1. Phiên bản 1.0 RC1 được phát hành vào ngày 16 tháng 5 năm 2006

|  |  |
| --- | --- |
| Phien bản | Ngày phát hành |
| GWT 1.0 | 17/05/2006 |
| GWT 1.1 | 11/08/2006 |
| GWT 1.2 | 16/11/2006 |
| GWT 1.3 | 05/02/2007 |
| GWT 1.4 | 18/08/2007 |
| GWT 1.5 | 27/08/2008 |
| GWT 1.6 | 07/04/2009 |
| GWT 1.7 | 13/07/2009 |
| GWT 2.0 | 08/12/2009 |
| GWT 2.1.0 | 19/10/2010 |
| GWT 2.2.0 | 11/02/2011 |
| GWT 2.3.0 | 03/05/2011 |
| GWT 2.4.0 | 08/09/2011 |
| GWT 2.5.0 | 10/2012 |
| GWT 2.5.1 | 03/2013 |
| GWT 2.6.0 | 30/01/2014 |
| GWT 2.6.1 | 10/05/2014 |
| GWT 2.7.0 | 20/11/2014 |
| GWT 2.8.1 | 24/04/2017 |
| GWT 2.8.2 | 19/11/2017 |
| GWT 2.9.0 | 02/05/2020 |
| GWT 2.10.0 | 09/06/2022 |

Vào tháng 8 năm 2010 google mua lại Instantiation chuyên cung cấp các công cụ phát triển Java Eclipse, vào năm 2011 google giwowisss thiệu ngôn ngữ lập trinh dart, google chấn an công đồng rằng GWT vẫn được hỗ trợ. Năm 2012 google cho GWT thành dự án nguồn mở.

GWT là bộ công cụ phát triển để xây dựng và tối ưu hóa các ứng dụng dựa trên trình duyệt phức tạp, mục tiêu là cho phát triển hiệu quả các ứng dụng web mà nhà phát triên không cần phải là chuyên gia về trình duyệt.

* Bên trong hộp công cụ:

GWT SDK chứa các thư viện Java API, trinh biên dịch và máy chu phát triển. Nó cho phep viết các ứng dụng từ máy khách bằng Java và triển khai dưới dạng JavaScript.

Phát triển với GWT

* Viết:

GWT SDK cung cấp một bộ Java API và Widget cốt lõi, những thư này cho phép bạn viết AJAX bằng ngôn ngư Javavà sau đó biên dịch nguồn yhanhf JavaScript, tối ưu hóa cao chạy trên tất cả các trình duyệt.

Xây dựng các ứng dụng AJAX theo cách này hiệu quả hơn nhờ mức đọ trừu tượng cao, với khái niệm phổ biến như DOM và giao tiếp XHR.

Bạn không bị giới hạn với các tiện ích đóng gói sẵn, bất kỳ điều gì bạn có thể làm với các tiện ích của DOM và JavaScript của trình duyệt đều có thể làm được tong GWT.

* Gỡ lỗi:

Bạn có thể gỡ lỗi các ứng dụngAJAX trong IDE yêu thích của mình giống như bạn làm với máy tính và trong trình duyệt, plugin dành cho nhà phát triển của GWT mở rộng mã byte Java.

Nhờ có plugin của GWT không cần biên dịch mã thành JavaScript để xem mã trong trình duyệt Bnạ có thể sử dụng cùng một chu trình làm mới chỉnh sửa xem mã, đồng thời kiểm tra các biến, đặt điểm dừng và sử dụng tất cả các công cụ gỡ lỗi.

* Tối ưu hóa:

GWT chứa hai công cụ mạnh mẽ để tạo các ứng dụng web được tối ưu hóa, trình biên dịch GWT thực hiện tối ưu hó hoàn toàn cở sở mã của bạn, các phưng thức nội tuyến, loại bỏ mã chết, tối ưu hóa chuỗi,.. Bằng cách đặt các điểm phân tách trong mã nó có thể phân tách các đoạn tải xuống thành nhiều đoạn JavaScript.

GWT không chỉ xoay quanh lập trình giao diện người dùng mà còn là một tập hợp rộng rãi các công cụ đẻ xay dựng các chức năng JavaScript.

Các dự án của GWT có thể được tổ chức theo nhiều cách khác nhau, tuy nhiên các quy ước cụ thể được khuyến khích để giúp đẽ dàng xác định mã nào sẽ chạy trên trình duyệt máy khách, máy chủ.

Trang máy chủ HTML

Các mô dun GWT được lưu trữ trên máy chủ web với một tập hợp JavaScript và các tệp có liên quan, để chạy mô dun nó phải được tải từ trang web nào đó môi một trang web nào đó cung phải bao gồm ứng dụng GWT và thẻ Script.Một trang myasy chủ HTML điển hình cho một ứng dụng được viết bằng GWT có thể bao gồm hặc không bao gồm bất kỳ nội dung HTML hiển thị nào.

<**html**>

<**head**>

*<!-- Properties can be specified to influence deferred binding -->*

<**meta** name=**'gwt:property'** content=**'locale=en\_UK'**>

*<!-- Stylesheets are optional, but useful -->*

<**link** rel=**"stylesheet"** href=**"Calendar.css"**>

*<!-- Titles are optional, but useful -->*

<**title**>Calendar App</**title**>

</**head**>

<**body**>

*<!-- This script tag is what actually loads the GWT module. The -->*

*<!-- 'nocache.js' file (also called a "selection script") is -->*

*<!-- produced by the GWT compiler in the module output directory -->*

*<!-- or generated automatically in development mode. -->*

<**script** language=**"javascript"** src=**"calendar/calendar.nocache.js"**></**script**>

*<!-- Include a history iframe to enable full GWT history support -->*

*<!-- (the id must be exactly as shown) -->*

<**iframe** src=**"javascript:''"** id=**"\_\_gwt\_historyFrame"** style=**"width:0;height:0;border:0"**></**iframe**>

</**body**>

</**html**>

Lưu ý rằng: phần thân thẻ chỉ có Script và iframe, con nội dung sẽ do GWT điền.

Nhưng GWT được thiết kế để dẽ dàng thêm các chức năng GWT vào ứng dụng web hiện có chỉ với những thay đổi nhỏ, có thể cho GWT chè có chọn lọc các widget vào nhưng vị trí cụ thể. Để làm được phải dùng các id thuộc tính trong thẻ HTML đẻ chỉ định một mã định danh duy nhất mà mã của GWT của bạn có thể đính kèm các tiêu đề tiện ích con vào phần HTML.

<**body**>

*<!-- ... other sample HTML omitted -->*

<**table** align=**center**>

<**tr**>

<**td** id=**"slot1"**></**td**>

<**td** id=**"slot2"**></**td**>

</**tr**>

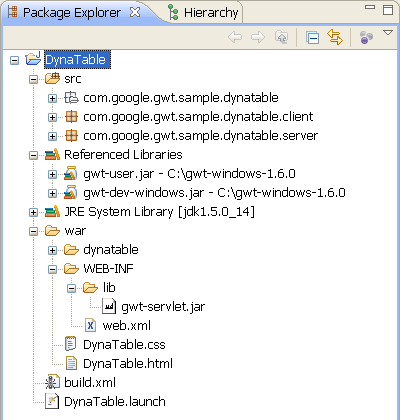
</**table**>

</**body**>

Theo cách này chức năng của GWT được thêm vào như một chức năng hiện có và việc thay đổi bố cục có thể thực hiện với HTML.

Trang HTML không nhát thiết phải là nội dung tĩnh mà có thể được thêm bằng servlet hoặc JSP.

Bố cục gói và thư mục:



Trong thư mục chính tạo ra các thư mục:

Thư mục src: chứa nguồn Java.

Thư mục war: ứng dụng web.

Thư mục test: mã kiểm tra Junit sẽ xuất hiện.

Trong gói src: tạo ra gói gốc của dự án, tạo ra gói ứng dụng khách, nếu c ó mã máy chủ thì tạo ra gói máy khách và máy chủ để phân biệt.

Trong thư mục war: đặt các tài nguyên tĩnh.

Trong các gói máy khách và máy chủ ta có thể tự do sắp xếp các gói con con mà mình cần.

1. Bố cục gói tiêu chuẩn GWT

Các tệp nguồn Java nằm trong thư mục DynaTable/src/com/gwt/sample/dynatable.

Mô-đun được xác định trong tệp XML:DynaTable/src/com/google/gwt/sample/dynatable/DynaTable.gwt.xml

Gói gốc của dự án là:com.google.gwt.sample.dynatable

Tên mô-đun logic là:com.google.gwt.sample.dynatable.DynaTable

Thư mục src chứa các tệp nguồn Java của ứng dụng

| **Bưu kiện** | **Tập tin** | **Mục đích** |
| --- | --- | --- |
| com.google.gwt.sample.dynatable |  | Gói gốc của dự án chứa các tệp XML mô-đun. |
| com.google.gwt.sample.dynatable | DynaTable.gwt.xml | Mô-đun ứng dụng của bạn. Kế thừa com.google.gwt.user.Uservà thêm một lớp điểm đầu vào, com.google.gwt.sample.dynatable.client.DynaTable. |
| com.google.gwt.sample.dynatable.public |  | Tài nguyên tĩnh được tải theo chương trình bằng mã GWT. Các tệp trong thư mục chung được sao chép vào cùng thư mục với đầu ra của trình biên dịch GWT. |
| com.google.gwt.sample.dynatable.public | logo.gif | Một tệp hình ảnh có sẵn cho mã ứng dụng. Bạn có thể tải tệp này theo chương trình bằng URL này: GWT.getModuleBaseForStaticFiles() + "logo.gif". |
| com.google.gwt.sample.dynatable.client |  | Các tệp và gói con nguồn phía máy khách. |
| com.google.gwt.sample.dynatable.client | DynaTable.java | Nguồn Java phía máy khách cho lớp điểm đầu vào. |
| com.google.gwt.sample.dynatable.client | SchoolCalendarService.java | Một giao diện dịch vụ RPC. |
| com.google.gwt.sample.dynatable.server |  | Mã phía máy chủ và các gói con. |
| com.google.gwt.sample.dynatable.server | SchoolCalendarServiceImpl.java | Nguồn Java phía máy chủ thực hiện logic của dịch vụ. |

1. **thư mục chiến tranh**

Thư mục chiến tranh là hình ảnh triển khai của ứng dụng web của bạn. Nó ở định dạng chiến tranh mở rộng tiêu chuẩn được nhiều máy chủ web Java công nhận, bao gồm Tomcat, Jetty và các thùng chứa servlet J2EE khác. Nó chứa nhiều loại tài nguyên:

* Nội dung tĩnh bạn cung cấp, chẳng hạn như trang HTML máy chủ
* Đầu ra được biên dịch GWT
* Tệp lớp Java và tệp jar cho mã phía máy chủ
* Tệp web.xml định cấu hình ứng dụng web của bạn và bất kỳ servlet nào.

| **Danh mục** | **Tập tin** | **Mục đích** |
| --- | --- | --- |
| DynaTable/war/ | DynaTable.html | Trang HTML lưu trữ tải ứng dụng DynaTable. |
| DynaTable/war/ | DynaTable.css | Biểu định kiểu tĩnh tạo kiểu cho ứng dụng DynaTable. |
| DynaTable/war/dynatable/ |  | Thư mục mô-đun DynaTable nơi trình biên dịch GWT ghi đầu ra và các tệp trên đường dẫn chung được sao chép. LƯU Ý: theo mặc định, thư mục này sẽ là tên mô-đun dài, đủ điều kiện com.google.gwt.sample.dynatable.DynaTable. Tuy nhiên, trong tệp XML mô-đun GWT của chúng tôi, chúng tôi đã sử dụng rename-to="dynatable"thuộc tính này để rút ngắn nó thành một tên đẹp. |
| DynaTable/war/dynatable/ | dynatable.nocache.js | “Tập lệnh lựa chọn” cho DynaTable. Đây là tập lệnh phải được tải từ HTML máy chủ để tải mô-đun GWT vào trang. |
| DynaTable/war/WEB-INF |  | Tất cả các tài nguyên không công khai đều có tại đây, hãy xem đặc tả servlet để biết thêm chi tiết. |
| DynaTable/war/WEB-INF | web.xml | Định cấu hình ứng dụng web của bạn và bất kỳ servlet nào. |
| DynaTable/war/WEB-INF/classes |  | Các tệp lớp được biên dịch Java trực tiếp tại đây để triển khai chức năng phía máy chủ. Nếu bạn đang sử dụng IDE, hãy đặt thư mục đầu ra thành thư mục này. |
| DynaTable/war/WEB-INF/lib |  | Mọi phụ thuộc thư viện mà mã máy chủ của bạn cần đều có tại đây. |
| DynaTable/war/WEB-INF/lib | gwt-servlet.jar | Nếu bạn có bất kỳ servlet nào sử dụng GWT RPC, bạn sẽ cần đặt một bản sao của nó gwt-servlet.jartại đây. |

#### Thư mục thử nghiệm

Thư mục kiểm tra chứa các tệp nguồn cho bất kỳ bài kiểm tra JUnit nào.

| **Bưu kiện** | **Tập tin** | **Mục đích** |
| --- | --- | --- |
| com.google.gwt.sample.dynatable.client |  | Các tệp và gói con kiểm tra phía máy khách. |
| com.google.gwt.sample.dynatable.client | DynaTableTest.java | Các trường hợp thử nghiệm cho lớp điểm đầu vào. |
| com.google.gwt.sample.dynatable.server |  | Các tệp và gói con kiểm tra phía máy chủ. |
| com.google.gwt.sample.dynatable.server | SchoolCalendarServiceImplTest.java | Các trường hợp thử nghiệm cho các lớp máy chủ. |

## Mô-đun: Đơn vị cấu hình

Các đơn vị riêng lẻ của cấu hình GWT được gọi là các mô-đun . Một mô-đun kết hợp tất cả các cài đặt cấu hình mà dự án GWT của bạn cần:

* mô-đun kế thừa
* một tên lớp ứng dụng điểm vào; đây là tùy chọn, mặc dù bất kỳ mô-đun nào được tham chiếu trong HTML phải có ít nhất một lớp điểm đầu vào được chỉ định
* mục đường dẫn nguồn
* mục đường dẫn công khai
* các quy tắc ràng buộc bị trì hoãn, bao gồm các nhà cung cấp thuộc tính và trình tạo lớp

Các mô-đun được [định nghĩa bằng XML](https://www.gwtproject.org/doc/latest/DevGuideOrganizingProjects.html#DevGuideModuleXml) và được đặt trong [hệ thống phân cấp gói của dự án](https://www.gwtproject.org/doc/latest/DevGuideOrganizingProjects.html#DevGuideDirectoriesPackageConventions) của bạn . Các mô-đun có thể xuất hiện trong bất kỳ gói nào trong đường dẫn lớp của bạn, mặc dù chúng tôi đặc biệt khuyến nghị chúng xuất hiện trong gói gốc của [bố cục dự án tiêu chuẩn](https://www.gwtproject.org/doc/latest/DevGuideOrganizingProjects.html#DevGuideDirectoriesPackageConventions) .

