1. **Tóm tắt luận văn**
   1. Vấn đề nghiên cứu
      1. Xây dựng Web Service , cung cấp các chuẩn API ứng dụng cho các ứng dụng kiểm tra chính tả tiếng Việt.
      2. Xây dựng một số Extension ứng dụng Web Service xây dựng được cho việc kiểm tra chính tả tiếng Việt.
   2. Cách tiếp cận
      1. Thuật toán :
         1. Khóa luận này sử dụng thuật toán và source code có sẵn trong việc kiểm tra lỗi chính tả của thầy Cường phát triển theo mô hình ngôn ngữ N-gram.
         2. Mô hình N-gram là một mô hình xác xuất sử dụng cho việc dự đoán phẩn tử tiếp theo trong dãy tuần tự các đối tượng. Các đối tượng được sử dụng chủ yếu ở đây là âm tiết và dấu câu.
      2. Công nghệ :
         1. Web service với Spring Boot với Intellij.
         2. Chrome Extension
   3. Kết quả
      1. **RESTful WebService**
         1. Xây dựng thành công web service từ thuật toán của thầy Lê Anh Cường với những API cơ bản sau:
      2. **Chrome Extension**
         1. Xây dựng thành công 2 extension kiểm tra lỗi chính tả sử dụng web service:
         2. Extension check lỗi chính tả trên nội dung của một trang, khi load một trang web bất kỳ.
         3. Extension kiểm tra lỗi chính tả ngay khi bạn gõ một đoạn text , đoạn mail hay làm một văn bản trên nền web.
      3. Khả năng ứng dụng cũng như độ chính xác trong việc kiểm tra chính tả có thể được cải thiện , cải tiến thêm dựa vào việc chúng ta tiếp tục tìm hiểu và nghiên cứu thêm về những mô hình , thuật toán xử lý ngôn ngữ tự nhiên , kiểm tra chính tả khác để có thể đạt được nhiều hơn những kết quả khả quan trong việc kiểm tra chính tả tiếng việt, từ đó mang lại giá trị to lớn, mở rộng hơn cho đề tài khóa luận này trong tương lai.
2. Mở đầu
   1. Trong tất cả các phương tiện mà con người sử dụng để giao tiếp thì ngôn ngữ là phương tiện duy nhất thoả mãn được tất cả các nhu cầu của con người. Phương tiện giao tiếp ấy được bổ sung và hoàn thiện dần theo lịch sử tiến hoá của nhân loại, theo những trào lưu và xu hướng tiếp xúc văn hoá có từ cổ xưa đến tận ngày nay. Ngày nay, hầu như không còn ngôn ngữ nào là chưa có ảnh hưởng của nền văn hoá ngoại lai. Nói cách khác, tất cả các ngôn ngữ đang tồn tại hiện nay đều đã từng trải qua những quá trình tiếp xúc văn hoá với ngôn ngữ khác bên ngoài.
   2. Một trong những chức năng đặc biệt quan trọng của ngôn ngữ đó là là hiện thực trực tiếp của tư tưởng , cầu nối giúp mọi người có thể giao tiếp được, hiểu được nhau. Và cũng chính vì vậy việc sai lỗi chính tả luôn là vấn đề nhức nhối đi kèm với ứng dụng to lớn của ngôn ngữ.
   3. Chính tả là cách viết chữ được coi là chuẩn, nó có tính chất bắt buộc đối với toàn thể cộng đồng ngôn ngữ. Sự thống nhất chính tả biểu hiện tính thống nhất của một ngôn ngữ. Chuẩn chính tả, cũng như mọi chuẩn ngôn ngữ khác, không phải là cái gì nhất thành, bất biến. Có những chuẩn cũ đã trở thành lỗi thời, nhường chỗ cho những chuẩn mới ra đời hoặc thay thế nếu nó đáp ứng nhu cầu của đời sống, phù hợp với sự phát triển của ngôn ngữ và được xã hội chấp nhận. Chính tả tiếng Việt cũng không nằm ngoài cái quy luật đó.
