

The Power of the New Web Browser Item



Michael Lindauer

Software Engineer
Zabbix Certified Expert
Rhein-Main Solutions





Zabbix Browser item - basics

Zabbix Browser item - basics

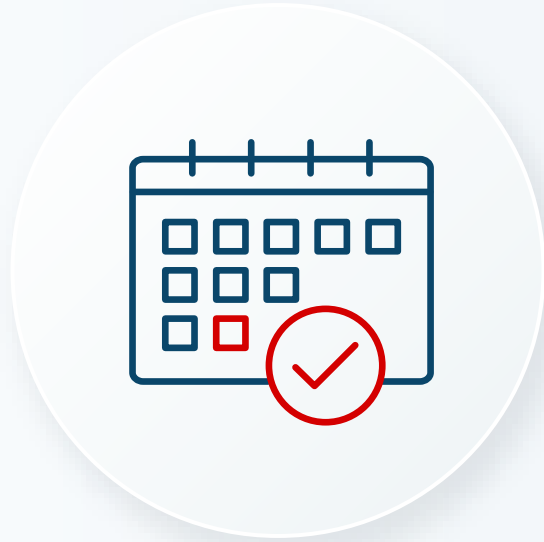
Introduced in Zabbix 7.0

Allows synthetic monitoring of complex websites

Uses an additional Web Driver (like Selenium or Chrome Driver)

Significantly expanded capabilities compared to the web scenarios





Monitoring a booking tool

Monitoring a booking tool

Scenario: Go through all the necessary steps to book a valid (!) appointment!

Extension: Delete/cancel this appointment after logging into the backend!

The example scenario is using Easy!Appointments



The screenshot shows a web browser window with the Rhein-Main Solutions logo and navigation tabs. The main content area is titled 'Service & Provider' and contains two dropdown menus. The first dropdown, labeled 'Service', has 'Consulting 60' selected. The second dropdown, labeled 'Provider', has 'Michael Lindauer' selected. Below these dropdowns, the text 'Duration: 60 Minutes, Price: 29.00 ♥' is displayed. At the bottom right of the form, there is a dark button labeled 'Next >'.

RHEIN-MAIN SOLUTIONS Rhein-Main Solutions
Consulting 60 | Michael Lindauer

1 2 3 4

Service & Provider

Service

Consulting 60 ▼

Provider

Michael Lindauer ▼

Duration: 60 Minutes, Price: 29.00 ♥

Next >

Step 1: Choose service

Step 2: Choose provider

Step 3: Click „Next“

The screenshot shows a web browser window with the Rhein-Main Solutions logo and navigation tabs at the top. The main heading is "Appointment Date & Time". On the left is a calendar for May 2025, with the 19th selected. On the right is a "Timezone" dropdown set to "UTC" and a list of time slots: 14:00, 14:15, 14:30, 14:45, and 15:00 (which is highlighted in blue). At the bottom are "Back" and "Next" buttons.

RHEIN-MAIN SOLUTIONS Rhein-Main Solutions
Consulting 60 | Michael Lindauer

1 2 3 4

Appointment Date & Time

< May 2025 >

Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Timezone

UTC

14:00

14:15

14:30

14:45

15:00

< Back

Next >

Step 4: Choose a date

Step 5: Choose a time

Step 6: Click „Next“

The screenshot shows a web browser window with a dark blue header bar. The header bar contains the logo for 'RHEIN-MAIN SOLUTIONS' and the text 'Rhein-Main Solutions Consulting 60 | Michael Lindauer'. Below the header bar, there are four numbered tabs: 1, 2, 3, and 4. The main content area is titled 'Customer Information' and contains a form with the following fields:

- First Name *: Maria
- Last Name *: Mustermann
- Email *: m.mustermann@rhein-main-solutions.com
- Phone Number *: +49 6022 90175 00
- Address: Wailandtstraße 18
- City: Aschaffenburg
- Zip Code: 63741
- Notes: (empty text area)

At the bottom of the form, there are two buttons: '< Back' and 'Next >'. The 'Next >' button is highlighted in dark blue.

Step 7: Enter data

Step 8: Click „Next“

Step 9: Click „Confirm“

The screenshot shows a web browser window with a dark blue header for 'RHEIN-MAIN SOLUTIONS'. The header includes the company logo, name, and contact information for Michael Lindauer. Below the header, the page title is 'Appointment Confirmation'. The main content is divided into two columns: 'Consulting 60' and 'Contact Info'. The 'Consulting 60' section lists the consultant's name, date and time, duration, time zone, and price. The 'Contact Info' section lists the contact person's name, email, phone number, and address. At the bottom, there are two buttons: 'Back' and 'Confirm'.