### Lớp đầu vào

Điểm vào mô-đun là bất kỳ lớp nào có thể gán cho [EntryPoint](https://www.gwtproject.org/javadoc/latest/com/google/gwt/core/client/EntryPoint.html) và có thể được xây dựng mà không cần tham số. Khi một mô-đun được tải, mọi lớp điểm vào được khởi tạo và phương thức [EntryPoint.onModuleLoad()](https://www.gwtproject.org/javadoc/latest/com/google/gwt/core/client/EntryPoint.html#onModuleLoad--) của nó được gọi.

### Đường dẫn nguồn

Các mô-đun có thể chỉ định gói con nào chứa nguồn có thể dịch được , làm cho gói được đặt tên và các gói con của nó được thêm vào đường dẫn nguồn . Chỉ các tệp được tìm thấy trên đường dẫn nguồn mới có thể được dịch sang JavaScript, giúp có thể kết hợp mã phía máy khách và [phía](https://www.gwtproject.org/doc/latest/DevGuideCodingBasics.html#DevGuideClientSide)[máy chủ](https://www.gwtproject.org/doc/latest/DevGuideServerCommunication.html#DevGuideServerSide) với nhau trong cùng một đường dẫn lớp mà không có xung đột. Khi mô-đun kế thừa các mô-đun khác, các đường dẫn nguồn của chúng được kết hợp để mỗi mô-đun sẽ có quyền truy cập vào nguồn có thể dịch được mà nó yêu cầu.

Đường dẫn nguồn mặc định là gói con máy khách bên dưới nơi lưu trữ [Tệp XML Mô-đun](https://www.gwtproject.org/doc/latest/DevGuideOrganizingProjects.html#DevGuideModuleXml) .

### Đường dẫn công cộng

Các mô-đun có thể chỉ định gói con nào là công khai , làm cho gói được đặt tên và các gói con của nó được thêm vào đường dẫn công cộng . Đường dẫn chung là nơi trong dự án của bạn, nơi lưu trữ các tài nguyên tĩnh được tham chiếu bởi mô-đun GWT của bạn, chẳng hạn như CSS hoặc hình ảnh. Khi bạn biên dịch ứng dụng của mình thành JavaScript, tất cả các tệp có thể tìm thấy trên đường dẫn công khai của bạn sẽ được sao chép vào thư mục đầu ra của mô-đun. Khi tham chiếu các tài nguyên công khai trong mã máy khách (ví dụ: đặt URL của một Imagetiện ích con, bạn nên tạo URL như sau: GWT.getModuleBaseForStaticFiles() + "resourceName.png". Khi tham chiếu các tài nguyên công khai từ [Tệp XML Mô-đun](https://www.gwtproject.org/doc/latest/DevGuideOrganizingProjects.html#DevGuideModuleXml), chỉ cần sử dụng đường dẫn tương đối trong thư mục chung, URL cơ sở của mô-đun sẽ tự động được thêm vào trước. Khi một mô-đun kế thừa các mô-đun khác, các đường dẫn chung của chúng được kết hợp để mỗi mô-đun sẽ có quyền truy cập vào các tài nguyên tĩnh mà nó mong đợi.

Đường dẫn công khai mặc định là thư mục con công khai bên dưới nơi lưu trữ [Tệp XML Mô-đun](https://www.gwtproject.org/doc/latest/DevGuideOrganizingProjects.html#DevGuideModuleXml) .

## Xác định mô-đun: định dạng của tệp XML mô-đun

Các mô-đun được xác định trong các tệp XML có phần mở rộng tệp là .gwt.xml . Các tệp XML mô-đun phải nằm trong gói gốc của dự án của bạn.

Nếu bạn đang sử dụng [cấu trúc dự án tiêu chuẩn](https://www.gwtproject.org/doc/latest/DevGuideOrganizingProjects.html#DevGuideDirectoriesPackageConventions) , XML mô-đun của bạn có thể đơn giản như ví dụ sau:

<**module** rename-to=**"dynatable"**>

<**inherits** name=**"com.google.gwt.user.User"** />

<**entry-point** class=**"com.google.gwt.sample.dynatable.client.DynaTable"** />

</**module**>

### tải nhiều mô-đun trong trang máy chủ HTML

Nếu bạn có nhiều mô-đun GWT trong ứng dụng của mình, có hai cách để tải chúng.

1. Biên dịch từng mô-đun riêng biệt và bao gồm từng mô-đun với một <script>thẻ riêng trong [trang chủ HTML](https://www.gwtproject.org/doc/latest/DevGuideOrganizingProjects.html#DevGuideHostPage) của bạn .
2. Tạo định nghĩa XML mô-đun cấp cao nhất bao gồm tất cả các mô-đun mà bạn muốn đưa vào. Biên dịch mô-đun cấp cao nhất để tạo một bộ đầu ra JavaScript duy nhất.

Cách tiếp cận đầu tiên có vẻ dễ dàng và rõ ràng nhất. Tuy nhiên, cách tiếp cận thứ hai sẽ dẫn đến hiệu suất của người dùng cuối tốt hơn nhiều. Vấn đề với việc tải nhiều mô-đun là mỗi mô-đun phải được trình duyệt của người dùng cuối tải xuống riêng biệt. Ngoài ra, mỗi mô-đun sẽ chứa các bản sao dự phòng của mã thư viện GWT và có thể xung đột với nhau trong quá trình xử lý sự kiện. Cách tiếp cận thứ hai được khuyến khích mạnh mẽ.

### Kiểm soát đầu ra của trình biên dịch

Trình biên dịch GWT tách hành động biên dịch và đóng gói đầu ra của nó với hệ thống con Trình liên kết. Nó chịu trách nhiệm đóng gói cuối cùng của mã JavaScript và cung cấp cơ chế bootstrap có thể cắm được cho bất kỳ kịch bản triển khai cụ thể nào.

Để biết thêm thông tin, xem [Trình liên kết](https://www.gwtproject.org/doc/latest/DevGuideLinkers.html)

### Ghi đè một triển khai gói bằng một gói khác

Thẻ <super-source>hướng dẫn trình biên dịch “root lại” một đường dẫn nguồn. Điều này hữu ích cho các trường hợp bạn muốn sử dụng lại API Java hiện có cho dự án GWT, nhưng nguồn ban đầu không có sẵn hoặc không thể dịch được. Một lý do phổ biến cho điều này là mô phỏng một phần của JRE không được GWT triển khai.

Ví dụ: giả sử bạn muốn triển khai lớp UUID do JRE cung cấp trong java.util. Giả sử tệp mô-đun dự án của bạn là com/example/myproject/MyProject.gwt.xml. Đặt nguồn cho lớp UUID vào com/example/myproject/jre/java/util/UUID.java. Sau đó thêm một dòng vào MyProject.gwt.xml:

<**super**-source path=**"jre"** />

## Làm cách nào để biết tôi cần kế thừa mô-đun GWT nào?

Các thư viện GWT được tổ chức thành các mô-đun. Các mô-đun tiêu chuẩn chứa các phần chức năng lớn được thiết kế để hoạt động độc lập với nhau. Bằng cách chỉ chọn các mô-đun bạn cần cho dự án của mình (ví dụ: mô-đun JSON chứ không phải mô-đun XML), bạn giảm thiểu độ phức tạp và giảm thời gian biên dịch.

Nói chung, bạn muốn kế thừa ít nhất mô-đun Người dùng. Mô-đun Người dùng chứa tất cả chức năng cốt lõi của GWT, bao gồm cả lớp EntryPoint. Mô-đun Người dùng cũng chứa các thành phần giao diện người dùng có thể tái sử dụng (tiện ích con và bảng điều khiển) và hỗ trợ cho tính năng Lịch sử, Quốc tế hóa, lập trình DOM, v.v.

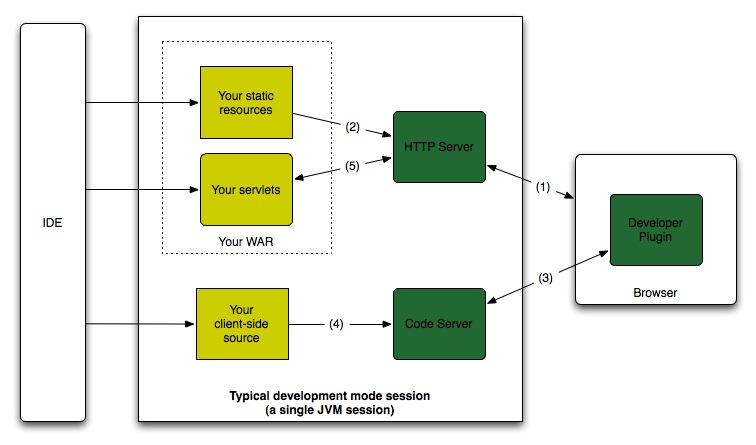
### Mô-đun tiêu chuẩn GWT 1.5

| **mô-đun** | **Tên logic** | **Định nghĩa mô-đun** | **nội dung** |
| --- | --- | --- | --- |
| Người dùng | com.google.gwt.user.User | Người dùng.gwt.xml | Chức năng GWT cốt lõi |
| HTTP | com.google.gwt.http.HTTP | HTTP.gwt.xml | Thư viện liên lạc HTTP cấp thấp |
| JSON | com.google.gwt.json.JSON | JSON.gwt.xml | Tạo và phân tích cú pháp JSON |
| JUnit | com.google.gwt.junit.JUnit | JUnit.gwt.xml | Tích hợp khung thử nghiệm JUnit |
| XML | com.google.gwt.xml.XML | XML.gwt.xml | Tạo và phân tích cú pháp tài liệu XML |

## Gỡ lỗi trong Chế độ phát triển

Bạn sẽ dành phần lớn thời gian phát triển để chạy ứng dụng của mình ở chế độ phát triển, có nghĩa là bạn đang tương tác với ứng dụng GWT của mình mà ứng dụng đó chưa được dịch sang JavaScript. Bất cứ khi nào bạn chỉnh sửa, chạy và gỡ lỗi ứng dụng từ môi trường phát triển tích hợp Java (IDE), bạn đang làm việc ở chế độ phát triển. Khi một ứng dụng đang chạy ở chế độ phát triển, Máy ảo Java (JVM) thực sự đang thực thi mã ứng dụng dưới dạng mã byte Java đã biên dịch, sử dụng hệ thống ống nước GWT để kết nối với cửa sổ trình duyệt. Điều này có nghĩa là các phương tiện sửa lỗi của IDE của bạn có sẵn để gỡ lỗi cả mã GWT phía máy khách và bất kỳ mã Java phía máy chủ nào. Bằng cách duy trì chu trình “kiểm tra mã-gỡ lỗi” truyền thống này, chế độ phát triển cho đến nay là cách hiệu quả nhất để phát triển ứng dụng của bạn một cách nhanh chóng.

Một phiên chế độ phát triển điển hình có thể được nhìn thấy bên dưới:



# Khái niệm cơ bản về mã

## Mã phía máy khách

[Mã phía máy khách](https://www.gwtproject.org/doc/latest/DevGuideCodingBasicsClient.html) mô tả cách tạo điểm vào (mã thực thi khi người dùng khởi động ứng dụng) vào ứng dụng phía máy khách. Khi ứng dụng của bạn được gửi qua mạng tới người dùng, nó sẽ chạy dưới dạng JavaScript bên trong trình duyệt web của họ.

## Khả năng tương thích với Ngôn ngữ và Thư viện Java

[Khả năng tương thích với Ngôn ngữ và Thư viện Java](https://www.gwtproject.org/doc/latest/DevGuideCodingBasicsCompatibility.html) mô tả sự khác biệt về cú pháp và ngữ nghĩa giữa GWT và ngôn ngữ Java cốt lõi. Điều quan trọng cần nhớ là ngôn ngữ đích của ứng dụng GWT của bạn cuối cùng là JavaScript, do đó, có một số khác biệt giữa việc chạy ứng dụng của bạn ở [chế độ dev-mode](https://www.gwtproject.org/doc/latest/DevGuideCompilingAndDebugging.html#DevGuideDevMode) , [superdev-mode](https://www.gwtproject.org/articles/superdevmode.html) và chế độ [sản xuất](https://www.gwtproject.org/doc/latest/DevGuideCompilingAndDebugging.html#DevGuideProdMode) .

## Môn lịch sử

[Lịch sử](https://www.gwtproject.org/doc/latest/DevGuideCodingBasicsHistory.html) mô tả cách tích hợp các ứng dụng Ajax với lịch sử trình duyệt. Các ứng dụng Ajax đôi khi không đáp ứng được kỳ vọng của người dùng vì chúng không tương tác với trình duyệt giống như các trang web tĩnh. Điều này thường rõ ràng — và gây khó chịu cho người dùng — khi một ứng dụng Ajax không tích hợp với lịch sử trình duyệt. Ví dụ: người dùng mong muốn trình duyệt có thể điều hướng quay lại các trang đã truy cập trước đó bằng các hành động quay lại và chuyển tiếp. Vì ứng dụng Ajax thường là một trang duy nhất chạy logic JavaScript chứ không phải một loạt trang, nên lịch sử trình duyệt cần có sự trợ giúp từ ứng dụng để hỗ trợ trường hợp sử dụng này. Cơ chế lịch sử của GWT giúp hỗ trợ lịch sử khá đơn giản.