   4. Đặc biệt trong thời đại công nghệ thông tin như hiện nay thì việc ngôn ngữ hầu hết được dùng , chuyển thể sang dạng viết , văn bản hơn là thuần giao tiếp giữa người với người như trước kia. Vậy vấn đề ở đây là gì? Liệu tất cả mọi người đều thực sự chắc chắn về khả năng chính tả của mình. Việc học và ngôn ngữ của con người bị ảnh hưởng trực tiếp và ảnh hưởng lẫn nhau dẫn tới việc sai chính tả không ít thì nhiều là hoàn toàn dễ hiểu.
   5. Có những trường hợp , ngữ cảnh thì việc sai chính tả có thể được coi nhẹ và được bỏ qua bởi người đọc. Tuy nhiên không phải tất cả các trường hợp , còn rất nhiều trường hợp đặc biệt không nên hoặc không được phép sai chính tả nhưng không phải ai cũng đủ cẩn thận , đủ kiên nhẫn và thời gian có thể kiểm tra lại chính tả cho chính bản thân mình.
   6. Vấn đề này đã phần nào được giải quyết bởi những công cụ kiểm tra chính tả tự động được ra đời như trong bộ Microsoft Office.
   7. Nhận thấy tầm quan trọng và ứng dụng cần thiết của việc kiểm tra chính tả đặc biệt hiện nay là việc kiểm tra chính tả online , trực tiếp trên nền web như viết mail , chat hay là kiểm tra chính tả của một văn bản bất kì mà ta quan tâm. Đó chính làm mục đích mà luận văn này hướng tới đó là xây dựng nên được Web service với cung cấp các chuẩn API ứng dụng cho các ứng dụng kiểm tra chính tả tiếng Việt. Cùng với đó là xây dựng nên 2 Extension cơ bản sử dụng web service kiểm tra chính tả xây dựng được là giúp mọi người kiểm tra được chính tả của các văn bản , bài báo quan tâm cũng như là kiểm tra chính tả cho cho chính bản thân mình như viết mail, chat chit, …
3. Web service là gì?
   1. Là những thành phần ứng dụng dùng để chuyển đổi một ứng dụng thông thường sang một ứng dụng web. Đồng thời nó cũng xuất bản các chức năng của mình để mọi người dùng internet trên thế giới đều có thể sử dụng thông qua nền tảng web. Web Serviece truyền thông bằng cách sử dụng các giao thức mở, tài nguyên phần mềm có thể xác định bằng địa chỉ URL, thực hiện các chức năng và đưa ra các thông tin người dùng yêu cầu, các ứng dụng độ lập và tự mô tả chính nó. Nó bao gồm các modun độc lập cho hoạt động của khác hàng và doanh nghiệp và bản thân nó được thực thi trên server. Nền tảng cơ bản của WS là XML + HTTP. Bất cứ một ứng dụng nào cũng đều có thể có một thành phần WS. WS có thể được tạo ra bằng bất kỳ một ngôn ngữ lập trình nào.
   2. Đặc điểm:
      1. Cho phép client và server tương tác ngay cả trong môi trường khác nhau. (Ví dụ server chạy linux, client chạy windows).
      2. Phần lớn được xây dựng dựa trên mã nguồn mở và phát triển các chuẩn đã được công nhận.
      3. Nó có thể triển khai bởi một phần mềm ứng dụng phía server (Ví dụ như PHP, Oracle Application server, Microsoft .NET, SpringBoot,…)
      4. Mặc dù có nhiều ưu điểm : Có nhiều mô-đun và có thể công bố lên internet, khả năng hoạt động rộng lớn với phần mềm khác nhau chạy nền tảng khác nhau, sử dụng các giao thức chuẩn mở, nâng cao khả năng tái sử dụng, cho phép các tiền trình / chức năng nghiệp vụ đóng gói trong giao diện dịch vụ web, dễ dàng cho việc phát triển các ứng dụng phân tán. Nhưng những thiệt hại lớn sẽ xảy ra vào khoảng thời gian chết của dịch vụ web, có quá nhiều chuẩn cho dịch vụ web khiến người dùng khó nắm bắt, phải quan tâm nhiều hơn tới vấn đề an toàn và bảo mật.
4. Tại sao lại sử dụng WebService:

Một vài năm trước, WS không được nhiều người chú ý đến vì tốc độ của nó không đủ nhanh.

* Khả năng tương tác được ưu tiên cao nhất:

Trong khi hầu hất các nền tảng có thể truy cập web bằng các trình duyệt web thì một số khác lại không thể tương thích được. Và để các nền tảng này có thể làm việc được với nhau, các ứng dụng web được ra đời. Ứng dụng web là các ứng dụng đơn giản chạt trên web. Chứng được xây dựng dựa trên các tiêu chuẩn về trình duyệt web và có thể được sử dụng dởi bất cứ một trình duyệt hay một nền tảng nào.

* WS nâng tầm các ứng dụng web:

Bằng các sử dụng WS, những ứng dụng của chúng ta có thể xuất bản các chứ năng cũng như nội dung của nó đến mọi người dùng internet trên thế giới. WS sử dụng XMLđể mã hóa và giải mã dữ liệu, và dùng SOAP để chuyển tải các dữ liệu này (bằng các sử dụng các giao thức mở). WS có thể kết nối Windows server và UNIX server lại với nhau.

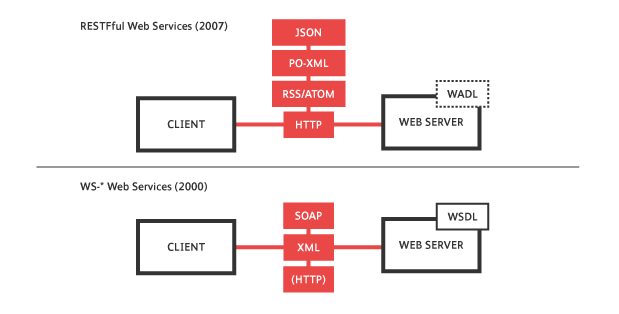
* WS có hai cách dùng:
  1. Tái sử dụng các thành phần ứng dụng:

Có nhiều lý do khiến một ứng dụng được dùng thường xuyên. Vì thế làm cách nào để một ứng dụng có thể được sử dụng lại nhiều lần? WS có thể cung cấp các thành phần iwngs dụng như: chuyển đổi các đơn vị tiền tệ, dự báo thời tiết, hoặc thậm chí là việc dịch ngôn ngữ.

* 1. Kết nối các phần mềm sẵn có:

WS có thể giải quyết vấn đề tương tác giữa các ứng dụng khác nhau bằng cách cho phép các ứng dụng này liên kết các dữ liệu của chúng lại với nhau. Với WS, chúng ta có thể trao đổi dữ liệu giữa các ứng dụng và các nền tảng khác nhau.

Ngày nay, người ta thường dùng hai giải pháp webservice chính để giao tiếp với ứng dụng web là: SOAP (Simple Object Access Protocol) và REST (Representational State Transfer). Cải hai giải pháp này đều là những lựa chọn tốt khi thiết kế hệ thống, nhưng nó cũng những ưu điểm và nhược điểm của riêng.



Để có thể xem xét và lựa chọn giải pháp nào phù hợp cho mục đích sử dụng trước tiên chúng ta phải hiểu được khái niệm và bản chất của 2 công nghệ này:

|  |  |  |
| --- | --- | --- |
| STT | SOAP | REST |
| 1 | SOAP là một giao thức | REST là một cách thiết kế kiến trúc |
|  | SOAP là từ viết tắt của Simple Object Access Protocol(Giao thức truy cập đối tượng đơn giản) | REST viết tắt của REpresentational State Transfer |
|  | SOAP không thể dùng REST bởi vì nó chỉ là một phương thức. | REST có thể dùng các web services sử dụng SOAP vì nó có thể dùng bất kỳ giao thức nào như HTTP, SOAP |
|  | SOAP cung cấp các giao diện dịch vụ(services interfaces) cho các thành phần bên ngoài sử dụng | REST sủ dụng đỉa chỉ URI để cung cấp các dịch vụ |
|  | JAX-WS là java API cài đặt web services theo giao thức SOAP | JAX-RS là java API cài đặt web services theo kiến trúc RESTful |
|  | SOAP định nghĩa các chuẩn và quy tắc chặt chẽ | REST không định nghĩa nhiều chuẩn như SOAP |
|  | SOAP sử dụng băng thông và tài nguyên nhiều hơn REST | REST sử dụng băng thông và tài nguyên ít hơn SOAP |
|  | SOAP định nghĩa chuẩn bảo mật của riêng nó | RESTful kế thừa chuẩn bảo mật tầng vận tải của giao thức mạng |
|  | SOAP chỉ hỗ trợ định dạng dữ liệu XML | REST hỗ trợ các định dạng dữ liệu khác nhau như text, HTML, XML, JSON |
|  | SOAP ít được dùng hơn REST | REST được ưa chuộng hơn SOAP |
|  | Được thiết kế để dùng trong tính toán phân tán | Thương không được dùng trong môi trường tính toán phân tán |
|  | Tin cậy hơn | Ít tin cậy hơn – chẳng hạn, HTTP DELETE có thể trả về trạng thái OK ngay cả khi tài nguyên không được xóas |
|  | Hỗ trợ hầu hết các chuẩn bảo mật, tin cậy và giao dịch | Sử dụng tốt với các giao thức như: HTTP, SSL. Các phương thức DELETE và PUT thường bị vô hiệu hóa bởi tường lửa hoặc vấn đề bảo mật |
|  | SOAP hỗ trợ cả hai giao thức SMTP và HTTP | REST gắn với giao thức HTTP |

Trong phạm vi khóa luận , dựa vào những tiêu chí như tính ứng dụng, sự phổ biến , quy thức chuẩn , dễ dàng và tiện lợi cho sử dụng , rõ ràng trong việc thiết kế ứng dụng thì RESTful đã được lựa chọn để xây dựng Webservice.

**Vậy thì REST là gì?**

REST định nghĩa các quy tắc kiến trúc để bạn thiết kế Web services chú trọng vào tài nguyên hệ thống, bao gồm các trạng thái tài nguyên được định dạng như thế nào và được chuyển tải qua HTTP thông qua số lượng lớn người dùng và được viết bởi những ngôn ngữ khác nhau.

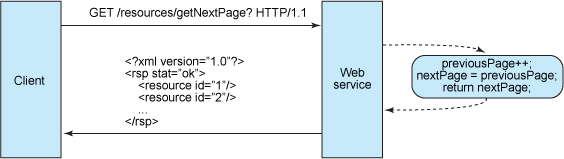
1. Sử dụng các phương thức chuẩn: Mọi resource đều hỗ trợ một số các hành động mà HTTP gọi là các verbs. Các phương thức chuẩn vào gồn PUT, DELETE, HEAD và OPTIONS. Ý nghĩa của các phương pháp này được định nghĩa trong đặc tả HTTP, cùng với một số bảo đảm về hành vi của chúng. Bạn có thể tưởng tượng rằng mọi resource trong RESTfull HTTP được mở rộng từ class:
   * Phương thức GET: Sử dụng GET theo cách này rất rõ ràng vì GET chỉ dành cho truy cập dữ liệu. GET không thay đổi giá trị của resource.
   * Phương thức PUT: Sử dụng để cập nhật dữ liệu của resource. Dữ liệu được cập nhật bằng cách xác định resource bằng URI. Nếu không tồn tại resource nó sẽ tạo ra một resource mới.
   * Phương thức DELETE dùng để xóa resource: xóa resource bởi URI.
   * Phương thức POST: tạo ra một resource mới.
2. **Resources với nhiều đại diện**
   * Làm thế nào một client biết làm thế nào để giải quyết được với các dữ liệu nó lấy được trả về, ví dụ như kết quả của một yêu cầu GET hoặc POST? Phương pháp của HTTP là cho phép một sự tách biệt giữa các mối quan tâm xử lý dữ liệu và gọi hành động. Nói cách khác, một client mà biết làm thế nào để tiếp cận một dạng dữ liệu đặc biệt có thể tương tác với tất cả các resource mà có thể cung cấp một đại diện trong định dạng này.
3. Giao tiếp phi trạng thái:

Các Web service REST cần được điều chỉnh về quy mô để đáp ứng được các yêu cầu ngày càng cao về chất lượng thực hiện. Các khu vực lưu trữ của máy chủ với khả năng cân bằng tải và vượt qua sự mất mát, các bức ngăn (tường lửa) và các cổng được sắp xếp theo một phương thức đặc thù nhằm tạo ra một cấu trúc dịch vụ bền vững cho phép chuyển tiếp yêu cầu từ một máy chủ tới máy chủ khác khi cần để giảm tổng thời gian phản hồi của một yêu cầu Web service. Sử dụng máy chủ trung gian nhằm nâng cao mức yêu cầu dịch vụ mạng REST của khách hàng để gửi các yêu cầu hoàn chỉnh và độc lập, có nghĩa là gửi các yêu cầu bao gồm tất cả dữ liệu cần thiết để đáp ứng sao cho các thành phần trong các máy chủ trung gian có thể gửi tiếp đi, gửi theo tuyến và cân bằng tải mà không cần các trạng thái được kiểm soát bên trong giữa các yêu cầu.

Một yêu cầu hoàn chỉnh, độc lập không đòi hỏi máy chủ để thu thập được bất kỳ ngữ cảnh hoặc trạng thái của ứng dụng nào trong lúc xử lý yêu cầu. Một ứng dụng (hoặc máy khách) Web service RESTchứa ở phần đầu và phần thân trang HTTP của một yêu cầu tất cả các tham số, ngữ cảnh và dữ liệu cần thiết bởi thành phần bên ngoài máy chủ để đưa ra một phản hồi. Phi trạng thái theo nghĩa này nâng cao tính hiệu quả của dịch vụ Web, đơn giản hoá thiết kế và sự thi hành của các thành phần của máy chủ vì khi máy chủ không có trạng thái sẽ huỷ bỏ nhu cầu để đồng bộ hoá các mảng dữ liệu với một ứng dụng bên ngoài.

Hình 1 minh hoạ một dịch vụ trạng thái, từ đó một ứng dụng có thể yêu cầu trang sau trong một tập hợp các trang kết quả, giả sử rằng dịch vụ theo sát ứng dụng dừng lại ở nơi trong khi điều chỉnh tập hợp đó. Đối với thiết kế trạng thái, dịch vụ gia tăng và lưu giữ một previousPage (trang trước) thay đổi ở nơi để có thể phản hồi các lệnh tiếp theo.

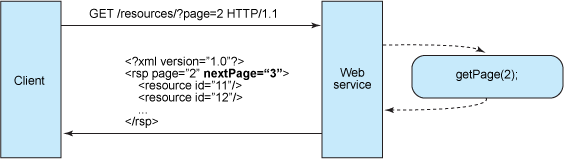
##### Hình 1. Thiết kế trạng thái



Dịch vụ trạng thái như thế này trở nên phức tạp. Trong môi trường Nền tảng Java, Phiên bản Doanh nghiệp (EE), dịch vụ trạng thái yêu cầu rất cẩn thận lúc ban đầu để lưu trữ hiệu quả và cho phép đồng bộ hoá dữ liệu session (phiên làm việc) qua một hệ thống container Java EE. Trong môi trường này, có một vấn đề quen thuộc đối với các chuyên viên phát triển servlet/JavaServer Pages (JSP) và Enterprise JavaBeans (EJB), những người này thường gặp khó khăn khi tìm gốc rễ nguyên nhân của java.io.NotSerializableException trong khi tái tạo session. Liệu nó được chuyển bởi thành phần chứa Servlet trong khi HttpSession được tái tạo hoặc chuyển đi bởi thành phần chứa EJB trong khi sao bản EJB trạng thái, đó là vấn đề mà có thể làm các chuyên viên phát triển mất nhiều ngày để xác định mấu chốt một đối tượng mà không thực thiSerializable, đôi khi trong một đồ thị phức tạp của các đối tượng mà đóng góp nên trạng thái của máy chủ. Ngoài ra, phần tối ưu hoá làm đội thêm chi phí ảnh hưởng đến hiệu quả của máy chủ.

