SOAP против REST API: Ключевые различия, объясненные для новичков

Описание

Выбор между REST и SOAP – сложный шаг в процессе создания API, если вы новичок. SOAP и REST – это разные подходы к передаче данных для API. Вы хотите подключить свое приложение к Instagram для перепоста изображений с использованием определенных хэштегов или к Facebook, чтобы оно могло автоматически создавать посты? А может быть, вы хотите встроить видео с YouTube на свой сайт? Все это и многое другое можно сделать с помощью интерфейсов прикладного программирования (API).

API, такие как Instagram API, Facebook API и YouTube API, обеспечивают безопасный и стандартизированный способ “общения” различных программ друг с другом. То есть приложение может извлекать функциональность или данные из другой части программного обеспечения и использовать их для улучшения собственной функциональности или UX. Но как приложения делают эти запросы, обрабатывают их и эффективно отвечают на них? Это зависит от того, как вы создадите API. Давайте рассмотрим два наиболее распространенных метода – SOAP и REST. Затем мы подробно сравним различия.

Что такое SOAP?

SOAP, что расшифровывается как Simple Object Access Protocol, – это очень строгий и безопасный способ создания API, который кодирует данные в XML.

Что такое REST?

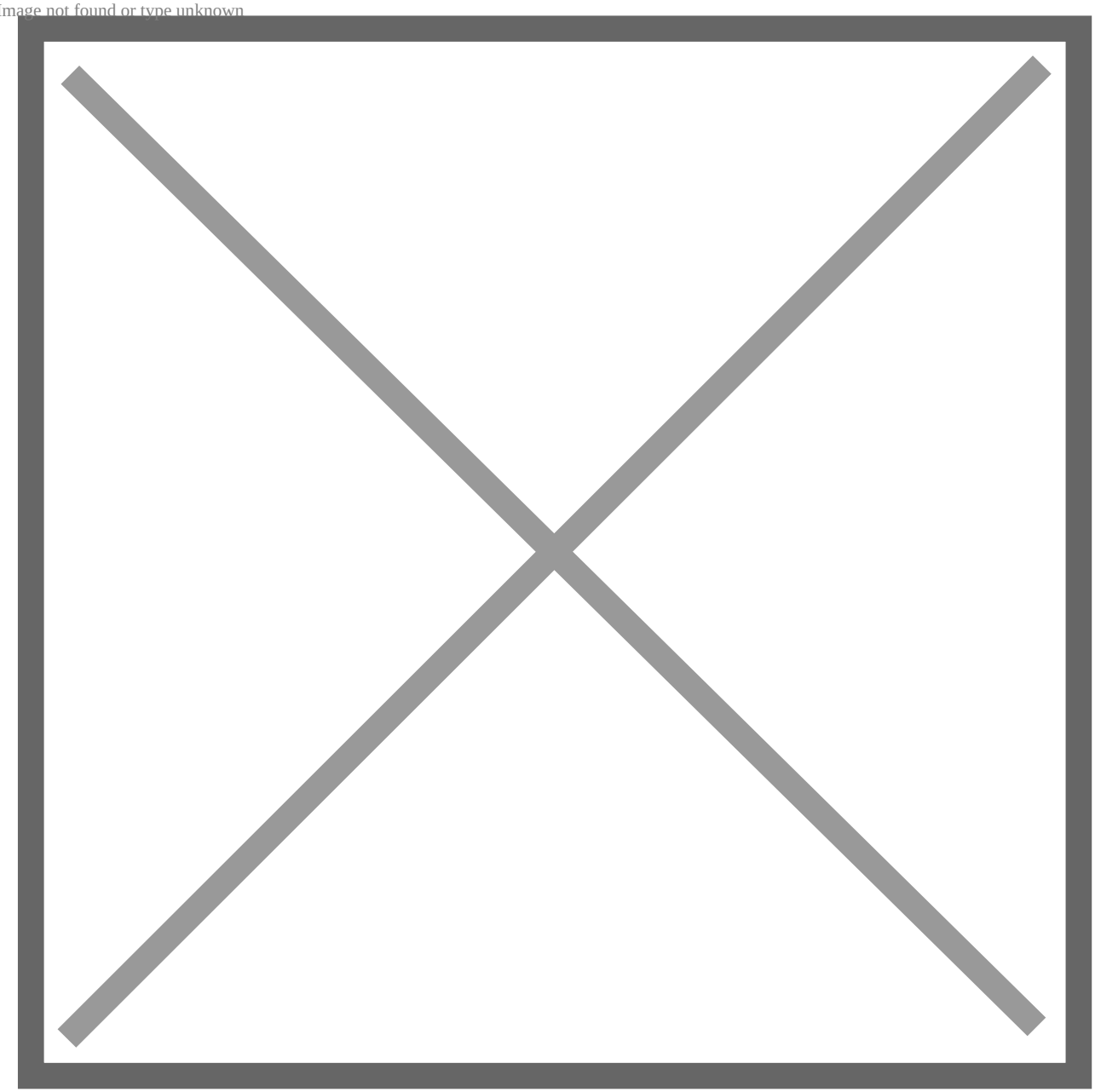
REST, что расшифровывается как Representational State Transfer, является более простым и гибким методом создания API. API REST могут передавать данные в различных форматах, включая XML, а также обычный текст, HTML и JSON.