## Định dạng số và ngày

[Định dạng số và ngày](https://www.gwtproject.org/doc/latest/DevGuideCodingBasicsFormatting.html) mô tả cách định dạng số và ngày trong GWT. GWT không cung cấp mô phỏng đầy đủ cho các lớp định dạng ngày và số (chẳng hạn như java.text.DateFormat, java.text.DecimalFormat, java.text.NumberFormat và java.TimeFormat). Thay vào đó, một tập hợp con chức năng của các lớp JRE được cung cấp bởi com.google.gwt.i18n.client.NumberFormat và com.google.gwt.i18n.client.DateTimeFormat. Sự khác biệt chính giữa các lớp Java tiêu chuẩn và các lớp GWT là khả năng chuyển đổi giữa các ngôn ngữ khác nhau để định dạng ngày và số trong thời gian chạy. Trong GWT, cơ chế liên kết hoãn lại được sử dụng để chỉ tải logic cần thiết cho ngôn ngữ hiện tại vào ứng dụng.

## Lập trình trễ logic

[Logic trì hoãn lập trình](https://www.gwtproject.org/doc/latest/DevGuideCodingBasicsDelayed.html) mô tả cách trì hoãn mã đang chạy cho đến một thời điểm sau đó bằng cách sử dụng ba lớp: Timer, DeferredCommand và IncrementalCommand. Điều này hữu ích để lên lịch cho một hoạt động trong một thời gian nào đó trong tương lai, định kỳ truy vấn máy chủ hoặc cập nhật giao diện, sắp xếp công việc để thực hiện công việc phải đợi quá trình khởi tạo khác kết thúc và thực hiện một lượng lớn tính toán.

## Làm việc với JSON

Nhiều nhà phát triển ứng dụng AJAX đã sử dụng JSON làm định dạng dữ liệu được lựa chọn để liên lạc với máy chủ. Đây là một định dạng tương đối đơn giản dựa trên ký hiệu đối tượng theo nghĩa đen của JavaScript. [Làm việc với JSON](https://www.gwtproject.org/doc/latest/DevGuideCodingBasicsJSON.html) giải thích cách bạn có thể sử dụng dữ liệu được mã hóa JSON trong ứng dụng của mình, GWT chứa các lớp mà bạn có thể sử dụng để phân tích cú pháp và thao tác với các đối tượng JSON, cũng như khái niệm rất hữu ích và tinh tế về Các loại lớp phủ JavaScript.

## Làm việc với XML

Ngôn ngữ đánh dấu mở rộng (XML) là một định dạng dữ liệu thường được sử dụng trong các ứng dụng web hiện đại. XML sử dụng các thẻ tùy chỉnh để mô tả dữ liệu và được mã hóa dưới dạng văn bản thuần túy, làm cho nó trở nên linh hoạt và dễ làm việc. [Làm việc với XML](https://www.gwtproject.org/doc/latest/DevGuideCodingBasicsXML.html) mô tả tập hợp các loại thư viện lớp GWT được thiết kế để xử lý dữ liệu XML.

## Giao diện gốc JavaScript (JSNI)

Thông thường, bạn sẽ cần tích hợp GWT với JavaScript viết tay hiện có hoặc với thư viện JavaScript của bên thứ ba. Đôi khi, bạn có thể cần truy cập vào chức năng trình duyệt cấp thấp không được hiển thị bởi API của lớp GWT. Tính năng [Giao diện gốc JavaScript](https://www.gwtproject.org/doc/latest/DevGuideCodingBasicsJSNI.html) (JSNI) của GWT có thể giải quyết cả hai vấn đề này bằng cách cho phép bạn tích hợp JavaScript trực tiếp vào mã nguồn Java của ứng dụng.

## Các loại lớp phủ JavaScript

Giả sử bạn đang vui vẻ sử dụng JSNI để gọi các bit JavaScript viết tay từ bên trong mô-đun GWT của mình. Nó hoạt động tốt, nhưng JSNI chỉ hoạt động ở cấp độ các phương thức riêng lẻ. Một số tình huống tích hợp yêu cầu bạn kết hợp sâu hơn các đối tượng JavaScript và Java — lập trình DOM và JSON là hai ví dụ điển hình — và vì vậy, điều chúng tôi thực sự muốn là một cách để tương tác trực tiếp với các đối tượng JavaScript từ mã nguồn Java của chúng tôi. Nói cách khác, chúng tôi muốn các đối tượng JavaScript trông giống như các đối tượng Java khi chúng tôi viết mã.

[Các loại Lớp phủ JavaScript](https://www.gwtproject.org/doc/latest/DevGuideCodingBasicsOverlay.html) giúp dễ dàng tích hợp toàn bộ họ các đối tượng JavaScript vào dự án GWT của bạn. Có rất nhiều lợi ích của kỹ thuật này, bao gồm khả năng sử dụng các khả năng tái cấu trúc và hoàn thành mã của Java IDE của bạn ngay cả khi bạn đang làm việc với các đối tượng JavaScript chưa được nhập.

## Ràng buộc hoãn lại

[Ràng buộc hoãn lại](https://www.gwtproject.org/doc/latest/DevGuideCodingBasicsDeferred.html) là một tính năng của trình biên dịch GWT hoạt động bằng cách tạo nhiều phiên bản mã tại thời điểm biên dịch, chỉ một trong số đó cần được tải bởi một máy khách cụ thể trong quá trình khởi động khi chạy. Mỗi phiên bản được tạo trên cơ sở mỗi trình duyệt, cùng với bất kỳ trục nào khác mà ứng dụng của bạn xác định hoặc sử dụng. Ví dụ: nếu bạn quốc tế hóa ứng dụng của mình bằng mô-đun Quốc tế hóa của GWT, trình biên dịch GWT sẽ tạo các phiên bản khác nhau của ứng dụng cho mỗi môi trường trình duyệt, chẳng hạn như “Firefox bằng tiếng Anh”, “Firefox bằng tiếng Pháp”, “Internet Explorer bằng tiếng Anh”, và kể từ đó trở đi. Do đó, mã JavaScript được triển khai nhỏ gọn và tải xuống nhanh hơn so với JavaScript được mã hóa thủ công, chỉ chứa mã và tài nguyên cần thiết cho một môi trường trình duyệt cụ thể.

# giao diện người dùng

Các lớp giao diện người dùng GWT tương tự như các lớp trong các khung giao diện người dùng hiện có như [Swing](http://java.sun.com/javase/6/docs/api/javax/swing/package-summary.html) và [SWT](http://www.eclipse.org/swt/) ngoại trừ việc các tiện ích con được hiển thị bằng cách sử dụng HTML được tạo động thay vì đồ họa theo định hướng pixel.

1. [Hỗ trợ nhiều trình duyệt](https://www.gwtproject.org/doc/latest/DevGuideUiBrowser.html) – Sử dụng các vật dụng và vật liệu tổng hợp để tương thích với nhiều trình duyệt

GWT giúp bạn không phải lo lắng quá nhiều về sự không tương thích giữa các trình duyệt. Nếu bạn sử [dụng tiện ích](https://www.gwtproject.org/doc/latest/DevGuideUiWidgets.html) con và [vật liệu tổng hợp tích hợp](https://www.gwtproject.org/doc/latest/DevGuideUiCustomWidgets.html) sẵn , ứng dụng của bạn sẽ hoạt động tương tự trên các phiên bản mới nhất của Internet Explorer, Firefox, Chrome và Safari. (Opera cũng vậy, hầu hết thời gian.) Tuy nhiên, giao diện người dùng HTML rất kỳ quặc, vì vậy hãy đảm bảo kiểm tra kỹ các ứng dụng của bạn trên mọi trình duyệt.

1. [Bố cục bằng bảng](https://www.gwtproject.org/doc/latest/DevGuideUiPanels.html) điều khiển – Khám phá các bảng khác nhau có sẵn để bố trí

## bảng cơ bản

### [RootPanel](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/RootPanel.html)

[RootPanel](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/RootPanel.html) là bảng trên cùng mà tất cả các vật dụng khác cuối cùng được đính kèm. [RootPanel.get()](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/RootPanel.html#get--) nhận một bảng điều khiển đơn bao bọc phần tử của tài liệu HTML <body>. Sử dụng [RootPanel.get(String id)](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/RootPanel.html#get-java.lang.String-) để nhận bảng điều khiển cho bất kỳ thành phần nào khác trên trang.

### [Bảng điều khiển lưu lượng](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/FlowPanel.html)

FlowPanel [là](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/FlowPanel.html) bảng điều khiển đơn giản nhất. Nó tạo ra một <div>phần tử duy nhất và gắn các phần tử con trực tiếp vào phần tử đó mà không cần sửa đổi. Sử dụng nó trong trường hợp bạn muốn luồng HTML tự nhiên xác định bố cục của các tiện ích con.

### [Bảng điều khiển HTML](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/HTMLPanel.html)

Bảng điều khiển này cung cấp một cách đơn giản để xác định cấu trúc HTML, trong đó các tiện ích con sẽ được nhúng tại các điểm đã xác định. Mặc dù bạn có thể sử dụng nó trực tiếp, nhưng nó được sử dụng phổ biến nhất trong các [mẫu UiBinder](https://www.gwtproject.org/doc/latest/DevGuideUiBinder.html) .

### [bảng điều khiển biểu mẫu](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/FormPanel.html)

Khi bạn cần tái tạo hành vi của biểu mẫu HTML (ví dụ: để tương tác với các máy chủ mong đợi các yêu cầu POST của biểu mẫu hoặc đơn giản là để nhận hành vi bàn phím biểu mẫu mặc định trong trình duyệt), bạn có thể sử dụng [FormPanel](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/FormPanel.html) . Bất kỳ vật dụng nào được bao bọc bởi bảng điều khiển này sẽ được bao bọc trong một <form>phần tử.

### [Bảng điều khiển cuộn](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/ScrollPanel.html)

Khi bạn muốn tạo một vùng có thể cuộn trong một bảng điều khiển khác, bạn nên sử dụng [ScrollPanel](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/ScrollPanel.html) . Bảng điều khiển này hoạt động tốt trong bảng bố cục (xem bên dưới), cung cấp cho nó kích thước rõ ràng mà nó cần để cuộn đúng cách.

### [PopupPanel](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/PopupPanel.html) và [Hộp thoại](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/DialogBox.html)

Sử dụng hai bảng này để tạo cửa sổ bật lên và hộp thoại phương thức đơn giản. Chúng chồng lên nội dung hiện có trong cửa sổ trình duyệt và có thể xếp chồng lên nhau.

### [Lưới](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Grid.html) và [FlexTable](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/FlexTable.html)

Hai tiện ích này được sử dụng để tạo các <table>thành phần HTML truyền thống và cũng có thể được sử dụng làm bảng điều khiển, trong đó các tiện ích tùy ý có thể được thêm vào các ô của chúng.

Bảng bố cục

### [RootLayoutPanel](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/RootLayoutPanel.html)

Giup lấp đầy màn hinh máy khách

DockLayoutPanel appPanel = **new** DockLayoutPanel(Unit.EM);

RootLayoutPanel.get().add(appPanel);

LayoutPanel

Widget child0, child1, child2;

LayoutPanel p = **new** LayoutPanel();

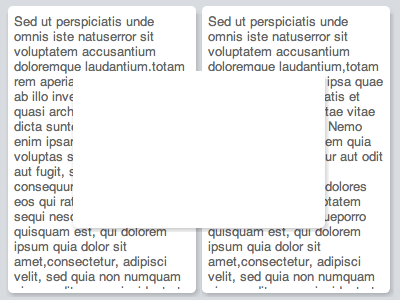
p.add(child0); p.add(child1); p.add(child2);

p.setWidgetLeftWidth(child0, 0, PCT, 50, PCT); *// Left panel*

p.setWidgetRightWidth(child1, 0, PCT, 50, PCT); *// Right panel*

p.setWidgetLeftRight(child2, 5, EM, 5, EM); *// Center panel*

p.setWidgetTopBottom(child2, 5, EM, 5, EM);



### [DockLayoutBảng điều khiển](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/DockLayoutPanel.html)

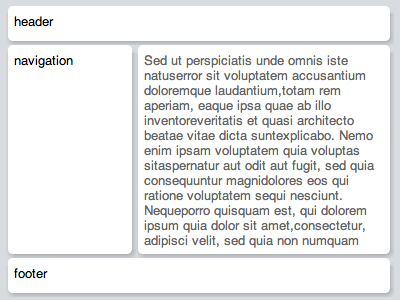
DockLayoutPanel p = **new** DockLayoutPanel(Unit.EM);

p.addNorth(**new** HTML(**"header"**), 2);

p.addSouth(**new** HTML(**"footer"**), 2);

p.addWest(**new** HTML(**"navigation"**), 10);

p.add(**new** HTML(content));



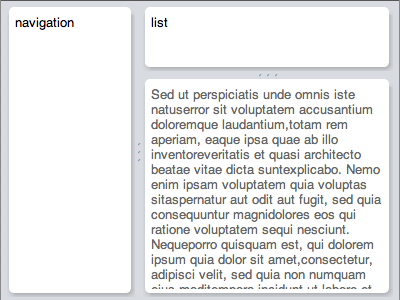
### SplitLayoutBảng điều khiển

SplitLayoutPanel p = **new** SplitLayoutPanel();

p.addWest(**new** HTML(**"navigation"**), 128);

p.addNorth(**new** HTML(**"list"**), 384);

p.add(**new** HTML(**"details"**));



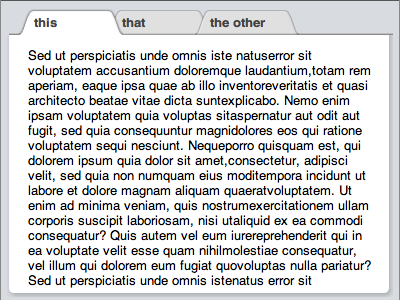
### TabBố cụcBảng điều khiển

TabLayoutPanel p = **new** TabLayoutPanel(1.5, Unit.EM);

p.add(**new** HTML(**"this content"**), **"this"**);

p.add(**new** HTML(**"that content"**), **"that"**);

p.add(**new** HTML(**"the other content"**), **"the other"**);

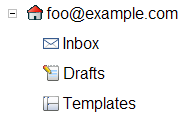
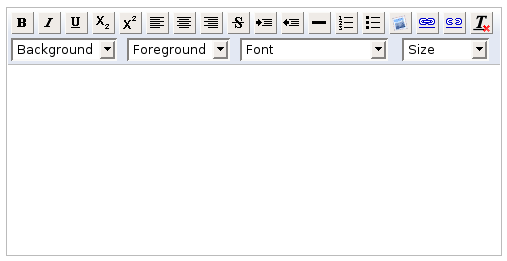


1. [Widgets](https://www.gwtproject.org/doc/latest/DevGuideUiWidgets.html) – Tạo điều khiển người dùng bằng widget

Bạn xây dựng các giao diện người dùng trong các ứng dụng GWT bằng cách sử dụng [các widget](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Widget.html) có trong [các bảng điều khiển](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Panel.html) . Widget cho phép bạn tương tác với người dùng. Các bảng kiểm soát vị trí của các phần tử giao diện người dùng trên trang. Widget và bảng điều khiển hoạt động theo cùng một cách trên tất cả các trình duyệt; bằng cách sử dụng chúng, bạn không cần phải viết mã chuyên dụng cho từng trình duyệt.