Mặt khác, các thành phần máy chủ phi trạng thái ít phức tạp hơn để thiết kế, viết và phân bổ thông qua máy chủ được cân bằng tải. Dịch vụ phi trạng thái không chỉ hoạt động tốt hơn, nó còn chuyển hầu hết vai trò duy trì trạng thái sang ứng dụng ở máy khách. Trong một dịch vụ mạng RESTful, máy chủ chịu trách nhiệm đưa ra các phản hồi và cung cấp một giao diện cho phép máy khách duy trì trạng thái ứng dụng của chính nó. Ví dụ, trong yêu cầu tập hợp trang kết quả, máy khách sẽ gồm số trang thực tế khi truy xuất thay vì đơn giản chỉ là yêu cầu tiếp theo (xem hình 2).

##### Hình 2. Thiết kế phi trạng thái



Một dịch Web phi trạng thái sinh ra một phản hồi liên kết với số trang tiếp theo trong một tổng thể và để máy khách làm những gì mà nó cần để giữ giá trị này ở mức nhất định. Khía cạnh này của thiết kế dịch vụ Web RESTful có thể được tách thành hai phần trách nhiệm như là mức phân chia cao nhất mà chỉ rõ một dịch vụ phi trạng thái có thể được duy trì như thế nào.

**Máy chủ**

* Tạo ra các phản hồi bao gồm các đường dẫn tới nguồn tài nguyên cho phép các ứng dụng điều hướng giữa các tài nguyên liên quan. Loại phản hồi này nhúng các liên kết. Tương tự, nếu các yêu cầu đối với máy chủ hoặc các kho tài nguyên, thì các phản hồi RESTful Web service điển hình có thể bao gồm các đường dẫn đến các máy con hoặc các tài nguyên phụ sao cho những phản hồi này được duy trì kết nối.
* Tạo ra các phản hồi mà xác định chúng có thể lưu trữ hoặc không phải để nâng cao được hiệu quả bằng cách giảm số lượng yêu cầu đối với các tài nguyên trùng nhau và bằng cách loại trừ một vài yêu cầu toàn bộ. Máy chủ làm được như vậy bằng cách gộp một phản hồi phần đầu HTTP Last - Modified (lần sửa gần nhất) (giá trị ngày) và Cache-Control (bộ điều khiển lưu trữ).

**Ứng dụng máy khách**

* Sử dụng phần đầu phản hồi Cache-Control (bộ điều khiển lưu trữ tạm) để xác định lưu trữ tài nguyên (lập một vùng sao chép nội bộ) hay không. Máy khách cũng đọc phần đầu phản hồi Last-Modified (lần sửa gần nhất) và gửi lại giá trị ngày vào phần đầu If-Modified-Since (nếu-sửa) để truy vấn máy chủ xem tài nguyên có thay đổi không. Việc này được gọi là truy vấn có điều kiện, và hai phần đầu đi với nhau trong phản hồi của máy chủ là mã 304 chuẩn (không sửa đổi) và bỏ qua tài nguyên thực được yêu cầu nếu nó không thay đổi. Mã phản hồi HTTP 304 có nghĩa rằng máy khách có thể sử dụng an toàn một vùng sao lưu nội bộ, lưu giữ một bản sao mới nhất của tài nguyên đại diện, hiệu quả bằng cách vượt qua yêu cầu GET tiếp theo cho đến khi tài nguyên thay đổi.
* Gửi các yêu cầu hoàn chỉnh có thể được đáp ứng độc lập bởi các yêu cầu khác. Điều này đòi hỏi máy khách sử dụng toàn bộ các phần đầu HTTP như chỉ định bởi giao diện dịch vụ mạng và để gửi các đại diện tài nguyên hoàn chỉnh trong phần giữa của yêu cầu. Máy khách gửi yêu cầu lập một vài giả thuyết về các yêu cầu trước đó, sự tồn tại của một vùng của máy chủ, khả năng của máy chủ để thêm các ngữ cảnh vào yêu cầu, hoặc về các trạng thái ứng dụng mà được giữ giữa các yêu cầu.

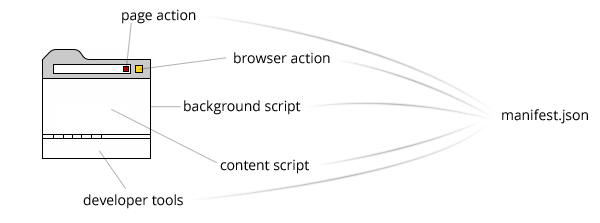
Sự hợp tác này giữa ứng dụng máy khách và máy chủ là cần thiết để có một phi trạng thái trong một Web service RESful. Nó nâng cao hiệu quả bằng cách tiết kiệm băng thông và tối thiểu hoá trạng thái ứng dụng về phía máy chủ.