SOAP против REST

В вопросе SOAP vs. REST сравниваются два подхода к передаче данных для API. Основное различие заключается в том, что SOAP – это структурированный протокол, а REST – более гибкий и менее определенный. REST и SOAP – это два разных способа соединения приложений с данными на стороне сервера. Оба формата API используют данные, которые могут читать как люди, так и машины, и оба часто используют протоколы HTTP. Однако между ними существует множество различий.

На графике ниже приведены некоторые различия между REST и SOAP.

Image not found or type unknown



Теперь, когда у нас есть общий обзор различий между SOAP и REST, давайте более подробно рассмотрим их сравнение с точки зрения сервисов, безопасности и примеров.

- Службы SOAP и REST
- Безопасность SOAP и REST
- Примеры SOAP и REST
    - Пример REST

SOAP против REST-сервисов

Чтобы сравнить SOAP и REST API, мы должны сначала понять, что именно делает каждый тип API.

Сервисы SOAP

SOAP – это стандартная система коммуникационных протоколов, использующая технологии XML для определения обширной структуры обмена сообщениями, которая позволяет обмениваться структурированной информацией в децентрализованной, распределенной среде. Другими словами, SOAP позволяет приложениям, работающим на разных операционных системах, взаимодействовать, используя различные технологии и языки программирования. Клиент может использовать API SOAP для создания, получения, обновления или удаления записей, таких как пароли, учетные записи, лиды и пользовательские объекты, с сервера.

Сервисы REST

REST, с другой стороны, является архитектурным стилем, а не протоколом. Как было сказано выше, он расшифровывается как Representational State Transfer. Это означает, что когда клиент запрашивает ресурс с помощью REST API, сервер передает обратно текущее состояние ресурса в стандартизированном представлении. Другими словами, REST API получают запросы на ресурс и возвращают всю необходимую информацию о ресурсе в формате, который клиенты могут легко интерпретировать. Помимо запроса ресурсов, клиенты могут использовать REST API для изменения и даже добавления новых элементов на сервере с помощью методов HTTP. Для более подробного описания читайте статью REST APIs: Как они работают и что вам нужно знать.

Безопасность SOAP и REST

Теперь, когда мы лучше понимаем, что делают SOAP и REST API, давайте сравним их меры безопасности и протоколы.