## tiện ích

Widget xác định đầu vào và đầu ra ứng dụng của bạn với người dùng. Ví dụ về các widget bao gồm:

* [Nút](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Button.html) Một người dùng nhấp vào nút chuột để kích hoạt nút.
* hình ảnh
* Hộp văn bản Ứng dụng có thể hiển thị văn bản và người dùng có thể nhập vào hộp văn bản [.](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/TextBox.html)
* hình ảnh
* [Cây](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Tree.html) Một hệ thống phân cấp có thể thu gọn của các vật dụng.
* 
* [RichTextArea](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/RichTextArea.html) Trình soạn thảo văn bản cho phép người dùng áp dụng định dạng phong phú của văn bản.
* 

Để có danh sách đầy đủ các phần tử giao diện người dùng GWT, hãy xem [Thư viện tiện ích](https://www.gwtproject.org/doc/latest/RefWidgetGallery.html) .

Bạn không bị giới hạn trong bộ tiện ích do GWT cung cấp. Có một số cách để [tạo tiện ích con tùy chỉnh](https://www.gwtproject.org/doc/latest/DevGuideUiCustomWidgets.html) :

* Bạn có thể nhóm các tiện ích hiện có lại với nhau và tạo một tiện ích Tổng [hợp](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Composite.html) .
* Bạn có thể viết các ràng buộc GWT vào một tiện ích JavaScript hiện có.
* Bạn có thể tạo tiện ích của riêng mình từ đầu bằng cách sử dụng Java hoặc JavaScript.

Bạn cũng có thể sử dụng một hoặc nhiều thư viện widget của bên thứ ba được viết cho GWT.

## phong cách

Các kiểu trực quan được áp dụng cho các widget bằng cách sử dụng [Cascading Style Sheets (CSS)](https://www.gwtproject.org/doc/latest/DevGuideUiCss.html) . Bên cạnh các định nghĩa do trình duyệt cung cấp mặc định, mỗi tiện ích và bảng điều khiển GWT đều có các định nghĩa lớp biểu định kiểu được xác định trước được ghi lại trong tài liệu tham khảo lớp.

1. [Tạo các tiện ích tùy chỉnh](https://www.gwtproject.org/doc/latest/DevGuideUiCustomWidgets.html) – Tạo các tiện ích mới, tiện ích tổng hợp hoặc tiện ích JavaScript gốc

GWT giúp dễ dàng tạo các thành phần giao diện người dùng tùy chỉnh. Có ba chiến lược chung để làm theo:

* [Tạo một tiện ích là tổng hợp](https://www.gwtproject.org/doc/latest/DevGuideUiCustomWidgets.html#composite) của các tiện ích hiện có.
* [Tạo một tiện ích hoàn toàn mới](https://www.gwtproject.org/doc/latest/DevGuideUiCustomWidgets.html#new) được viết bằng ngôn ngữ Java.
* [Tạo một tiện ích bao bọc JavaScript](https://www.gwtproject.org/doc/latest/DevGuideUiCustomWidgets.html#javascript) bằng các phương thức [JSNI .](https://www.gwtproject.org/doc/latest/DevGuideCodingBasics.html#DevGuideJavaScriptNativeInterface)

Có rất nhiều thư viện của bên thứ ba cung cấp các tiện ích mà bạn có thể tích hợp vào mô-đun GWT của mình. Các tiện ích này được tạo bằng cách sử dụng các chiến lược được liệt kê ở trên.

## vật liệu tổng hợp xây dựng

Cách hiệu quả nhất để tạo các widget mới là mở rộng lớp [Composite](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Composite.html) . Một hỗn hợp là một tiện ích chuyên dụng có thể chứa một thành phần khác (thường là [Panel](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Panel.html) ) nhưng hoạt động như thể đó là tiện ích được chứa trong nó. Bạn có thể dễ dàng kết hợp các nhóm tiện ích hiện có thành một hỗn hợp mà bản thân nó là một tiện ích có thể tái sử dụng. Một số thành phần giao diện người dùng được cung cấp trong GWT là vật liệu tổng hợp: ví dụ: [TabPanel](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/TabPanel.html) (kết hợp của TabBar và DeckPanel) và [SuggestBox](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/SuggestBox.html) .

Thay vì tạo các widget phức tạp bằng cách phân lớp con [Panel](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Panel.html) hoặc một loại [Widget](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Widget.html) khác , tốt hơn là tạo một hỗn hợp vì một hỗn hợp thường muốn kiểm soát phương thức nào có thể truy cập công khai mà không để lộ các phương thức mà nó sẽ kế thừa từ siêu lớp Panel của nó.

### Widget tổng hợp ví dụ

Đoạn mã sau đây cho biết cách tạo một tiện ích tổng hợp bao gồm tiện ích Hộp văn bản và [tiện ích](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/TextBox.html)[Hộp](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/CheckBox.html) kiểm được bố trí trong một [VerticalPanel](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/VerticalPanel.html) .

**package** com.google.gwt.examples;

**import** com.google.gwt.core.client.EntryPoint;

**import** com.google.gwt.event.dom.client.ClickHandler;

**import** com.google.gwt.user.client.ui.CheckBox;

**import** com.google.gwt.user.client.ui.Composite;

**import** com.google.gwt.user.client.ui.RootPanel;

**import** com.google.gwt.user.client.ui.TextBox;

**import** com.google.gwt.user.client.ui.VerticalPanel;

**import** com.google.gwt.user.client.ui.Widget;

**public** **class** CompositeExample **implements** EntryPoint {

*/\*\**

*\* A composite of a TextBox and a CheckBox that optionally enables it.*

*\*/*

**private** **static** **class** OptionalTextBox **extends** Composite **implements**

ClickHandler {

**private** TextBox textBox = **new** TextBox();

**private** CheckBox checkBox = **new** CheckBox();

*/\*\**

*\* Constructs an OptionalTextBox with the given caption on the check.*

*\**

*\* @param caption the caption to be displayed with the check box*

*\*/*

**public** OptionalTextBox(String caption) {

*// Place the check above the text box using a vertical panel.*

VerticalPanel panel = **new** VerticalPanel();

panel.add(checkBox);

panel.add(textBox);

*// Set the check box's caption, and check it by default.*

checkBox.setText(caption);

checkBox.setChecked(**true**);

checkBox.addClickHandler(**this**);

*// All composites must call initWidget() in their constructors.*

initWidget(panel);

*// Give the overall composite a style name.*

setStyleName(**"example-OptionalCheckBox"**);

}

**public** **void** onClick(ClickEvent event) {

Object sender = event.getSource();

**if** (sender == checkBox) {

*// When the check box is clicked, update the text box's enabled state.*

textBox.setEnabled(checkBox.isChecked());

}

}

*/\*\**

*\* Sets the caption associated with the check box.*

*\**

*\* @param caption the check box's caption*

*\*/*

**public** **void** setCaption(String caption) {

*// Note how we use the use composition of the contained widgets to provide*

*// only the methods that we want to.*

checkBox.setText(caption);

}

*/\*\**

*\* Gets the caption associated with the check box.*

*\**

*\* @return the check box's caption*

*\*/*

**public** String getCaption() {

**return** checkBox.getText();

}

}

**public** **void** onModuleLoad() {

*// Create an optional text box and add it to the root panel.*

OptionalTextBox otb = **new** OptionalTextBox(**"Check this to enable me"**);

RootPanel.get().add(otb);

}

}

## Từ đầu trong mã Java

Cũng có thể tạo một widget từ đầu, mặc dù nó phức tạp hơn vì bạn phải viết mã ở mức thấp hơn. Nhiều widget cơ bản được viết theo cách này, chẳng hạn như [Button](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Button.html) và [TextBox](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/TextBox.html) . Vui lòng tham khảo cách triển khai các tiện ích con này để hiểu cách tạo tiện ích con của riêng bạn.

Để hiểu cách tạo tiện ích của riêng bạn, hãy tham khảo cách triển khai các tiện ích này trong gói com.google.gwt.user.client.ui. Mã nguồn nằm trong gwt-user.jar.

## Sử dụng JavaScript

Khi triển khai một tiện ích tùy chỉnh xuất phát trực tiếp từ lớp cơ sở [Tiện ích](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Widget.html) , bạn cũng có thể viết một số phương thức của tiện ích bằng JavaScript. Điều này thường chỉ nên được thực hiện như là phương án cuối cùng, vì cần phải xem xét ý nghĩa của các phương thức gốc mà bạn viết trên nhiều trình duyệt và cũng trở nên khó khăn hơn để gỡ lỗi. Để biết ví dụ về mẫu này trong thực tế, hãy xem tiện ích [TextBox](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/TextBox.html) và triển khai JavaScript cơ bản của một số phương thức của nó trong lớp TextBoxImpl. Bạn nên sử dụng [ràng buộc hoãn lại](https://www.gwtproject.org/doc/latest/DevGuideCodingBasics.html#DevGuideDeferredBinding) để cô lập mã cụ thể của trình duyệt.

1. [Cell Widgets](https://www.gwtproject.org/doc/latest/DevGuideUiCellWidgets.html)Mới 2.1 – Làm việc với các widget, bảng điều khiển, DOM, sự kiện, CSS, giao diện người dùng khai báo và hình ảnh.
2. [Trình chỉnh sửa](https://www.gwtproject.org/doc/latest/DevGuideUiEditors.html)Mới 2.1 – Cho phép dữ liệu được lưu trữ trong biểu đồ đối tượng được ánh xạ lên biểu đồ Trình chỉnh sửa.
3. [Thao tác với DOM](https://www.gwtproject.org/doc/latest/DevGuideUiDom.html) – Khi cần, thao tác trực tiếp với DOM của trình duyệt
4. [Sự kiện và Trình xử lý](https://www.gwtproject.org/doc/latest/DevGuideUiHandlers.html) – Xử lý các sự kiện do widget xuất bản
5. [Làm việc với CSS](https://www.gwtproject.org/doc/latest/DevGuideUiCss.html) – Style widgets với cascading style sheet
6. [UI khai báo với UiBinder](https://www.gwtproject.org/doc/latest/DevGuideUiBinder.html) – Xây dựng cấu trúc widget và DOM từ đánh dấu XML
7. [Đóng gói tài nguyên hình ảnh](https://www.gwtproject.org/doc/latest/DevGuideUiImageBundles.html) – Tối ưu hóa tải hình ảnh bằng cách giảm số lượng yêu cầu HTTP cho hình ảnh
8. Lưu trữ HTML5

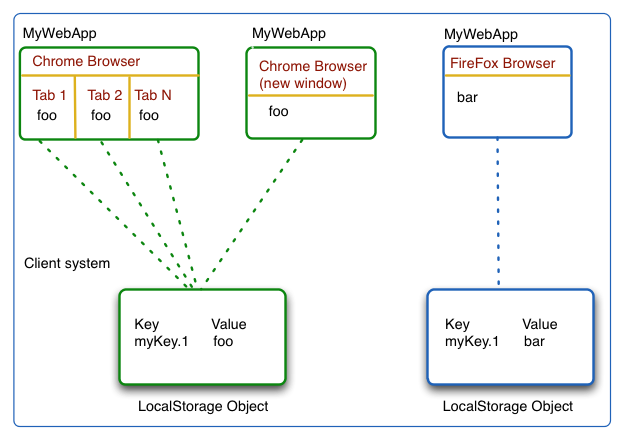
## Lưu trữ HTML5 là gì?

Thông số kỹ thuật lưu trữ HTML5 (web) là một cách được tiêu chuẩn hóa để cung cấp lượng lưu trữ phía máy khách lớn hơn và lưu trữ phiên "phân vùng" thích hợp hơn và lưu trữ liên tục cục bộ. Thông số kỹ thuật HTML5 cũng cung cấp các sự kiện lưu trữ được tạo và xử lý bởi những người nghe quan tâm. Tác động đầy đủ của các tính năng này do lưu trữ HTML5 cung cấp có thể được nhìn thấy rõ nhất bằng cách xem xét lưu trữ phía máy khách trong thế giới không phải HTML5.