RHEIN-MAIN SOLUTIONS Rhein-Main Solutions
Consulting 60 | Michael Lindauer

1 2 3 4

Appointment Confirmation

Consulting 60

Michael Lindauer

19/05/2025 15:00

60 Minutes

UTC

29.00 ♥

Contact Info

Maria Mustermann

m.mustermann@rhein-main-solutions.com

+49 6022 90175 00

Wailandtstraße 18

Aschaffenburg, 63741

[< Back](#) [Confirm](#)



Automation with JavaScript

Automation with JavaScript

We use the Zabbix standard template „Website by Browser“

```
const Website = {
  params: {},

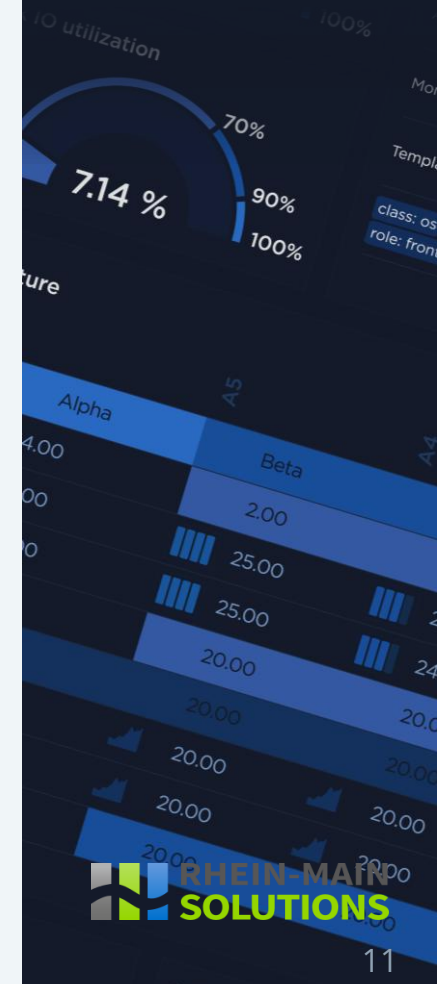
  setParams(params) {
    ['scheme', 'domain', 'width', 'height'].forEach(function (field) {
      if (typeof params !== 'object' || !params[field]) {
        throw new Error('Required param is not set: ' + field + '.');
      }
    });
    this.params = params;
  },

  getOptions(browser) {
    switch ((browser || '').trim().toLowerCase()) {
      case 'firefox':
        return Browser.firefoxOptions();
      case 'safari':
        return Browser.safariOptions();
      case 'edge':
        return Browser.edgeOptions();
      default:
        return Browser.chromeOptions();
    }
  },

  getPerformance() {
    const browser = new Browser(Website.getOptions(Website.params.browser));
    const url = Website.params.scheme + '://' + Website.params.domain + '/' + Website.params.path;
    const screenshot = "";
    browser.setScreenSize(Number(Website.params.width), Number(Website.params.height));
    browser.navigate(url);
    browser.collectPerfEntries();
    screenshot = browser.getScreenshot();
    const result = browser.getResult();
    result.screenshot = screenshot;

    return JSON.stringify(result);
  }
};

try {
  Website.setParams(JSON.parse(value));
  return Website.getPerformance();
} catch (error) {
  error += (String(error).endsWith('.')) ? '' : '.';
  Zabbix.log(3, ['Website get metrics'] ERROR: ' + error);
  return JSON.stringify({ 'error': error });
}
```



