes up your environment. To create an element, right click on the category you wish to add it to and choose *create element.*

This will create an element labeled “New Element” which you can edit by clicking on it in the tree, or right clicking and choosing *Edit.*

Elements can be nested inside other elements. To create a sub element, right click on the element and choose *Add Element*. You can edit this child element in the same manner as its parent. You can create as many levels of elements as you need to represent the complexity of your environment.

Properties

Properties describe your element. You can add properties to your elements by right clicking the element and choosing *Add Property*. You can add as many properties to your elements as you need. Properties cannot be nested however.

To edit your property, click on it in the tree or right click on it and choose *Edit*.

Property Values

Once you have defined a property for your element, you will need to provide a list of values from which to select for you environment. Right click on your property and choose *Add Value* to create a value for your property. You can edit property values by right clicking and choosing Edit, or by clicking on one of the values under your property.

You can change the name or reorder the list of values from the form provided. You must hit Save Changes in order for you changes to be committed.

Creating Your Environment

Once you have set up the elements that will be used in your environment, you can now create environments with those elements.

Click *New Environment* in the footer.

You are prompted to name your environment and choose a product for it. The product is used only for classification. It does not limit your choices of which elements can be placed in your environment.

Clicking *Create* will take you to the environment editor. Here you will see two trees, one representing your new environment and the other containing the list of variables from which to choose. Your environment will consist of the elements you defined earlier. To add an element, find it in the list and simply drag it onto your environment tree. The order does not matter. You can grab childe elements at any level, but dragging an element with children will bring the children as

well.

Once you have selected the elements for your environment, you can now select which of the property values apply to your environment. Expand the element and property and simply click the value you wish to use. It will have a star placed next to it to show your selection.

To remove an element, right click on it and choose R*emove*.

All changes to your tree are saved immediately.

Searching

Testopia incorporates Bugzilla's search engine capabilities for searching on test cases, runs and plans. Each of these has its own set of search parameters which can be found by clicking the corresponding tab in the search page.

Search results are displayed as a table which is sortable. Clicking on the column you wish to sort by will return the results sorted on that column. Currently only single key ascending sort is supported.

The results table is also paginated. Clicking the navigation links at the bottom will allow you to jump from page to page of the results. Sorting will reset you back to page one however.

The Testing Life Cycle

Testopia can be made to fit a wide variety of product development and testing models. This is just one suggested model. Feel free to to tailor Testopia to fit your needs.

Create a Test Plan

As covered above, everything in testopia revolves around test plans. A typical test plan would be used to encompass a single product and product version. You should also set up categories, builds, and environments to match your testing scenario.

Create or Select Test Cases

When just starting out, you will need to create test cases for each of the areas that are to be tested in your product. The number of test cases can be in the hundreds or hundreds of thousands. Testopia is designed to scale to meet the needs of large and small testing groups alike. If you are creating a plan for a different version of an existing product, you can link the existing test cases by cloning the test plan, adding test cases where appropriate to cover new features or old bugs.

Start a Test Run

Once you have test cases created, you can start running tests. Create a test run and select the test cases you wish to cover. If you have multiple environments to test in, you may wish to create a run for each environment. Each run has an associated build. Depending on how often builds happen, you may want to have separate runs for each build. If builds happen more frequently than test cycles, you can change the build on each individual test case-run as you go.

Test cases have an assigned tester which is designated when the test case is created. As you go through the run however, you may wish to reassign particular test cases to different testers. This can be done individually or to a group of case-runs at once. Though the default tester is responsible for setting a status on a test case-run, anyone with rights to edit a test run can update the case-run. The actual person to do so is captured in the tested by field.

Cleaning Up

Once all the test cases are given a closing status (PASSED, FAILED, or BLOCKED), the test run should be moved from the Running state to the Stopped state.

When all test runs are completed and the product is released and reaches it's sunset, the test plan can be archived. If any test cases are no longer relevant they can be moved to the disabled status.