Không có HTML5, lưu trữ phía máy khách cho các ứng dụng web bị giới hạn ở dung lượng lưu trữ nhỏ do cookie cung cấp (4KB mỗi cookie, 20 cookie mỗi miền) trừ khi sử dụng các sơ đồ lưu trữ độc quyền, chẳng hạn như đối tượng chia sẻ cục bộ Flash hoặc Google Gears. Nếu cookie được sử dụng, chúng cung cấp đồng thời cả lưu trữ phiên và lưu trữ liên tục cục bộ, đồng thời có thể truy cập được bằng tất cả các tab và cửa sổ trình duyệt. Cookie miền được gửi cùng với mọi yêu cầu tới miền đó, điều này sẽ tiêu tốn băng thông. “Cơ chế” xử lý cookie cũng hơi cồng kềnh.

| **Loại lưu trữ** | **Kích thước tối đa** | **kiên trì** | **Khả dụng cho các Windows/tab khác** | **Loại dữ liệu được hỗ trợ** |
| --- | --- | --- | --- | --- |
| Lưu trữ cục bộ | 5 MB mỗi ứng dụng mỗi trình duyệt. Theo thông số [HTML5](http://www.w3.org/TR/2009/WD-webstorage-20091222/#the-storage-interface) , người dùng có thể tăng giới hạn này khi cần; tuy nhiên, chỉ có một số trình duyệt hỗ trợ điều này | Trên đĩa cho đến khi bị xóa bởi người dùng (xóa bộ đệm) hoặc bởi ứng dụng | Được chia sẻ trên mọi cửa sổ và tab của một trình duyệt chạy cùng một ứng dụng web | Chỉ chuỗi, dưới dạng cặp khóa-giá trị |
| PhiênStorage | Chỉ giới hạn bởi bộ nhớ hệ thống | Chỉ tồn tại miễn là cửa sổ hoặc tab ban đầu của nó | Chỉ có thể truy cập trong cửa sổ hoặc tab đã tạo ra nó | Chỉ chuỗi, dưới dạng cặp khóa-giá trị |

### ộ nhớ cục bộ được trình duyệt chia sẻ như thế nào



|  |
| --- |
| Lớp âm thanh  * java.lang.Object   + [com.google.gwt.user.client.ui.UIObject](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/UIObject.html)     - [com.google.gwt.user.client.ui.Widget](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Widget.html)       * [com.google.gwt.user.client.ui.FocusWidget](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/FocusWidget.html)         + [com.google.gwt.media.client.MediaBase](https://www.gwtproject.org/javadoc/latest/com/google/gwt/media/client/MediaBase.html)   com.google.gwt.media.client.Audio   * **Tất cả các giao diện được triển khai:**   [HasAllDragAndDropHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasAllDragAndDropHandlers.html) , [HasAllFocusHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasAllFocusHandlers.html) , [HasAllGestureHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasAllGestureHandlers.html) , [HasAllKeyHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasAllKeyHandlers.html) , [HasAllMediaHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasAllMediaHandlers.html) , [HasAllMouseHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasAllMouseHandlers.html) , [HasAllTouchHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasAllTouchHandlers.html) , [HasBlurHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasBlurHandlers.html) , [HasCanPlayThroughHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasCanPlayThroughHandlers.html) , [HasClickHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasClickHandlers.html) , [HasDoubleClickHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasDoubleClickHandlers.html) , [HasDragEndHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasDragEndHandlers.html) , [HasDragEnterHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasDragEnterHandlers.html) , [HasDragHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasDragHandlers.html) , [HasDragLeaveHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasDragLeaveHandlers.html) , [HasDragOverHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasDragOverHandlers.html) , [HasDragStartHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasDragStartHandlers.html) , [HasDropHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasDropHandlers.html) ,[HasEndedHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasEndedHandlers.html) , [HasFocusHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasFocusHandlers.html) , [HasGestureChangeHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasGestureChangeHandlers.html) , [HasGestureEndHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasGestureEndHandlers.html) , [HasGestureStartHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasGestureStartHandlers.html) , [HasKeyDownHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasKeyDownHandlers.html) , [HasKeyPressHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasKeyPressHandlers.html) , [HasKeyUpHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasKeyUpHandlers.html) , [HasLoadedMetadataHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasLoadedMetadataHandlers.html) , [HasMouseDownHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasMouseDownHandlers.html) , [HasMouseMoveHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasMouseMoveHandlers.html) , [HasMouseOutHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasMouseOutHandlers.html) , [HasMouseOverHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasMouseOverHandlers.html) , [HasMouseUpHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasMouseUpHandlers.html) , [HasMouseWheelHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasMouseWheelHandlers.html) , [HasProgressHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasProgressHandlers.html) , [HasTouchCancelHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasTouchCancelHandlers.html) , [HasTouchEndHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasTouchEndHandlers.html) ,[HasTouchMoveHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasTouchMoveHandlers.html) , [HasTouchStartHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/dom/client/HasTouchStartHandlers.html) , [HasAttachHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/logical/shared/HasAttachHandlers.html) , [HasHandlers](https://www.gwtproject.org/javadoc/latest/com/google/gwt/event/shared/HasHandlers.html) , [EventListener](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/EventListener.html) , [Focusable](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/Focusable.html) , [HasEnabled](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/HasEnabled.html) , [HasFocus](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/HasFocus.html) , [HasVisibility](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/HasVisibility.html) , [IsWidget](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/IsWidget.html) , [SourcesClickEvents](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/SourcesClickEvents.html) , [SourcesFocusEvents](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/SourcesFocusEvents.html) , [SourcesKeyboardEvents](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/SourcesKeyboardEvents.html) , [SourcesMouseEvents](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/SourcesMouseEvents.html) |

# Bảo mật HTML an toàn

Lỗ hổng Cross-Site-Scripting (XSS) là một loại lỗi bảo mật ứng dụng web cho phép kẻ tấn công thực thi JavaScript độc hại tùy ý trong bối cảnh phiên trình duyệt của nạn nhân. Đổi lại, tập lệnh độc hại như vậy có thể lấy cắp thông tin đăng nhập phiên của người dùng (dẫn đến chiếm quyền điều khiển và xâm phạm hoàn toàn phiên của người dùng), trích xuất và rò rỉ dữ liệu nhạy cảm hoặc bí mật từ tài khoản của nạn nhân hoặc thực hiện các giao dịch do kẻ tấn công chọn dưới tên của nạn nhân.

## Nguyên tắc mã hóa

Mục tiêu của những hướng dẫn này là hai lần:

1. Một ứng dụng GWT có cơ sở mã mà các nguyên tắc này đã được áp dụng một cách nhất quán và toàn diện sẽ không có loại lỗ hổng XSS do các chuỗi do kẻ tấn công kiểm soát được đánh giá là HTML trong trình duyệt.
2. Mã phải được cấu trúc sao cho có thể dễ dàng xem lại khi không có loại XSS này – đối với mỗi lần sử dụng phương pháp có khả năng dễ bị XSS, chẳng hạn như Element.setInnerHTML"rõ ràng" rằng việc sử dụng này không thể dẫn đến lỗ hổng XSS.

Mục tiêu thứ hai dựa trên mong muốn đạt được mức độ tin cậy cao khi không có loại lỗi này. Khi xem xét mã, thường rất khó và dễ xảy ra lỗi để xác định liệu một giá trị được truyền cho một số phương thức có thể bị kiểm soát bởi kẻ tấn công hay không, đặc biệt nếu giá trị được nhận thông qua một chuỗi dài các lệnh gán và lệnh gọi phương thức.

Do đó, ý tưởng chính đằng sau các hướng dẫn này là sử dụng một loại để đóng gói các chuỗi an toàn để sử dụng trong ngữ cảnh HTML, xây dựng HTML an toàn thành các phiên bản của loại này và sử dụng loại này để "vận chuyển" các chuỗi càng gần mã càng tốt. trang web nơi chúng được sử dụng dưới dạng HTML. Việc sử dụng như vậy sau đó dễ dàng thấy rõ là không có các lỗ hổng XSS, dựa trên hợp đồng của loại (cũng như tính an toàn của loại Java) như một giả định.

## Nguyên tắc mã hóa dành cho nhà phát triển mã máy khách Widget

Các hướng dẫn sau đây nhằm vào các nhà phát triển mã máy khách sử dụng các thư viện tiện ích con hiện có, đặc biệt là thư viện tiện ích con cốt lõi được phân phối với GWT.

### Thích các tiện ích văn bản thuần túy

Cách tốt nhất để tránh các lỗi XSS và để viết mã dễ dàng nhận thấy là không có lỗ hổng XSS, đơn giản là không sử dụng các phương thức và tiện ích API diễn giải các tham số dưới dạng HTML trừ khi thực sự cần thiết.

Ví dụ: không có gì lạ khi thấy mã ứng dụng GWT, chẳng hạn như:

HTML widget = **new** HTML(**"Some text in the widget"**);

hoặc

widget**.setHTML**(someText);

Trong ví dụ đầu tiên, rõ ràng là giá trị được chuyển đến hàm HTMLtạo không thể dẫn đến lỗ hổng XSS: Giá trị không chứa đánh dấu HTML và hơn nữa là hằng số thời gian biên dịch và do đó kẻ tấn công không thể thao túng được. Trong ví dụ thứ hai, người đánh giá có thể thấy rõ rằng cuộc gọi là an toàn nếu biến someTextđược gán cho “gần đó” trong mã; tuy nhiên, nếu giá trị được cung cấp được truyền qua một tham số thông qua một vài lớp lệnh gọi phương thức thì điều này ít rõ ràng hơn nhiều. Ngoài ra, tình huống như vậy có thể dẫn đến lỗi trong lần lặp mã trong tương lai nếu lệnh gọi bị thay đổi bởi nhà phát triển không nhận ra rằng giá trị sẽ được sử dụng trong ngữ cảnh HTML.

Trong các tình huống như vậy, nên sử dụng tương đương không phải HTML, nghĩa là Labeltiện ích con và setTextphương thức, tương ứng, cả hai đều luôn an toàn trước XSS ngay cả khi chuỗi được chuyển đến hàm Labeltạo hoặc setTextphương thức nằm dưới sự kiểm soát của Tiền đạo. Tương tự, sử dụng setInnerTextinstead of setInnerHTMLtrên các phần tử DOM.

### Sử dụng UiBinder cho bố cục khai báo

Sử dụng [GWT UiBinder](https://www.gwtproject.org/doc/latest/DevGuideUiBinder.html) là cách tiếp cận ưa thích để tạo các cấu trúc Widget và DOM một cách khai báo trong các ứng dụng GWT. Ngoài các lợi ích chính của UiBinder (tách mã và bố cục rõ ràng, hiệu suất, hỗ trợ quốc tế hóa tích hợp), việc sử dụng UiBinder cũng thường dẫn đến mã ít bị lỗ hổng XSS hơn nhiều so với mã sử dụng cách tiếp cận đặc biệt để lắp ráp đánh dấu HTML để, chẳng hạn, được đặt vào một HTMLtiện ích con.

Thông thường, “các nút lá” trong bố cục do UiBinder khai báo sẽ là các widget có nội dung dữ liệu là văn bản thuần chứ không phải là đánh dấu HTML. Do đó, chúng được phổ biến một cách tự nhiên thông qua setTextchứ không phải setHTMLhoặc tương đương, điều này hoàn toàn tránh được khả năng xảy ra lỗ hổng XSS.

### Sử dụng Loại SafeHtml để đại diện cho XSS-Safe HTML

Sẽ có những lúc việc sử dụng UiBinderhoặc các cách tiếp cận tương tự không thực tế hoặc quá bất tiện. Gói com.google.gwt.safehtmlnày cung cấp các loại và lớp có thể được sử dụng để viết mã như vậy và vẫn tự tin rằng nó không có XSS.

Gói này cung cấp một giao diện, [SafeHtml](https://www.gwtproject.org/javadoc/latest/com/google/gwt/safehtml/shared/SafeHtml.html), để biểu thị tập hợp con các chuỗi an toàn để sử dụng trong ngữ cảnh HTML, theo nghĩa là việc đánh giá chuỗi dưới dạng HTML trong trình duyệt sẽ không dẫn đến việc thực thi tập lệnh. Cụ thể hơn, tất cả các triển khai của giao diện này phải tuân thủ hợp đồng loại gọi asString()phương thức trên một phiên bản sẽ luôn trả về một chuỗi an toàn cho HTML theo nghĩa trên. Ngoài ra, hợp đồng của loại yêu cầu rằng sự nối của bất kỳ hai SafeHtmlchuỗi bao bọc nào phải an toàn để sử dụng trong ngữ cảnh HTML.

Với việc giới thiệu com.google.gwt.safehtmlgói này, tất cả các tiện ích con của thư viện GWT cốt lõi nhận các Stringđối số được hiểu là HTML đã được tăng cường bằng các phương thức tương ứng nhận SafeHtmlgiá trị -typed. Đặc biệt, tất cả các tiện ích triển khai giao diện [HasHTML](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/HasHTML.html)(hoặc [HasDirectionalHtml](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/HasDirectionalHtml.html)) cũng triển khai giao diện [HasSafeHtml](https://www.gwtproject.org/javadoc/latest/com/google/gwt/safehtml/client/HasSafeHtml.html)(hoặc [HasDirectionalSafeHtml](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/HasDirectionalSafeHtml.html), tương ứng). Các giao diện này xác định:

**public** **void** setHTML(SafeHtml html);

**public** **void** setHTML(SafeHtml html, Direction dir);

là lựa chọn thay thế an toàn cho setHTML(String)và setHTML(String, Direction).

Ví dụ: [HTML](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/ui/HTML.html)tiện ích đã được tăng cường với các hàm tạo và phương thức sau:

**public** **class** HTML **extends** Label

**implements** HasDirectionalHtml, HasDirectionalSafeHtml {

*// ...*

**public** HTML(SafeHtml html);

**public** HTML(SafeHtml html, Direction dir);

@Override

**public** **void** setHTML(SafeHtml html);

@Override

**public** **void** setHTML(SafeHtml html, Direction dir);

}

Khía cạnh trọng tâm của các hướng dẫn viết mã này là các nhà phát triển ứng dụng GWT không nên sử dụng hàm tạo và phương thức có Stringtham số -typed có giá trị được hiểu là HTML mà thay vào đó hãy sử dụng giá trị SafeHtmltương đương.

### Tạo các giá trị SafeHtml

Gói safehtmlnày cung cấp một số công cụ để tạo các phiên SafeHtmlbản bao gồm nhiều tình huống phổ biến trong đó các ứng dụng GWT thường thao tác các chuỗi chứa đánh dấu HTML:

* Lớp trình [tạo](https://www.gwtproject.org/doc/latest/DevGuideSecuritySafeHtml.html#SafeHtmlBuilder) tạo điều kiện thuận lợi cho việc tạo SafeHtmlgiá trị bằng cách kết hợp an toàn các đoạn đánh dấu HTML do nhà phát triển kiểm soát với các giá trị (có thể do kẻ tấn công kiểm soát).
* Cơ [chế tạo khuôn mẫu](https://www.gwtproject.org/doc/latest/DevGuideSecuritySafeHtml.html#SafeHtmlTemplates) cho phép định nghĩa các đoạn đánh dấu HTML có cấu trúc, trong đó các giá trị được nội suy an toàn trong thời gian chạy.
* [I18NMessages](https://www.gwtproject.org/doc/latest/DevGuideI18nMessages.html#SafeHtmlMessages) có thể trả về các tin nhắn được bản địa hóa ở dạng tệp SafeHtml.
* Một số [phương thức thuận tiện](https://www.gwtproject.org/doc/latest/DevGuideSecuritySafeHtml.html#Convenience_Methods) có sẵn để tạo SafeHtmlgiá trị từ chuỗi.
* Một [trình khử trùng HTML đơn giản](https://www.gwtproject.org/doc/latest/DevGuideSecuritySafeHtml.html#SimpleHtmlSanitizer) chấp nhận một tập hợp con giới hạn của đánh dấu HTML trong đầu vào của nó và HTML thoát khỏi bất kỳ đánh dấu HTML nào không nằm trong tập hợp con đó.