Automation with JavaScript

Let's concentrate on the fun part

```
const Website = {
  params: {},

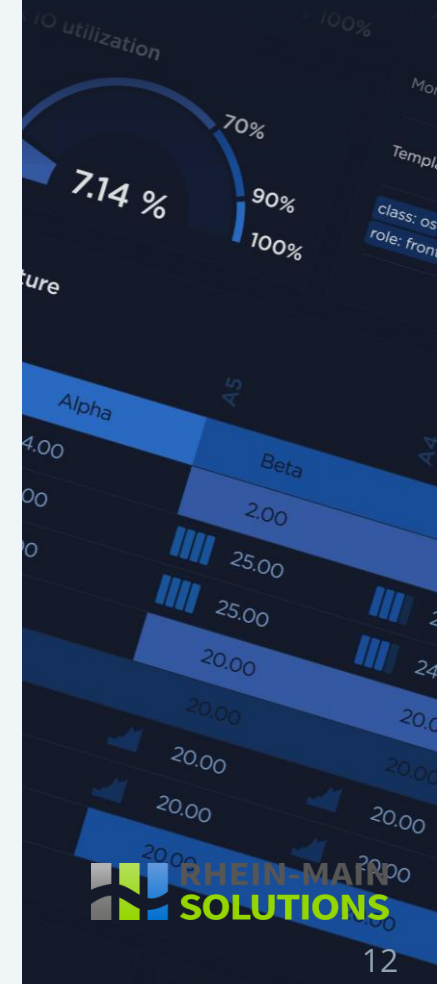
  setParams(params) {
    ['scheme', 'domain', 'width', 'height'].forEach(function (field) {
      if (typeof params !== 'object' || !params[field]) {
        throw new Error('Required param is not set: ' + field + '.');
      }
    });
    this.params = params;
  },

  getOptions(browser) {
    switch ((browser || '').trim().toLowerCase()) {
      case 'firefox':
        return Browser.firefoxOptions();
      case 'safari':
        return Browser.safariOptions();
      case 'edge':
        return Browser.edgeOptions();
      default:
        return Browser.chromeOptions();
    }
  },
};
```

```
getPerformance() {
  const browser = new Browser(Website.getOptions(Website.params.browser));
  const url = Website.params.scheme + '://' + Website.params.domain + '/' + Website.params.path;
  const screenshot = "";
  browser.setScreenSize(Number(Website.params.width), Number(Website.params.height));
  browser.navigate(url);
  browser.collectPerfEntries();
  screenshot = browser.getScreenshot();
  const result = browser.getResult();
  result.screenshot = screenshot;

  return JSON.stringify(result);
};

try {
  Website.setParams(JSON.parse(value));
  return Website.getPerformance();
} catch (error) {
  error += (String(error).endsWith('.')) ? '' : '.';
  Zabbix.log(3, ['Website get metrics'] ERROR: ' + error);
  return JSON.stringify({ 'error': error });
}
```



Automation with JavaScript

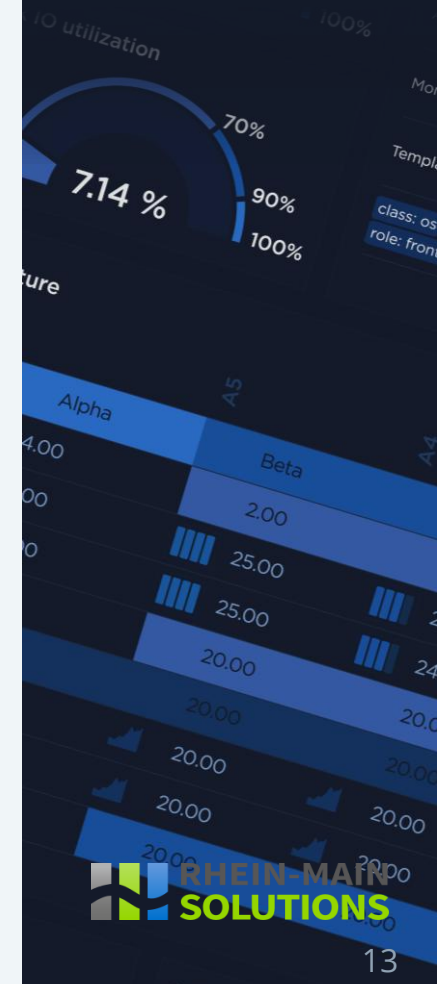
We use the Zabbix standard template „Website by Browser“

```
const browser = new Browser(Website.getOptions(Website.params.browser));
const url = Website.params.scheme + '://' + Website.params.domain + '/' + Website.params.path
const screenshot = '';
browser.setScreenSize(Number(Website.params.width), Number(Website.params.height))
browser.navigate(url);
```

// Here the magic happens

```
browser.collectPerfEntries();
screenshot = browser.getScreenshot();
const result = browser.getResult();
result.screenshot = screenshot;
```

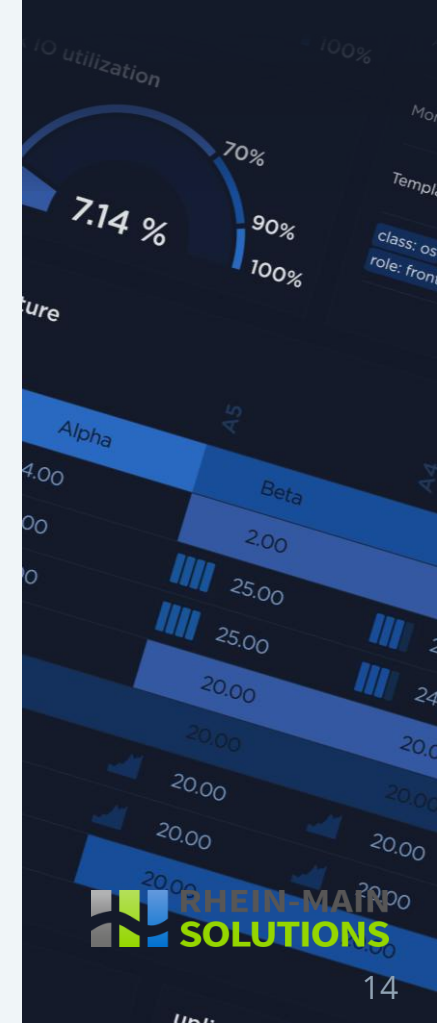
```
return JSON.stringify(result);
```



Automation with JavaScript

We can start with only three commands:

- ▶ „Browser.findElement()“ to find and select an element.
- ▶ „Element.click()“ to click.
- ▶ „Element.sendKeys()“ to enter text into input elements.