Glossary

Action	The list of steps that a test case must complete.
Alias	A globally unique string that identifies a test case in conjunction with the test case ID.
Archive	Test plans may be archived and hidden from regular searches.
Arguments	A list of parameters to send to an automatic test script .
Assignee	The person responsible for applying a status to a test case-run
Blocks	A test case that blocks another test case.
BLOCKED	A status of a test case-run indicating the prerequisite test case failed.
Build	In software testing, a string denoting the compiled results of a period of development.
Category	A property of a product that is used to classify test cases.
Clone	An exact replica of data between two objects. In Testopia you can clone plans, runs, and cases.
Component	A Bugzilla component. An attribute of a product.
CONFIRMED	A status of a case. Confirmed test cases have been approved for use in test runs.
Default Tester	The default person responsible for applying a status to the test case-run for a given test case.
Dependency	Test cases can be dependent on other test cases. There are two types of relationships, depends on and blocks. A test case that is blocked by another should not be examined before the prerequisite test case as success is determined in part by the outcome of the predecessor.
Depends on	Sets up a dependency between test cases. Depends on lists the test cases that a particular test case requires to be completed before this case.
DISABLED	A status of a test case denoting it is no longer used for active testing. Similar to archival for a plan.

Expected Results	The expected results upon completing the action of a test case.
Environment	A list of the surrounding conditions that a test run is performed in.
FAILED	A status of a test case-run. Denotes the test case failed in the given run.
IDLE	A status of a test case-run. Denotes the test case has not been examined in the given run.
Manager	The person in charge of a given run.
Milestone	A Bugzilla object. A property of a product that implies when a given bug or feature will be fixed or included. Testopia builds are associated with milestones.
PASSED	A status of a test case-run. Denotes the test case has met the conditions of success detailed in the Expected Results of the test case in the given run.
PAUSED	A status of a test case-run. Denotes the test case has been under examination in the given run and is on hold. Used primarily for performance tests that may span long time periods.
Plan Document	The information of a test plan detailing what the test plan will cover for what by when. Depending on the level of scrutiny required it can be very verbose.
Plan Version	The version of the plan document used for a particular run.
Priority	The Bugzilla priority. Test cases can be assigned a priority similar to bugs.
PROPOSED	A status of a test case that denotes it has not yet been approved for use in test runs.
Requirement	A field of a test case provided to capture information about a requirement. Typically an ID of a requirement in a separate requirement tracking system.
RUNNING	A status of a test case-run. Denotes the test case is in the process of being examined in the given run.
Running	A status of a test run. Running test runs can have case-runs updated and implies that there is further testing to be done in the run.

Script	A path to an external automated test script for a given test case. Testopia does not run this script, the field is only provided as a way of informing the user where to find it.
Stopped	A status of a test run. Stopped test runs can not have case-runs updated. This status denotes the run is complete.
Tag	A user defined string used to classify test plans, cases, and runs.
Test Case	A list of conditions and expected results for success for a particular feature or object under scrutiny. Test cases are associated with one or more test plans and with zero or more test runs.
Test Case-run	The union of a test case and a test run. Each time a test case is included in a new test run, an entry is made for it in the test case-runs table. This captures whether the test case passed or failed in the given run. Each case-run should be associated with only one build for a given status.
Test Plan	The defining object in Testopia. Organizes the other objects in Testopia.
Test Run	The instance of performance in Testopia. Each run is associated with a single plan and environment. It contains a list of test cases to be examined and stores the results in the case-runs table.
Tested By	The person who examined and applied a status to a given case-run
Type	The plan type. Plan types might include System, Unit, Integration etc. Each plan can be of only one type.

Getting Help

There are a number of resources for getting help in Testopia.

Mailing lists

- support-webtools@lists.mozilla.org
- dev-apps-webtools@lists.mozilla.org

IRC

irc://irc.mozilla.org/testopia

Wiki

http://wiki.mozilla.org/Testopia