Mỗi cơ chế trên đã được xem xét cẩn thận về việc tuân thủ SafeHtmlhợp đồng loại.

#### An ToànHtmlBuilder

Trong nhiều trường hợp, các chuỗi sẽ được sử dụng trong ngữ cảnh HTML được nối một phần từ các chuỗi đáng tin cậy (ví dụ: đoạn mã đánh dấu HTML được xác định trong nguồn của ứng dụng) và các chuỗi không đáng tin cậy có thể nằm dưới sự kiểm soát của kẻ tấn công tiềm năng. Lớp [SafeHtmlBuilder](https://www.gwtproject.org/javadoc/latest/com/google/gwt/safehtml/shared/SafeHtmlBuilder.html)này cung cấp giao diện trình tạo hỗ trợ trường hợp sử dụng này đồng thời đảm bảo rằng các phần không đáng tin cậy của chuỗi được thoát một cách thích hợp.

Xem xét ví dụ sử dụng này:

**public** **void** showItems(List<String> items) {

SafeHtmlBuilder builder = **new** SafeHtmlBuilder();

**for** (String item : items) {

builder.appendEscaped(item).appendHtmlConstant(**"<br/>"**);

}

itemsListHtml.setHTML(builder.toSafeHtml());

}

SafeHtmlBuilderPhương thức của ' appendHtmlConstantđược sử dụng để nối một đoạn mã HTML không đổi vào trình tạo mà không thoát khỏi đối số. Ngược lại , appendEscapedphương thức này sẽ thoát khỏi đối số chuỗi HTML của nó trước khi nối thêm.

Để cho phép SafeHtmlBuildertuân thủ SafeHtmlhợp đồng, mã sử dụng nó **phải** tuân theo các quy tắc sau:

1. Đối số của appendHtmlConstantphải là một chuỗi ký tự (hoặc tổng quát hơn, phải được xác định đầy đủ tại thời điểm biên dịch).
2. Chuỗi được cung cấp không được kết thúc trong thẻ HTML. Ví dụ: việc sử dụng sau đây sẽ là bất hợp pháp vì đối số của lần đầu tiên appendHtmlConstantchứa một <a>thẻ không đầy đủ; chuỗi kết thúc trong ngữ cảnh giá trị của hrefthuộc tính của thẻ đó:

builder.appendHtmlConstant(**"<a href='"**).appendEscaped(url).appendHtmlConstant(**"'>"**)

Quy tắc đầu tiên là cần thiết để đảm bảo rằng các chuỗi được chuyển đến appendHtmlConstantkhông thể nằm dưới sự kiểm soát của kẻ tấn công. Quy tắc thứ hai là cần thiết vì các chuỗi không đáng tin cậy được sử dụng bên trong thuộc tính thẻ HTML có thể dẫn đến việc thực thi tập lệnh ngay cả khi chúng được thoát khỏi HTML. Trong ví dụ trên, việc thực thi tập lệnh có thể xảy ra nếu giá trị của urllà javascript:evil\_js\_code().

Khi thực thi phía máy khách ở chế độ được lưu trữ hoặc phía máy chủ có bật xác nhận, hãy appendHtmlConstantphân tích cú pháp đối số của nó và kiểm tra xem nó có thỏa mãn ràng buộc thứ hai hay không. Vì lý do hiệu suất, quá trình kiểm tra này không được thực hiện ở chế độ sản xuất trong mã máy khách và với các xác nhận bị tắt trên máy chủ.

SafeHtmlBuildercũng cung cấp append(SafeHtml)phương pháp. Nội dung của phần được cung cấp SafeHtmlsẽ được thêm vào trình tạo mà không cần thoát trước (do SafeHtmlhợp đồng, nó có thể được coi là an toàn với HTML). Phương pháp này cho phép các đoạn mã HTML được bao bọc SafeHtmlđể được tổng hợp thành các SafeHtmlđoạn mã lớn hơn.

#### Tạo HTML bằng Giao diện SafeHtmlTemplates

Để tạo điều kiện thuận lợi cho việc tạo các phiên bản SafeHtml chứa đánh dấu HTML phức tạp hơn, safehtmlgói này cung cấp một cơ chế mẫu bị ràng buộc thời gian biên dịch có thể được sử dụng như trong ví dụ này:

**public** **class** MyWidget ... {

*// ...*

**public** **interface** MyTemplates **extends** SafeHtmlTemplates {

@Template(**"<span class=\"{3}\">{0}: <a href=\"{1}\">{2}</a></span>"**)

SafeHtml messageWithLink(SafeHtml message, String url, String linkText,

String style);

}

**private** **static** **final** MyTemplates TEMPLATES =

GWT.create(MyTemplates.class);

**public** **void** useTemplate(...) {

SafeHtml message;

String url;

String linkText;

String style;

*// ...*

InlineHTML messageWithLinkInlineHTML = **new** InlineHTML(

TEMPLATES.messageWithLink(message, url, linkText, style));

*// ...*

}

### Cung cấp các phương thức SafeHtml và Constructor

Các widget có hàm tạo hoặc phương thức có đối số được hiểu là HTML sẽ cung cấp các hàm tạo và phương thức tương đương nhận SafeHtmlgiá trị. Cụ thể, các tiện ích tương ứng triển khai HasHTMLhoặc HasDirectionalHtmlcũng nên triển khai HasSafeHtmlhoặc HasDirectionalSafeHtml.

### “Unwrap” SafeHtml Gần với việc sử dụng giá trị

Các giá trị được bao bọc bên trong a SafeHtmlcuối cùng sẽ được sử dụng trong ngữ cảnh HTML và, chẳng hạn, được sử dụng để đặt innerHTML.

Để làm cho việc triển khai tiện ích "rõ ràng là an toàn" nhất có thể, Stringnội dung của a SafeHtmlphải được trích xuất càng gần với mục đích sử dụng đó càng tốt. Ví dụ: một SafeHtmlgiá trị phải được mở khóa ngay lập tức trước khi nó được gán cho innerHTMLvà không sớm hơn:

element**.setInnerHTML**(safeHtml**.asString**());

Các tiện ích bao gồm các tiện ích con khác không được mở SafeHtmlcác giá trị khi khởi tạo các tiện ích con mà thay vào đó hãy chuyển các giá trị đó SafeHtmltới tiện ích con. Ví dụ: viết:

**public** **class** MyPanel **extends** HorizontalPanel {

InlineHTML messageWidget;

SafeHtml currentMessage;

**public** **void** setMessage(SafeHtml newMessage) {

currentMessage = newMessage;

updateUi();

}

**private** **void** updateUi() {

messageWidget.setHTML(currentMessage);

}

}

thay vì:

**public** **class** MyPanel **extends** HorizontalPanel {

InlineHTML messageWidget;

String currentMessage;

**public** **void** setMessage(SafeHtml newMessage) {

currentMessage = newMessage.asString();

updateUi();

}

**private** **void** updateUi() {

*// Potentially unsafe call to setHTML(String)*

messageWidget.setHTML(currentMessage);

}

}

# Bảo mật RPC XSRF

Giả mạo yêu cầu trên nhiều trang web (XSRF hoặc CSRF) là một kiểu tấn công web trong đó kẻ tấn công có thể thực hiện các hành động thay mặt cho người dùng được xác thực mà người dùng không biết. Thông thường, nó liên quan đến việc tạo một trang HTML độc hại, trang này sau khi được nạn nhân truy cập sẽ khiến trình duyệt của nạn nhân đưa ra yêu cầu do kẻ tấn công kiểm soát đối với miền của bên thứ ba. Nếu nạn nhân được xác thực với miền của bên thứ ba, yêu cầu sẽ được gửi cùng với cookie của trình duyệt cho miền đó và có khả năng kích hoạt một hành động không mong muốn thay mặt cho nạn nhân và không có sự đồng ý của nạn nhân.

## Tổng quan

Tính năng bảo vệ RPC XSRF được xây dựng bằng [RpcToken](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/rpc/RpcToken.html)tính năng cho phép nhà phát triển đặt mã thông báo trên điểm cuối RPC bằng giao diện và có mã thông báo đó đi kèm với mỗi lệnh gọi RPC được thực hiện qua điểm cuối đó.[HasRpcToken](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/rpc/HasRpcToken.html)

Triển khai bảo vệ XSRF mặc định lấy mã thông báo XSRF từ cookie xác thực phiên bằng cách tạo hàm băm MD5 của giá trị cookie phiên và sử dụng hàm băm kết quả làm mã thông báo XSRF. Việc triển khai bảo vệ XSRF không trạng thái này dựa trên thực tế là kẻ tấn công không có quyền truy cập vào cookie phiên và do đó không thể tạo mã thông báo XSRF hợp lệ.

## Thay đổi phía máy chủ

### định cấu hìnhXsrfTokenServiceServlet

Mã phía máy khách sẽ nhận mã thông báo XSRF bằng cách gọi [XsrfTokenService.getNewXsrfToken()](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/rpc/XsrfTokenService.html)triển khai phía máy chủ được định cấu hình trong web.xml:

<**servlet**>

<**servlet-name**>xsrf</**servlet-name**>

<**servlet-class**>

com.google.gwt.user.server.rpc.XsrfTokenServiceServlet

</**servlet-class**>

</**servlet**>

<**servlet-mapping**>

<**servlet-name**>xsrf</**servlet-name**>

<**url-pattern**>/gwt/xsrf</**url-pattern**>

</**servlet-mapping**>

Vì mã thông báo XSRF được liên kết với cookie phiên xác thực, nên tên của cookie đó phải được chuyển đến XsrfTokenServiceServletcũng như tất cả các máy chủ dịch vụ RPC được bảo vệ bởi XSRF. Điều này được thực hiện thông qua tham số bối cảnh trong web.xml:

<**context-param**>

<**param-name**>gwt.xsrf.session\_cookie\_name</**param-name**>

<**param-value**>JSESSIONID</**param-value**>

</**context-param**>

**Lưu ý:** Tham số khởi tạo servlet ( <init-param>) cũng có thể được sử dụng để chuyển tên của cookie phiên cho từng servlet riêng lẻ.

### Làm cho RPC servlet XSRF được bảo vệ

Tất cả các triển khai phía máy chủ của dịch vụ RPC phải mở rộng [XsrfProtectedServiceServlet](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/server/rpc/XsrfProtectedServiceServlet.html):

**package** com.example.foo.server;

**import** com.google.gwt.user.server.rpc.XsrfProtectedServiceServlet;

**import** com.example.client.MyService;

**public** **class** MyServiceImpl **extends** XsrfProtectedServiceServlet **implements**

MyService {

**public** String myMethod(String s) {

*// Do something interesting with 's' here on the server.*

**return** s;

}

}

## Thay đổi phía máy khách

### Làm cho giao diện RPC của máy khách được bảo vệ XSRF

Các giao diện RPC phía máy khách có thể được đánh dấu là được bảo vệ bằng XSRF bằng một trong các cách sau:

* bằng cách mở rộng [XsrfProtectedService](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/rpc/XsrfProtectedService.html), trong trường hợp đó, tất cả các lệnh gọi phương thức sẽ yêu cầu XsrfToken:

**package** com.example.foo.client;

**import** com.google.gwt.user.client.rpc.XsrfProtectedService;

**public** **interface** MyService **extends** XsrfProtectedService {

**public** String myMethod(String s);

}

* bằng cách chú thích rõ ràng giao diện hoặc phương thức có [@XsrfProtect](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/server/rpc/XsrfProtect.html)chú thích. [@NoXsrfProtect](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/server/rpc/NoXsrfProtect.html)chú thích có thể được sử dụng để tắt tính năng bảo vệ XSRF trên một phương thức hoặc dịch vụ để tắt tính năng bảo vệ XSRF:

package com**.example.foo.client**;

import com**.google.gwt.user.client.rpc.RemoteService**;

import com**.google.gwt.user.server.rpc.XsrfProtect**

@**XsrfProtect**

public interface MyService extends RemoteService {

public String myMethod(String s);

}

Chú thích mức phương thức ghi đè chú thích mức giao diện RPC. Nếu không có chú thích nào và giao diện RPC chứa phương thức trả về [RpcToken](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/rpc/RpcToken.html)hoặc triển khai của nó, thì xác thực mã thông báo XSRF được thực hiện trên tất cả các phương thức của giao diện đó ngoại trừ phương thức trả về RpcToken.

**Mẹo:** Để chỉ định RpcTokenGWT triển khai nào sẽ tạo bộ nối tiếp để sử dụng [@RpcTokenImplementation](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/rpc/RpcToken.RpcTokenImplementation.html)chú thích.

### Bao gồm XsrfToken với các cuộc gọi RPC

Để thực hiện cuộc gọi đến dịch vụ được bảo vệ bằng XSRF, máy khách phải có giá trị hợp lệ XsrfTokenvà đặt nó trên điểm cuối dịch vụ bằng cách chuyển giao diện không đồng bộ của dịch vụ tới phương thức [HasRpcToken](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/rpc/HasRpcToken.html)gọi setRpcToken():

XsrfTokenServiceAsync xsrf = (XsrfTokenServiceAsync)GWT.create(XsrfTokenService.class);

((ServiceDefTarget)xsrf).setServiceEntryPoint(GWT.getModuleBaseURL() + **"xsrf"**);

xsrf.getNewXsrfToken(**new** AsyncCallback<XsrfToken>() {

**public** **void** onSuccess(XsrfToken token) {

MyServiceAsync rpc = (MyServiceAsync)GWT.create(MyService.class);

((HasRpcToken) rpc).setRpcToken(token);

*// make XSRF protected RPC call*

rpc.doStuff(**new** AsyncCallback<Void>() {

*// ...*

});

}

**public** **void** onFailure(Throwable caught) {

**try** {

**throw** caught;

} **catch** (RpcTokenException e) {

*// Can be thrown for several reasons:*

*// - duplicate session cookie, which may be a sign of a cookie*

*// overwrite attack*

*// - XSRF token cannot be generated because session cookie isn't*

*// present*

} **catch** (Throwable e) {

*// unexpected*

}

});

# Các hoạt động và địa điểm của MVP

 Khung Hoạt động và Địa điểm cho phép bạn tạo các URL có thể đánh dấu trang trong ứng dụng của mình, do đó cho phép nút quay lại và dấu trang của trình duyệt hoạt động như mong đợi của người dùng. [Nó được xây dựng trên cơ chế lịch sử](https://www.gwtproject.org/doc/latest/DevGuideCodingBasicsHistory.html) của GWT và có thể được sử dụng cùng với quá trình phát triển MVP, mặc dù không bắt buộc.