Automation with JavaScript

// Step 1: Select service

```
var select_service = browser.findElement(...);
select_service.click();
```

// Step 2: Select provider

```
var select_service = browser.findElement(...);
select_service.click();
```

// Step 3: Click "Next"

```
var button_next = browser.findElement(...);
button_next.click();
```

// Step 4: Select a date

```
var button_day = browser.findElement(...);
button_day.click();
```

// Step 5: Select a time

```
var button_time = browser.findElement(...);
button_time.click();
```

// Step 6: Click "Next"

```
button_next = browser.findElement(...);
button_next.click();
```

// Step 7: Enter your data

```
var first_name_input = browser.findElement(...);
first_name_input.sendKeys("Maria");
```

```
var last_name_input = browser.findElement(...);
last_name_input.sendKeys("Mustermann");
```

```
var email_input = browser.findElement(...);
email_input.sendKeys("m.mustermann@rhein-main-solutions.com");
```

```
var phone_number_input = browser.findElement(...);
phone_number_input.sendKeys("+49 6022 90175 00");
```

```
var address_input = browser.findElement(...);
address_input.sendKeys("Wailandtstraße 18");
```

```
var city_input = browser.findElement(...);
city_input.sendKeys("Aschaffenburg");
```

```
var zip_code_input = browser.findElement(...);
zip_code_input.sendKeys("63741");
```

// Step 8: Click "Next"

```
button_next = browser.findElement(...);
button_next.click();
```

// Step 9: Click "Confirm"

```
var button_confirm = browser.findElement(...);
button_confirm.click();
```



The screenshot displays a web browser window with two main panels. The left panel shows a web application for 'Rhein-Main SOLUTIONS Consulting 60 | Provider'. The application has a dark blue header with the company logo and name. Below the header, there's a section titled 'Service & Provider'. Under 'Service', there's a dropdown menu currently showing 'Consulting 60'. Under 'Provider', there's a dropdown menu with 'Please Select' and a list of providers: 'select#select-provider.form-select | 445.333 x 39'. Below this, there's a yellow box with the text 'Duration: 60 Minutes, Price: 29.00'. At the bottom right, there's a 'Next >' button. The footer includes 'Powered By EasyAppointments', a language selector set to 'English', and a 'Login' button.

The right panel shows the HTML code for the page. It's a Vue.js template with a dark background. The code includes a header section with the company name and logo. The main content area is a form for booking a service. It includes a dropdown for 'Service' and a dropdown for 'Provider'. The 'Provider' dropdown is currently open, showing a list of providers. The code also includes a section for 'Duration' and 'Price'. At the bottom, there's a 'Next' button. The footer includes 'Powered By EasyAppointments', a language selector, and a 'Login' button.

Michael Lindauer

Automation with JavaScript

When we know the properties of the desired element and its parents we can use Xpath to access it:

```
▼ <select id="select-provider" class="form-select"> event  
  <option value="">Please Select</option>  
  <option value="any-provider">Any Provider</option>  
  <option value="10">André Kempf</option>  
  <option value="4">Michael Lindauer</option>  
</select>
```

We want to find the option element for Michael Lindauer that is a child of the select-element with the id „select-provider“.



Automation with JavaScript

When we know the properties of the desired element and its parents we can use Xpath to access it:

```
▼ <select id="select-provider" class="form-select"> event
  <option value="">Please Select</option>
  <option value="any-provider">Any Provider</option>
  <option value="10">André Kempf</option>
  <option value="4">Michael Lindauer</option>
</select>
```

We want to find the option element for Michael Lindauer that is a child of the select-element with the id „select-provider“.