Безопасность SOAP

Поскольку SOAP является протоколом обмена сообщениями, защита SOAP API в

первую очередь направлена на предотвращение несанкционированного доступа к сообщениям (и информации пользователей, содержащейся в этих сообщениях), получаемым и отправляемым из SOAP API. Основным средством защиты от несанкционированного доступа являются Web Standards (WS) Security, набор принципов, регулирующих процедуры конфиденциальности и аутентификации для обмена сообщениями SOAP. Меры, соответствующие WS Security, включают пароли, шифрование XML и маркеры безопасности, среди прочих механизмов. WS Security выходит за рамки традиционных механизмов веб-безопасности, таких как HTTPS, которые обеспечивают безопасность сообщений только во время транспортировки между клиентом, сделавшим запрос, и сервером или веб-службой, имеющей запрашиваемые данные. WS Security, с другой стороны, обеспечивает безопасность сообщений за пределами соединения HTTPS, а иногда даже за пределами транспортного уровня.

Как именно она это делает? SOAP-сообщение либо содержит информацию, необходимую для его защиты, либо содержит информацию о том, где можно получить информацию, необходимую для защиты. Сообщение SOAP также содержит в своем заголовке информацию, относящуюся к протоколам и процедурам обработки указанного уровня безопасности сообщения. Это означает, что когда конечная точка веб-службы получает сообщение SOAP, она проверяет информацию о безопасности в заголовке, чтобы убедиться, что она не была подделана. Вот почему о SOAP говорят, что он имеет “безопасность на уровне сообщений”.

SOAP поддерживает такие спецификации WS, как:

- WS-Addressing
- WS-Policy
- WS-Security
- WS-Federation
- WS-ReliableMessaging
- WS-Coordination
- WS-AtomicTransaction
- WS-RemotePortlets

Поэтому SOAP API идеально подходят для внутренней передачи данных и других конфиденциальных задач – но не подходят для использования публичной веб-службы, которая доступна всем, например, для получения данных о погоде.

Подробнее о том, когда следует использовать REST и SOAP API, мы поговорим далее в этом посте.

Безопасность REST

API REST поддерживают только традиционные механизмы веб-безопасности, такие как HTTPS. Это означает, что когда приложение отправляет и получает сообщение от REST API, используя HTTPS, сообщение защищено только для HTTPS-соединения. То есть, сообщение защищено только во время передачи между клиентом и службой. Это хорошо для публичных веб-служб, но может быть недостаточно для передачи более чувствительных данных.

Поскольку REST API не имеют встроенных возможностей безопасности или расширений, которые есть у SOAP, их безопасность зависит от дизайна самих API. REST API могут быть разработаны с определенными механизмами безопасности, которые гарантируют, что только аутентифицированные и авторизованные пользователи могут получить к ним доступ. Общими методами аутентификации REST API являются базовая аутентификация HTTP, веб-маркеры JSON, OAuth и ключи API. REST API также должны иметь подробные спецификации и отклонять любые запросы, которые, например, не содержат правильных деклараций в заголовках HTTP или не соответствуют их спецификациям. Это поможет защитить базовое веб-приложение от неправильного и вредоносного ввода данных даже после того, как клиент получил доступ.

Примеры SOAP и REST

Давайте сравним некоторые примеры запросов и ответов на запросы и ответы от SOAP API и REST API. Приведенные ниже примеры основаны на QAComplete SOAP API и REST API соответственно. QAComplete – это комплексный инструмент управления тестированием программного обеспечения от SmartBear.

Пример SOAP

Запросы QAComplete SOAP – это запросы HTTP POST, выполняемые к URL конечной точки веб-сервиса. Клиент и сервер обмениваются данными в формате XML в теле HTTP-запросов и ответов. Этот SOAP API принимает только HTTP POST запросы, но он также поддерживает несколько общих операций для всех типов элементов, включая Add, Delete, Load, LoadByCriteria и Update. Вы увидите эти операции вместо

HTTP-глаголов GET, PUT, PATCH и DELETE.