Nói một cách chính xác, kiến ​​trúc MVP không liên quan đến việc quản lý lịch sử trình duyệt, nhưng Hoạt động và Địa điểm có thể được sử dụng để phát triển MVP như trong bài viết này. Nếu chưa quen với MVP, bạn có thể muốn đọc các bài viết này trước:

* [Phát triển ứng dụng quy mô lớn và MVP, Phần I](https://www.gwtproject.org/articles/mvp-architecture.html)
* [Phát triển ứng dụng quy mô lớn và MVP, Phần II](https://www.gwtproject.org/articles/mvp-architecture-2.html)

## Lượt xem

Chế độ xem chỉ đơn giản là một phần của giao diện người dùng được liên kết với Hoạt động.

**public** **interface** GoodbyeView **extends** IsWidget {

**void** setName(String goodbyeName);

}

Việc triển khai chế độ xem tương ứng mở rộng Composite, giúp giữ cho các phụ thuộc vào một Widget cụ thể không bị rò rỉ.

**public** **class** GoodbyeViewImpl **extends** Composite **implements** GoodbyeView {

SimplePanel viewPanel = **new** SimplePanel();

Element nameSpan = DOM.createSpan();

**public** GoodbyeViewImpl() {

viewPanel.getElement().appendChild(nameSpan);

initWidget(viewPanel);

}

@Override

**public** **void** setName(String name) {

nameSpan.setInnerText(**"Good-bye, "** + name);

}

}

Đây là một dạng xem phức tạp hơn một chút, nó xác định thêm một giao diện cho người trình bày (hoạt động) tương ứng của nó.

**public** **interface** HelloView **extends** IsWidget {

**void** setName(String helloName);

**void** setPresenter(Presenter presenter);

**public** **interface** Presenter {

**void** goTo(Place place);

}

}

# ghi nhật ký

## Công thức siêu đơn giản để thêm nhật ký GWT

Thêm ghi nhật ký GWT thực sự khá đơn giản, đơn giản như ví dụ mã sau đây. Tuy nhiên — việc hiểu cách thức hoạt động của quá trình ghi nhật ký và cách định cấu hình chính xác là rất quan trọng, vì vậy vui lòng dành thời gian đọc phần còn lại của tài liệu này.

*# In your .gwt.xml file*

<inherits name=**"com.google.gwt.logging.Logging"**/>

*# In your .java file*

Logger logger = Logger.getLogger(**"NameOfYourLogger"**);

logger.log(Level.SEVERE, **"this message should get logged"**);

## Loggers, Handlers và Root Logger

Loggers được tổ chức trong một cấu trúc cây, với Root Logger ở gốc của cây. Mối quan hệ Cha/Con được xác định theo tên của thiết bị ghi nhật ký, sử dụng “.” để tách các phần của tên. Vì vậy, nếu chúng ta có hai logger Foo.Bar và Foo.Baz, thì chúng là anh em ruột, với cha mẹ của chúng là logger tên là Foo. Trình ghi nhật ký Foo (và bất kỳ trình ghi nhật ký nào có tên không chứa dấu chấm “.”) có Trình ghi nhật ký gốc làm cha mẹ.

Khi bạn ghi một thông báo vào một bộ ghi nhật ký, nếu Cấp độ của thông báo đủ cao, nó sẽ chuyển thông báo đó cho cấp độ gốc của nó, cấp độ này sẽ chuyển tiếp nó cho cấp độ gốc của nó, v.v., cho đến khi đạt được Trình ghi nhật ký gốc. Đồng thời, bất kỳ trình ghi cụ thể nào (bao gồm Trình ghi nhật ký gốc) cũng sẽ chuyển thông báo tới bất kỳ Trình xử lý nào của nó và nếu Cấp độ của thông báo đủ cao, các trình xử lý đó sẽ xuất thông báo theo một cách nào đó (tới một cửa sổ bật lên, tới stderr, v.v.). Để có giải thích chi tiết hơn về vấn đề này, hãy xem <http://java.sun.com/j2se/1.4.2/docs/guide/util/logging/overview.html> .

Nếu bạn mở [LogExample.java](https://gwt.googlesource.com/gwt/+/master/samples/logexample/src/com/google/gwt/sample/logexample/client/LogExample.java), bạn có thể thấy rằng chúng tôi đã tạo 3 logger:

*// <a href="https://gwt.googlesource.com/gwt/+/master/samples/logexample/src/com/google/gwt/sample/logexample/client/LogExample.java">LogExample.java</a>*

**private** **static** Logger childLogger = Logger.getLogger(**"ParentLogger.Child"**);

**private** **static** Logger parentLogger = Logger.getLogger(**"ParentLogger"**);

**private** **static** Logger rootLogger = Logger.getLogger(**""**);

Chúng tôi đã chuyển 3 trình ghi nhật ký này vào [LoggerController](https://gwt.googlesource.com/gwt/+/master/samples/logexample/src/com/google/gwt/sample/logexample/client/LoggerController.java), do đó, tạo một phiên bản của [OneLoggerController](https://gwt.googlesource.com/gwt/+/master/samples/logexample/src/com/google/gwt/sample/logexample/client/OneLoggerController.java)cho từng trình ghi. Trong OneLoggerController.javađó, bạn có thể xem mã ví dụ để thay đổi Cấp độ của bộ ghi nhật ký, ghi nhật ký vào bộ ghi nhật ký và ghi lại một ngoại lệ đối với bộ ghi nhật ký.

// <**a** href=**"https://gwt.googlesource.com/gwt/+/master/samples/logexample/src/com/google/gwt/sample/logexample/client/OneLoggerController.java"**>OneLoggerController</**a**>

// Change the level of the logger

@UiHandler("levelTextBox")

void <**b**>handleLevelClick</**b**>(ChangeEvent e) {

Level level = Level.parse(levelTextBox.getItemText(

levelTextBox.getSelectedIndex()));

logger.log(Level.SEVERE,

"Setting level to: " + level.getName());

<**b**>logger.setLevel(level);</**b**>

}

// Log a message to the logger

@UiHandler("logButton")

void <**b**>handleLogClick</**b**>(ClickEvent e) {

Level level = Level.parse(logTextBox.getItemText(

logTextBox.getSelectedIndex()));

<**b**>logger.log(level, "This is a client log message");</**b**>

}

// Trigger an exception and log it to the logger

@UiHandler("exceptionButton")

void <**b**>handleExceptionClick</**b**>(ClickEvent e) {

try {

Level n = null;

n.getName();

} catch (NullPointerException ex) {

<**b**>logger.log(Level.SEVERE, "Null Exception Hit", ex);</**b**>

}

}

# Giao tiếp máy chủ

Tại một số thời điểm, hầu hết các ứng dụng GWT sẽ cần tương tác với máy chủ phụ trợ. GWT cung cấp một số cách khác nhau để giao tiếp với máy chủ thông qua HTTP. Bạn có thể sử dụng khung [GWT RPC](https://www.gwtproject.org/doc/latest/DevGuideServerCommunication.html#DevGuideRemoteProcedureCalls) để thực hiện các cuộc gọi đến các máy chủ Java một cách minh bạch và để GWT xử lý các chi tiết cấp thấp như tuần tự hóa đối tượng. [Ngoài ra, bạn có thể sử dụng các lớp máy khách HTTP](https://www.gwtproject.org/javadoc/latest/com/google/gwt/http/client/package-summary.html) của GWT để xây dựng và gửi các yêu cầu HTTP tùy chỉnh.

## Mã phía máy chủ

Mọi thứ xảy ra trong máy chủ web của bạn được gọi là quá trình xử lý phía máy chủ . Khi ứng dụng của bạn đang chạy trong trình duyệt của người dùng cần tương tác với máy chủ của bạn (ví dụ: để tải hoặc lưu dữ liệu), nó sẽ tạo một yêu cầu HTTP trên toàn mạng bằng cách sử dụng [lệnh gọi thủ tục từ xa (RPC)](https://www.gwtproject.org/doc/latest/DevGuideServerCommunication.html#DevGuideRemoteProcedureCalls) . Trong khi xử lý RPC, máy chủ của bạn đang thực thi mã phía máy chủ.

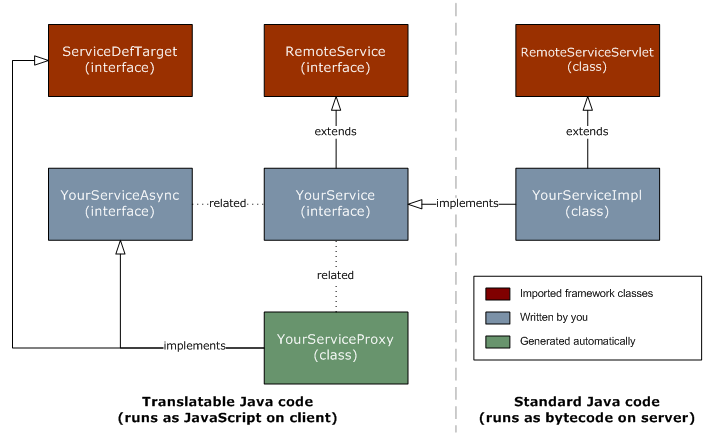
GWT cung cấp cơ chế RPC dựa trên Java Servlets để cung cấp quyền truy cập vào tài nguyên phía máy chủ. Cơ chế này bao gồm việc tạo mã phía máy khách và phía máy chủ hiệu quả để [tuần tự hóa](https://www.gwtproject.org/doc/latest/DevGuideServerCommunication.html#DevGuideSerializableTypes) các đối tượng trên mạng bằng cách sử dụng [liên kết hoãn lại](https://www.gwtproject.org/doc/latest/DevGuideCodingBasics.html#DevGuideDeferredBinding) .

## Mã phía máy chủ

Mọi thứ xảy ra trong máy chủ web của bạn được gọi là quá trình xử lý phía máy chủ . Khi ứng dụng của bạn đang chạy trong trình duyệt của người dùng cần tương tác với máy chủ của bạn (ví dụ: để tải hoặc lưu dữ liệu), nó sẽ tạo một yêu cầu HTTP trên toàn mạng bằng cách sử dụng [lệnh gọi thủ tục từ xa (RPC)](https://www.gwtproject.org/doc/latest/DevGuideServerCommunication.html#DevGuideRemoteProcedureCalls) . Trong khi xử lý RPC, máy chủ của bạn đang thực thi mã phía máy chủ.

GWT cung cấp cơ chế RPC dựa trên Java Servlets để cung cấp quyền truy cập vào tài nguyên phía máy chủ. Cơ chế này bao gồm việc tạo mã phía máy khách và phía máy chủ hiệu quả để [tuần tự hóa](https://www.gwtproject.org/doc/latest/DevGuideServerCommunication.html#DevGuideSerializableTypes) các đối tượng trên mạng bằng cách sử dụng [liên kết hoãn lại](https://www.gwtproject.org/doc/latest/DevGuideCodingBasics.html#DevGuideDeferredBinding) .

## Sơ đồ Hệ thống nước RPC



## Tạo dịch vụ

### Giao diện đồng bộ

Để bắt đầu phát triển giao diện dịch vụ mới, hãy tạo giao diện Java [phía máy khách để mở rộng giao](https://www.gwtproject.org/doc/latest/DevGuideCodingBasics.html#DevGuideClientSide)[diện](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/client/rpc/RemoteService.html) thẻ Dịch vụ Từ xa.

**package** com.example.foo.client;

**import** com.google.gwt.user.client.rpc.RemoteService;

**public** **interface** MyService **extends** RemoteService {

**public** String myMethod(String s);

}

Giao diện đồng bộ này là phiên bản cuối cùng của đặc tả dịch vụ của bạn. Mọi triển khai dịch vụ này ở phía [máy chủ](https://www.gwtproject.org/doc/latest/DevGuideServerCommunication.html#DevGuideServerSide) đều phải mở rộng [RemoteServiceServlet](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/server/rpc/RemoteServiceServlet.html) và triển khai giao diện dịch vụ này.

**package** com.example.foo.server;

**import** com.google.gwt.user.server.rpc.RemoteServiceServlet;

**import** com.example.foo.client.MyService;

**public** **class** MyServiceImpl **extends** RemoteServiceServlet **implements**

MyService {

**public** String myMethod(String s) {

*// Do something interesting with 's' here on the server.*

**return** s;

}

}

### Giao diện không đồng bộ

Bản chất của các cuộc gọi phương thức không đồng bộ yêu cầu người gọi chuyển vào một đối tượng gọi lại có thể được thông báo khi cuộc gọi không đồng bộ hoàn tất, vì theo định nghĩa, người gọi không thể bị chặn cho đến khi cuộc gọi hoàn tất. Vì lý do tương tự, các phương thức không đồng bộ không có kiểu trả về; chúng thường trả về khoảng trống. Nếu bạn muốn có nhiều quyền kiểm soát hơn đối với trạng thái của một yêu cầu đang chờ xử lý, hãy trả lại [Yêu cầu](https://www.gwtproject.org/javadoc/latest/com/google/gwt/http/client/Request.html) thay thế. Sau khi một cuộc gọi không đồng bộ được thực hiện, tất cả các giao tiếp trở lại người gọi là thông qua đối tượng gọi lại được truyền vào.

**package** com.example.foo.client;

**interface** MyServiceAsync {

**public** **void** myMethod(String s, AsyncCallback<String> callback);

}

## Dịch vụ triển khai

Cuối cùng, mọi dịch vụ đều cần thực hiện một số xử lý để đáp ứng các yêu cầu của khách hàng. Quá trình xử lý [phía máy chủ](https://www.gwtproject.org/doc/latest/DevGuideServerCommunication.html#DevGuideServerSide) như vậy xảy ra trong quá trình triển khai dịch vụ , dựa trên kiến ​​trúc [servlet](http://java.sun.com/products/servlet/) nổi tiếng . Việc triển khai dịch vụ phải mở rộng [RemoteServiceServlet](https://www.gwtproject.org/javadoc/latest/com/google/gwt/user/server/rpc/RemoteServiceServlet.html) và phải triển khai giao diện dịch vụ liên quan. Lưu ý rằng việc triển khai dịch vụ không triển khai phiên bản không đồng bộ của giao diện dịch vụ.