▶ `"//select[@id='select-provider']/option[contains(text(), 'Lindauer')]"`



Automation with JavaScript

// Step 1: Select service

```
var select_service = browser.findElement("xpath", "//select[@id='select-service']/option[3]");
select_service.click();
```

// Step 2: Select provider

```
var select_provider = browser.findElement("xpath", "//select[@id='select-provider']/option[contains(text(), 'Lindauer')]");
select_provider.click();
```

// Step 3: Click "Next"

```
var button_next = browser.findElement("xpath", "//button[@id='button-next-1']");
button_next.click();
```

// Step 4: Select a date

```
var button_day = browser.findElement("xpath", "//span[text()='19']");
button_day.click();
```

// Step 5: Select a time

```
var button_time = browser.findElement("xpath", "//button[text()='15:00']");
button_time.click();
```

// Step 6: Click "Next"

```
button_next = browser.findElement("xpath", "//button[@id='button-next-2']");
button_next.click();
```



ZABBIX '25
CONFERENCE
GERMANY

```
var first_name_input = browser.findElement("xpath", "//input[@id='first-name']");
first_name_input.sendKeys("Maria");
```

```
var last_name_input = browser.findElement("xpath", "//input[@id='last-name']");
last_name_input.sendKeys("Mustermann");
```

```
var email_input = browser.findElement("xpath", "//*[@id='email']");
email_input.sendKeys("m.mustermann@rhein-main-solutions.com");
```

```
var phone_number_input = browser.findElement("xpath", "//input[@id='phone-number']");
phone_number_input.sendKeys("+49 6022 90175 00");
```

```
var address_input = browser.findElement("xpath", "//*[@id='address']");
address_input.sendKeys("Wailandtstraße 18");
```

```
var city_input = browser.findElement("xpath", "//*[@id='city']");
city_input.sendKeys("Aschaffenburg");
```

```
var zip_code_input = browser.findElement("xpath", "//input[@id='zip-code']");
zip_code_input.sendKeys("63741");
```



Automation with JavaScript

// Step 8: Click "Next"

```
button_next = browser.findElement("xpath", "//*[@id='button-next-3']");  
button_next.click();
```

// Step 9: Click "Confirm"

```
var button_confirm = browser.findElement("xpath", "//*[@id='book-appointment-submit']");  
button_confirm.click();
```



Automation with JavaScript

// Step 8: Click "Next"

```
button_next = browser.findElement("xpath", "//*[@id='button-next-3']");  
button_next.click();
```

// Step 9: Click "Confirm"

```
var button_confirm = browser.findElement("xpath", "//*[@id='book-appointment-submit']");  
button_confirm.click();
```

But will it work like this?



Sadly, it will not.

RHEIN-MAIN SOLUTIONS Rhein-Main Solutions
Consulting 60 | Michael Lindauer

1 2 3 4

Appointment Date & Time

< May 2025 >

Mon	Tue	Wed	Thu	Fri	Sat	Sun
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

Timezone

UTC

14:00

14:15

14:30

14:45

15:00

< Back

Next >

Automation with JavaScript

// Step 4: Select a date

```
var button_day = browser.findElement("xpath", "//span[text()='19']");  
button_day.click();
```

// Step 5: Select a time

```
var button_time = browser.findElement("xpath", "//button[text()='15:00']");  
button_time.click();
```

This won't always work.



Automation with JavaScript

To get a valid date and time we have to:

- ▶ Select a future date
- ▶ Check if it is in this month
- ▶ Get next month and a new date
- ▶ Check if the date is valid
- ▶ Check for available time slots
- ▶ No appointment on this day?
Back to the start



Automation with JavaScript

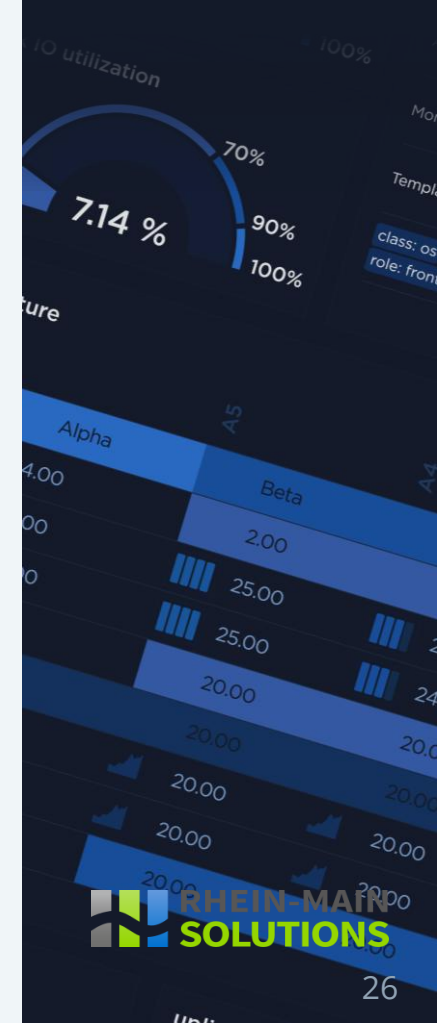
Let's rephrase that and ask an LLM to create a function for that:

```
// Step 3: Click "Next"  
var button_next = browser.findElement("xpath", "//*[@id='button-next-1']");  
button_next.click();
```

```
// Step 4 and 5: Select a valid date and time  
var timeslot = getFreeAppointment(browser);  
timeslot.click();
```

```
// Step 6: Click "Next"  
button_next = browser.findElement("xpath", "//*[@id='button-next-2']");  
button_next.click();
```

Now it's even easier.