Chính vì vậy , Đưa một tài nguyên hệ thống thông qua một RESTful API là một cách linh động để cung cấp các loại ứng dụng khác nhau với dữ liệu đã được định dạng theo cách tiêu chuẩn. Nó giúp đáp ứng các yêu cầu tích hợp, điều rất quan trọng để xây dựng hệ thống khi dữ liệu được kết hợp dễ dàng (mashups) và để mở rộng hoặc xây dựng trên một gói hệ thống căn bản

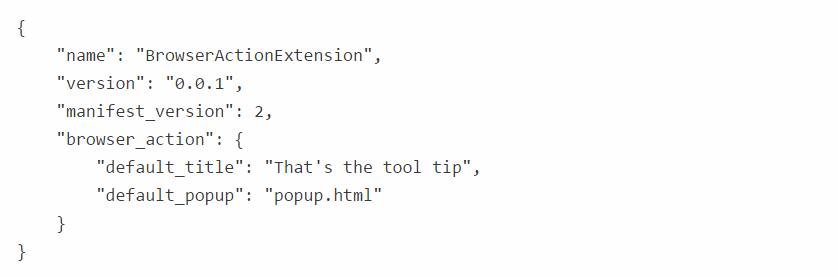
**Extension là gì?**

* **Extension** là một chương trình phần mềm nhỏ có thể chỉnh sửa và kế thừa được các tính năng của trình duyệt Chrome.
* **Extension** thường không có hoặc có giao diện rất nhỏ ví dụ như một icon ảnh nhỏ ở phía trên bên phải của thanh công cụ của trình duyệt Chrome.
* **Extension** thì bao gồm tất cả các file của nó vào thành một file duy nhất để người dùng có thể tải và cài đặt. Từ bao gồm ở đây có nghĩa là Extension không giống với các ứng dụng web , Extension không cần phải phụ thuộc vào nội dung từ web.

**Cấu trúc của Extension?**

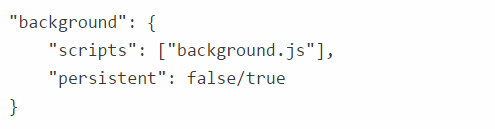
****

1. **Manifest**
   * Đầu vào của một Extension chính là file manifest.json. Nó chứa các đối tượng JSON hợp lệ khai báo các thành phần của extension đó:

****

* Bao gồm một số các thuộc tính cơ bản và cần thiết như:
  + Version : Đại diện cho phiên bản của Extension , có thể là một số bất kỳ từ 1 -> 4.
  + Name : Tên của Extension.
  + Manifest\_version thường và nên được lấy với giá trị là 2.
* Manifest file có thể chứa một số thuộc tính khác dựa vào kiểu Extension mà bạn sẽ xây dựng.

1. **Background page**
   * Tất cả các Extension đều có một background page ẩn được chạy bởi trình duyệt.
   * Có 2 loại background page đó là
     1. Persistent background pages : Đây là loại background page luôn hoạt động trong tất cả thời gian sử dụng trình duyệt.
     2. Event pages : Chỉ hoạt động khi trình duyệt cần sử dụng tới.
   * Background pages đóng vai trò là cầu nối giữa việc chạy thực nghiệm với tư duy chính của chúng ta đặt vào Extension.

****

1. **Content Script**
   * Nếu bạn muốn truy nhập và sử dụng được cây DOM của page hiện tại thì đây chính là file mà bạn nên sử dụng.

****

1. **User Interface**
   * **Browser Action :** Hầu hết các Extension đều có sử dụng thuộc tính browser\_action. Đây chính là icon đại diện cho Extension được đặt trên góc trên cùng bên phải của thanh địa chỉ.
   * **Page Action :** Cũng giống như browser action, tuy nhiên page action là một icon được đặt trong thanh địa chỉ.