*<!-- Example servlet loaded into servlet container -->*

<**servlet**>

<**servlet-name**>myServiceImpl</**servlet-name**>

<**servlet-class**>

com.example.foo.server.MyServiceImpl

</**servlet-class**>

</**servlet**>

<**servlet-mapping**>

<**servlet-name**>myServiceImpl</**servlet-name**>

<**url-pattern**>/com.example.foo.Foo/myService</**url-pattern**>

</**servlet-mapping**>

## Trên thực tế Thực hiện cuộc gọi

Quá trình tạo RPC từ máy khách luôn bao gồm các bước giống nhau:

1. Khởi tạo giao diện dịch vụ bằng cách sử dụng [GWT.create()](https://www.gwtproject.org/javadoc/latest/com/google/gwt/core/client/GWT.html#create-java.lang.Class-) .
2. Tạo một đối tượng gọi lại không đồng bộ để được thông báo khi RPC hoàn thành.
3. Thuc hien cuoc goi.

### Ví dụ

Giả sử bạn muốn gọi một phương thức trên giao diện dịch vụ được định nghĩa như sau:

*// The RemoteServiceRelativePath annotation automatically calls setServiceEntryPoint()*

@RemoteServiceRelativePath(**"email"**)

**public** **interface** MyEmailService **extends** RemoteService {

**void** emptyMyInbox(String username, String password);

}

Giao diện không đồng bộ tương ứng của nó sẽ trông như thế này:

**public** **interface** MyEmailServiceAsync {

**void** emptyMyInbox(String username, String password,

AsyncCallback<Void> callback);

}

Cuộc gọi phía máy khách sẽ như thế này:

**public** **void** menuCommandEmptyInbox() {

*// (1) Create the client proxy. Note that although you are creating the*

*// service interface proper, you cast the result to the asynchronous*

*// version of the interface. The cast is always safe because the*

*// generated proxy implements the asynchronous interface automatically.*

*//*

MyEmailServiceAsync emailService = (MyEmailServiceAsync) GWT.create(MyEmailService.class);

*// (2) Create an asynchronous callback to handle the result.*

*//*

AsyncCallback callback = **new** AsyncCallback() {

**public** **void** onSuccess(Void result) {

*// do some UI stuff to show success*

}

**public** **void** onFailure(Throwable caught) {

*// do some UI stuff to show failure*

}

};

*// (3) Make the call. Control flow will continue immediately and later*

*// 'callback' will be invoked when the RPC completes.*

*//*

emailService.emptyMyInbox(fUsername, fPassword, callback);

}

Lưu trữ proxy dịch vụ đã khởi tạo vào bộ nhớ cache để tránh tạo proxy cho các cuộc gọi tiếp theo là an toàn. Ví dụ: bạn có thể khởi tạo proxy dịch vụ trong onModuleLoad()phương thức của mô-đun và lưu phiên bản kết quả dưới dạng thành viên lớp.

**public** **class** Foo **implements** EntryPoint {

**private** MyEmailServiceAsync myEmailService = (MyEmailServiceAsync) GWT.create(MyEmailService.class);

**public** **void** onModuleLoad() {

*// ... other initialization*

}

*/\*\**

*\* Make a GWT-RPC call to the server. The myEmailService class member*

*\* was initialized when the module started up.*

*\*/*

**void** sendEmail (String message) {

myEmailService.sendEmail(message, **new** AsyncCallback<String>() {

**public** **void** onFailure(Throwable caught) {

Window.alert(**"RPC to sendEmail() failed."**);

}

**public** **void** onSuccess(String result) {

label.setText(result);

}

});

}

}

# thử nghiệm

Tạo một loạt các trường hợp kiểm tra đơn vị tốt là một phần quan trọng để đảm bảo chất lượng ứng dụng của bạn trong suốt vòng đời của nó. Để hỗ trợ các nhà phát triển nỗ lực thử nghiệm, GWT cung cấp khả năng tích hợp với khung thử nghiệm đơn vị [JUnit](http://www.junit.org/) phổ biến và công cụ bảo hiểm mã [Emma .](http://emma.sourceforge.net/)GWT cho phép các trường hợp thử nghiệm JUnit chạy ở chế độ phát triển hoặc chế độ sản xuất.

### Phía khách hàng Ví dụ

Trước hết, bạn sẽ cần một mô-đun GWT hợp lệ để lưu trữ lớp trường hợp thử nghiệm của mình. Thông thường, bạn không cần tạo [tệp XML mô-đun](https://www.gwtproject.org/doc/latest/DevGuideOrganizingProjects.html#DevGuideModuleXml) mới - bạn chỉ cần sử dụng tệp bạn đã tạo để phát triển mô-đun GWT của mình. Nhưng nếu bạn chưa có mô-đun, bạn có thể tạo một mô-đun như sau:

<**module**>

*<!-- Module com.example.foo.Foo -->*

*<!-- Standard inherit. -->*

<**inherits** name=**'com.google.gwt.user.User'**/>

*<!-- implicitly includes com.example.foo.client package -->*

*<!-- OPTIONAL STUFF FOLLOWS -->*

*<!-- It's okay for your module to declare an entry point. -->*

*<!-- This gets ignored when running under JUnit. -->*

<**entry-point** class=**'com.example.foo.FooModule'**/>

*<!-- You can also test remote services during a JUnit run. -->*

<**servlet** path=**'/foo'** class=**'com.example.foo.server.FooServiceImpl'**/>

</**module**>

trong com.example.foo.client(hoặc bất kỳ gói con nào) đều có thể chia sẻ com.example.foo.Foomô-đun.

Giả sử bạn đã tạo một tiện ích con trong foogói, UpperCasingLabelđể đảm bảo rằng văn bản mà nó hiển thị hoàn toàn là chữ hoa. Đây là cách bạn có thể kiểm tra nó.

**package** com.example.foo.client;

**import** com.google.gwt.junit.client.GWTTestCase;

**public** **class** UpperCasingLabelTest **extends** GWTTestCase {

*/\*\**

*\* Specifies a module to use when running this test case. The returned*

*\* module must include the source for this class.*

*\**

*\* @see com.google.gwt.junit.client.GWTTestCase#getModuleName()*

*\*/*

@Override

**public** String getModuleName() {

**return** **"com.example.foo.Foo"**;

}

**public** **void** testUpperCasingLabel() {

UpperCasingLabel upperCasingLabel = **new** UpperCasingLabel();

upperCasingLabel.setText(**"foo"**);

assertEquals(**"FOO"**, upperCasingLabel.getText());

upperCasingLabel.setText(**"BAR"**);

assertEquals(**"BAR"**, upperCasingLabel.getText());

upperCasingLabel.setText(**"BaZ"**);

assertEquals(**"BAZ"**, upperCasingLabel.getText());

}

}

## Kết hợp các lớp TestCase thành một TestSuite

**public** **class** MapsTestSuite **extends** GWTTestSuite {

**public** **static** Test suite() {

TestSuite suite = **new** TestSuite(**"Test for a Maps Application"**);

suite.addTestSuite(MapTest.class);

suite.addTestSuite(EventTest.class);

suite.addTestSuite(CopyTest.class);

**return** suite;

}

}

## Thiết lập và chia nhỏ các trường hợp thử nghiệm JUnit sử dụng mã GWT

Khi sử dụng phương thức kiểm tra trong JUnit [TestCase](http://junit.sourceforge.net/junit3.8.1/javadoc/junit/framework/TestCase.html) , bất kỳ đối tượng nào mà bài kiểm tra của bạn tạo và để lại tham chiếu sẽ vẫn hoạt động. Điều này có thể ảnh hưởng đến các phương pháp thử nghiệm trong tương lai. Bạn có thể ghi đè hai phương pháp mới để chuẩn bị và/hoặc dọn dẹp sau mỗi phương pháp thử nghiệm.

**import** com.google.gwt.junit.client.GWTTestCase;

**import** com.google.gwt.user.client.DOM;

**import** com.google.gwt.user.client.Element;

**private** **static** **native** String getNodeName(Element elem) */\*-{*

*return (elem.nodeName || "").toLowerCase();*

*}-\*/*;

*/\*\**

*\* Removes all elements in the body, except scripts and iframes.*

*\*/*

**public** **void** gwtSetUp () {

Element bodyElem = RootPanel.getBodyElement();

List<Element> toRemove = **new** ArrayList<Element>();

**for** (**int** i = 0, n = DOM.getChildCount(bodyElem); i < n; ++i) {

Element elem = DOM.getChild(bodyElem, i);

String nodeName = getNodeName(elem);

**if** (!**"script"**.equals(nodeName) &amp;&amp; !**"iframe"**.equals(nodeName)) {

toRemove.add(elem);

}

}

**for** (**int** i = 0, n = toRemove.size(); i < n; ++i) {

DOM.removeChild(bodyElem, toRemove.get(i));

}

}

# triển khai

Việc triển khai một ứng dụng GWT rất đơn giản và dễ dàng. Đầu ra được tạo bởi trình biên dịch chỉ đơn giản là một vài tệp JavaScript và HTML, cùng với các tài nguyên công khai khác (css, hình ảnh, v.v.). Tất cả những gì bạn cần làm để triển khai ứng dụng của mình là đặt các tài nguyên này trên máy chủ web của bạn.

## Triển khai trên máy chủ web

Việc triển khai ứng dụng GWT cho máy chủ web rất đơn giản. Tất cả những gì bạn cần làm là sao chép các [tệp ứng dụng GWT đã tạo](https://www.gwtproject.org/doc/latest/DevGuideCompilingAndDebugging.html#DevGuideJavaToJavaScriptCompiler) sau khi biên dịch và lưu trữ chúng trên máy chủ web của bạn. Tất nhiên, bạn cũng sẽ cần thiết lập mã phía máy chủ của mình và quá trình thiết lập này có thể có một số dạng khác nhau: giao tiếp qua [JSONP](https://www.gwtproject.org/javadoc/latest/com/google/gwt/jsonp/client/JsonpRequestBuilder.html) và [làm việc với dữ liệu JSON](https://www.gwtproject.org/doc/latest/DevGuideCodingBasics.html#DevGuideJSON) , tập lệnh phía máy chủ nhận và phân tích cú pháp các yêu cầu HTTP được gửi qua GWT [RequestBuilder](https://www.gwtproject.org/javadoc/latest/com/google/gwt/http/client/RequestBuilder.html) hoặc GWT RPC

# tối ưu hóa

Khi ứng dụng của bạn đã hoạt động cơ bản, đã đến lúc cải thiện hiệu suất của nó. Bạn có thể sử dụng Công cụ theo dõi tốc độ để tìm hiểu ứng dụng của mình đang hoạt động như thế nào và bạn có thể sử dụng một số công cụ để giải quyết các vấn đề về hiệu suất cụ thể mà bạn tìm thấy.

## Tách mã

Khi một ứng dụng AJAX phát triển, phần JavaScript của nó có xu hướng phát triển, cuối cùng đến mức tải xuống và cài đặt mã JavaScript sẽ làm tăng thêm thời gian khởi động của ứng dụng. [Bộ tách mã của GWT](https://www.gwtproject.org/doc/latest/DevGuideCodeSplitting.html) có thể tăng tốc độ khởi động của ứng dụng bằng cách cho phép ứng dụng bắt đầu chạy trước khi tất cả mã của nó được cài đặt.

## Biên soạn báo cáo

Khi lập trình trong GWT, đôi khi có thể khó hiểu đầu ra được biên dịch. Điều này đặc biệt đúng đối với người dùng [Chia mã](https://www.gwtproject.org/doc/latest/DevGuideCodeSplitting.html) : tại sao một số đoạn lớn hơn, một số nhỏ hơn? Câu trả lời của chúng tôi cho những câu hỏi này là [Biên dịch báo cáo](https://www.gwtproject.org/doc/latest/DevGuideCompileReport.html) . Báo cáo biên dịch cho phép các lập trình viên GWT hiểu rõ hơn về những gì xảy ra trong ứng dụng của họ trong quá trình biên dịch: mã của họ dẫn đến bao nhiêu đầu ra, gói và lớp Java nào dẫn đến đầu ra JavaScript lớn và cách mã được phân tách trong quá trình Tách mã. Được trang bị thông tin này, các lập trình viên sau đó có thể sửa đổi ứng dụng của họ theo cách được nhắm mục tiêu để giảm kích thước của toàn bộ ứng dụng đã biên dịch hoặc kích thước của một số đoạn nhất định.

## Gói khách hàng

Các tài nguyên trong ứng dụng GWT đã triển khai có thể được phân loại đại khái thành các tài nguyên không bao giờ lưu vào bộ nhớ cache ( .nocache.js), lưu vào bộ nhớ cache mãi mãi ( .cache.html) và mọi thứ khác ( myapp.css). [Gói ứng dụng khách](https://www.gwtproject.org/doc/latest/DevGuideClientBundle.html) cho phép bạn di chuyển tài nguyên từ danh mục mọi thứ khác sang danh mục bộ đệm vĩnh viễn.

## Số liệu nhẹ

Hệ thống [Số liệu nhẹ](https://www.gwtproject.org/doc/latest/DevGuideLightweightMetrics.html) là một công cụ để tìm các khu vực chính mà độ trễ có thể được người dùng cuối của bạn chú ý. Nó có rất ít chi phí hoạt động, có thể báo cáo số liệu về thời gian tải ứng dụng và lệnh gọi RPC, bạn có thể lập cấu hình nhiều mô-đun GWT cùng lúc và có thể mở rộng cho nhu cầu đo lường của riêng bạn. Bảng gỡ lỗi cho GWT sử dụng hệ thống số liệu Trọng lượng nhẹ. Nó cung cấp một cách dễ dàng để thu thập số liệu và kiểm tra ứng dụng GWT của bạn.

Tài liệu tham khảo

<https://www.gwtproject.org/doc/latest/DevGuide.html>

<https://www.gwtproject.org/javadoc/latest/com/google/gwt/media/client/Audio.html>

<https://support.google.com/code/answer/54830?hl=en>

<https://en.wikipedia.org/wiki/Google_Web_Toolkit>

<https://gist.github.com/lannp/f19d911e7cf81b3a8b3c723eae6a5b8c>

<https://www.tutorialspoint.com/gwt/index.htm#>