Automation with JavaScript

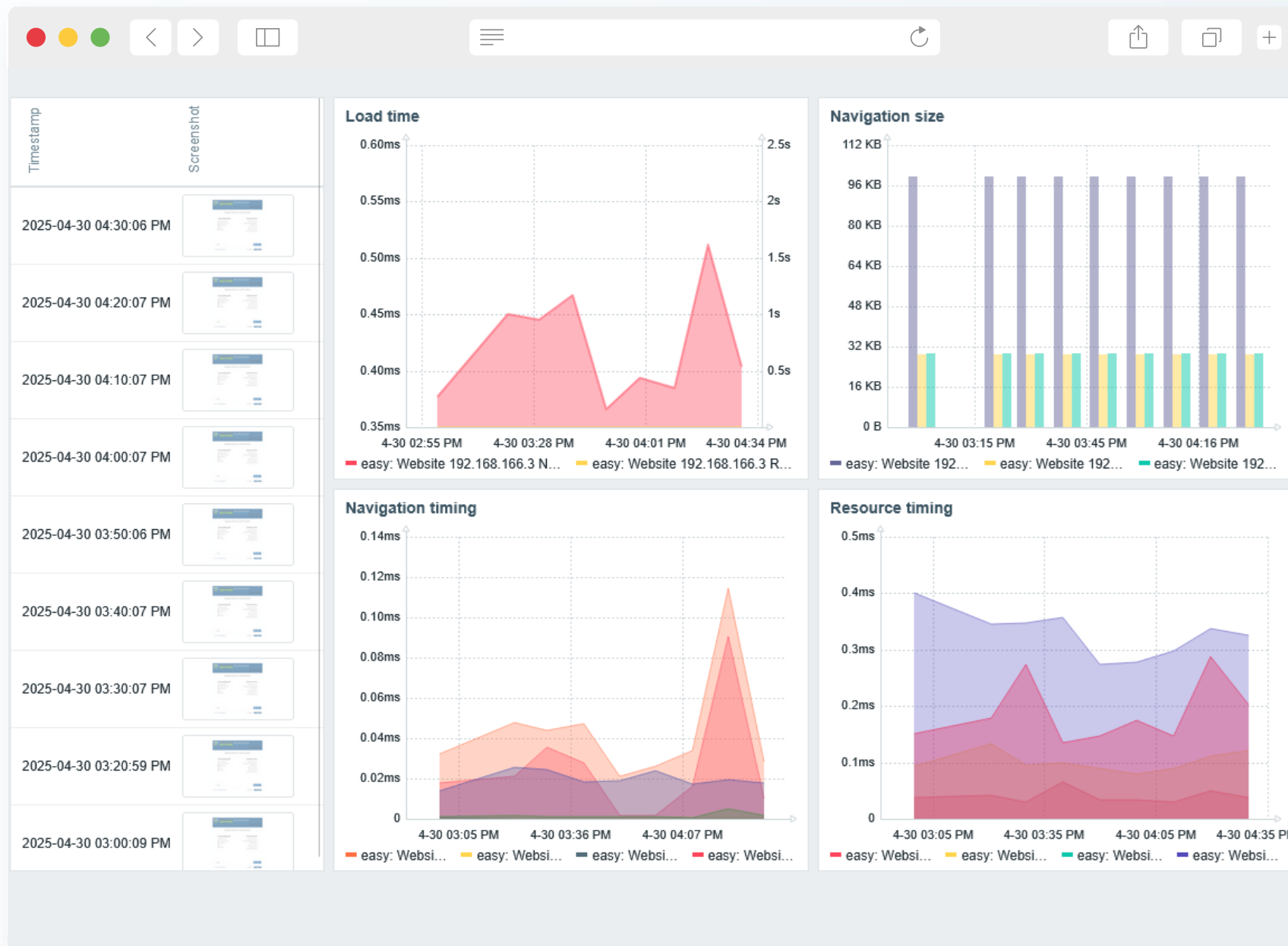
Or the easiest way:

```
// Step 3: Click "Next"  
var button_next = browser.findElement("xpath", "//*[@id='button-next-1']");  
button_next.click();
```

// Step 4 and 5: Let the website choose the next available date and time

```
// Step 6: Click "Next"  
button_next = browser.findElement("xpath", "//*[@id='button-next-2']");  
button_next.click();
```







Potential pitfalls

Potential pitfalls

On page changes Zabbix waits until elements are available.

Some websites work by changing the visibility of content. But invisible elements don't support clicking and entering data.

Solution: Check the element's attributes after a page change for things like: „display:none“



Potential pitfalls

Return a JSON with the property „error“ if something went wrong.

```
const Website = {
  params: {},

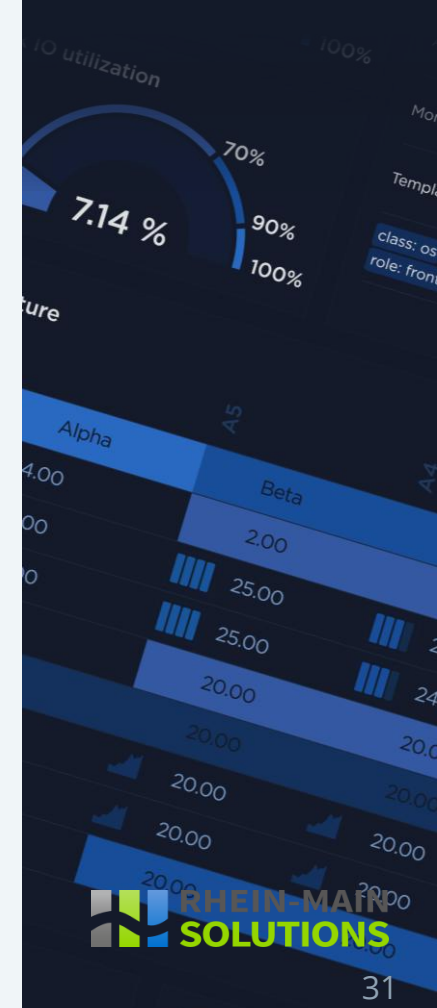
  setParams(params) {
    ['scheme', 'domain', 'width', 'height'].forEach(function (field) {
      if (typeof params !== 'object' || !params[field]) {
        throw new Error('Required param is not set: ' + field + '.');
      }
    });
    this.params = params;
  },

  getOptions(browser) {
    switch ((browser || '').trim().toLowerCase()) {
      case 'firefox':
        return Browser.firefoxOptions();
      case 'safari':
        return Browser.safariOptions();
      case 'edge':
        return Browser.edgeOptions();
      default:
        return Browser.chromeOptions();
    }
  },
```

```
  getPerformance() {
    const browser = new Browser(Website.getOptions(Website.params.browser));
    const url = Website.params.scheme + '://' + Website.params.domain + '/' + Website.params.path;
    const screenshot = "";
    browser.setScreenSize(Number(Website.params.width), Number(Website.params.height));
    browser.navigate(url);
    browser.collectPerfEntries();
    screenshot = browser.getScreenshot();
    const result = browser.getResult();
    result.screenshot = screenshot;

    return JSON.stringify(result);
  }
};

try {
  Website.setParams(JSON.parse(value));
  return Website.getPerformance();
} catch (error) {
  error += (String(error).endsWith('.')) ? '' : '.';
  Zabbix log(3, '[Website get metrics] ERROR: ' + error);
  return JSON.stringify({ 'error': error });
}
```



The screenshot shows a Zabbix web interface with a calendar view. On the left, a list of events for 'Heute' (Today) is displayed, all from 'Rhein-Main Solutions' with the message 'Ein neuer Termin ist zu Ihrer Planung hinzugefügt worden.' and various times. The right pane shows a detailed view of one such event, including a notification, the sender 'Michael Lindauer', an attached 'invitation.ics' file, and a structured data block with event details.

Ein neuer Termin ist zu Ihrer Planung hinzugefügt worden.

Rhein-Main Solutions
An: Michael Lindauer

invitation.ics
2 KB

Rhein-Main Solutions

Ein neuer Termin ist zu Ihrer Planung hinzugefügt worden

Sie können Änderungen durch Klicken auf den Termin-Link durchführen.

Terminaten

Dienstleistung	Consulting 60
Anbieter	Michael Lindauer
Beginn	06/05/2025 12:00
Ende	06/05/2025 13:00
Zeitzone	Berlin (+1:00)
Status	Booked
Beschreibung	

Kundendaten

Name	Maria Mustermann
E-Mail	m.mustermann@rhein-main-solutions.com
Telefonnummer	+49 6022 90175 00
Adresse	Wailandstraße 18

Termin-Link

<http://192.168.166.3/index.php/calendar/reschedule/MkjdvOgborJZ>

Watch out for
side effects!

ZABBIX '25

CONFERENCE

GERMANY

Thank you!



Michael Lindauer

Software Engineer
Zabbix Certified Expert
Rhein-Main Solutions



Appendix A: Links

Video: Zabbix 7 - Synthetic Web Browser Monitoring

- ▶ <https://www.youtube.com/watch?v=QHlbMKfllcY>

Zabbix Blog: An Introduction to Browser Monitoring

- ▶ <https://blog.zabbix.com/an-introduction-to-browser-monitoring/29245/>

Documentation: Monitor websites with Browser items

- ▶ https://www.zabbix.com/documentation/7.2/en/manual/guides/monitor_browser

Documentation: Browser items

- ▶ <https://www.zabbix.com/documentation/7.2/en/manual/config/items/itemtypes/browser>

Documentation: Browser item JavaScript objects

- ▶ https://www.zabbix.com/documentation/7.2/en/manual/config/items/preprocessing/javascript/browser_item_javascript_objects

Appendix B: Time selection prompt

Create a JavaScript function for Zabbix with only Duktape capabilities.

The function has an object named „browser“ as parameter and is named „getFreeAppointment“.

Element selects in this function work with xpath in the form: browser.findElement("xpath", "<the element's xpath>").

You can get an element's attribute with: Element.getAttribute(<attributeName>)

The function has the following steps in as many loops as are needed:

First, define an array that has quarter hours as strings „07:15“, „07:30“ and so on, until „15:00“.

Get the date(day of month) of the day in 7 days. Then select the element that has the date as text.

If the date also has the class „nextMonthDay“, select the span element with class „flatpickr-next-month“ and use the function .click() on it. Then select the date by text again. If the element has the class „flatpickr-disabled“ get the next day's element.

Do this until the element only has different classes. While doing this, check the month for each element again. If the element is valid, use .click() on it.

Then select the button element that has the last array entry as text and is a child of a div element with id „available-hours“. If it exists, return it.

If not, select the element that matches the previous array entry. When we arrived at the first array entry and the element, doesn't exist, select the next day, with all previous conditions for that.

Appendix C: Time selection code

```
function getFreeAppointment(browser) {  
    // Define quarter-hour intervals  
    var times = [  
        "07:15", "07:30", "07:45", "08:00", "08:15", "08:30", "08:45",  
        "09:00", "09:15", "09:30", "09:45", "10:00", "10:15", "10:30", "10:45",  
        "11:00", "11:15", "11:30", "11:45", "12:00", "12:15", "12:30", "12:45",  
        "13:00", "13:15", "13:30", "13:45", "14:00", "14:15", "14:30", "14:45", "15:00"  
    ];  
  
    function getDateOffset(offsetDays) {  
        var now = new Date();  
        now.setDate(now.getDate() + offsetDays);  
        return now;  
    }  
  
    function getDayText(date) {  
        return date.getDate().toString();  
    }  
  
    var offset = 7;  
  
    while (true) {  
        var date = getDateOffset(offset);  
        var dayText = getDayText(date);  
  
        var dateEl = browser.findElement("xpath", "//span[text()='\" + dayText + "\"]");
```

Appendix C: Time selection code

```
// If it's in the next month, go to next month and reselect
if (dateEl.getAttribute("class").indexOf("nextMonthDay") !== -1) {
    var nextMonthBtn = browser.findElement("xpath", "//span[contains(@class, 'flatpickr-next-month')]");
    nextMonthBtn.click();
    dateEl = browser.findElement("xpath", "//span[text()='\" + dayText + "\"]");
}

// Skip disabled dates
while (dateEl.getAttribute("class").indexOf("flatpickr-disabled") !== -1) {
    offset += 1;
    date = getDateOffset(offset);
    dayText = getDayText(date);

    dateEl = browser.findElement("xpath", "//span[text()='\" + dayText + "\"]");

    if (dateEl.getAttribute("class").indexOf("nextMonthDay") !== -1) {
        var nextMonthBtn = browser.findElement("xpath", "//span[contains(@class, 'flatpickr-next-month')]");
        nextMonthBtn.click();
        dateEl = browser.findElement("xpath", "//span[text()='\" + dayText + "\"]");
    }
}

// Click on the valid date
dateEl.click();
```

Appendix C: Time selection code

```
// Try time slots from latest to earliest
for (var i = times.length - 1; i >= 0; i--) {
    var timeText = times[i];
    var timeXPath = "//div[@id='available-hours']//button[text()='\" + timeText + \"']";

    try {
        var timeBtn = browser.findElement("xpath", timeXPath);
        if (timeBtn) {
            return timeBtn; // Return the button element
        }
    } catch (e) {
        // Button not found, continue
    }
} // While loop ends here

// No time available, go to next day
offset += 1;
}
}
```