Приведенный ниже пример кода запрашивает получение дефекта по его ID в проекте в QAComplete, используя SOAP API. Фрагмент кода написан на языке XML и использует операцию Bugs_Load:

```
POST /psws/psws.asmx HTTP/1.1
Host: myteam.mysite.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 486 {Insert an appropriate value here}

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:x
<soap12:Body>
<Bugs_Load xmlns="http://www.pragmaticsw.com/">
<AuthenticationData>
<AppCode>agSP</AppCode>
<DeptId>7154</DeptId>
<ProjId>1032</ProjId>
<UserId>25315</UserId>
<PassCode>p@ssword</PassCode>
</AuthenticationData>
<BugId>5</BugId>
</Bugs_Load>
</soap12:Body>
</soap12:Envelope>
```

Ответ будет содержать код состояния HTTP и объект Bug, содержащий информацию о необходимом дефекте. Она будет записана в формате XML. В приведенном ниже примере есть код состояния HTTP 200, показывающий, что запрос был успешным, а также информация о дефекте в XML:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 2244 {The server returns an appropriate value here}

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:x
<soap12:Body>
<Bugs_LoadResponse xmlns="http://www.pragmaticsw.com/">
<Bugs_LoadResult>
<CustomFields>
<string>0.9.19</string>
<string>1.0.24</string>
</CustomFields>
<CustomFieldNames>
<string>Found in build</string>
<string>Fixed in build</string>
</CustomFieldNames>
<BugId>13254</BugId>
```

```
<Title>Floating toolbar improvements</Title>
<StatusCode>Resolved</StatusCode>
<PriorityCode>2-Fix Soon</PriorityCode>
<HowFoundCode>Test-Ad hoc</HowFoundCode>
<ResolutionCode>Fixed</ResolutionCode>
<AssigneeUserId>27942</AssigneeUserId>
<OpenedBy>27568</OpenedBy>
<ClosedBy>27942</ClosedBy>
<ResolvedBy>27568</ResolvedBy>
<Description><![CDATA[The design of the floating toolbar needs improvement so
<Resolution>Resolved by Susan McLaren on 24-Jun-2014 at 03:55 PM</Resolution>
<OwnerUserId>27572</OwnerUserId>
<TestCaseId>0</TestCaseId>
<FolderId>52359</FolderId>
<EstHrs>0.000</EstHrs>
<EstStart>2014-06-10T10:00:00</EstStart>
<EstFinish>2014-06-13T10:00:00</EstFinish>
<PctComplete>100</PctComplete>
<ActHrs>0.000</ActHrs>
<ActualStart>2014-06-17T00:00:00</ActualStart>
<ActFinish>2014-06-31T00:00:00</ActFinish>
<EstHrsRemaining>0.000</EstHrsRemaining>
<DateOpened>2014-05-26T15:54:21.093</DateOpened>
<DateResolved>2014-06-24T15:55:25</DateResolved>
<DateClosed>0001-01-01T00:00:00</DateClosed>
<DateUpdated>2014-06-31T05:54:22</DateUpdated>
<ProjId>17823</ProjId>
<DateCreated>2014-05-26T15:54:21.093</DateCreated>
<UpdateUserId>27534</UpdateUserId>
<OriginalId>0</OriginalId>
<ImportId>0</ImportId>
<AssignedToName>Doe, John</AssignedToName>
<OpenedByName>McLaren, Susan</OpenedByName>
<OpenedByEmail>susan@example.com</OpenedByEmail>
<OpenedByCompany>Edgar Solutions</OpenedByCompany>
<ResolvedByName>McLaren, Susan</ResolvedByName>
<UserName>Fry, Alex</UserName>
<OwnerName>Davis, Eugeny</OwnerName>
<NbrFiles>0</NbrFiles>
<NbrNotes>1</NbrNotes>
<NbrEvents>0</NbrEvents>
<NbrTasks>2</NbrTasks>
<FolderName>FamilyAlbum/Release 1.1/Iteration 1</FolderName>
</Bugs_LoadResult>
</Bugs_LoadResponse>
</soap12:Body>
</soap12:Envelope>
```

Обратите внимание, что это SOAP-сообщение содержит конверт, заголовок и элемент body, все из которых являются обязательными.

Пример REST

REST-запросы – это HTTP-запросы, выполняемые к URL конечной точки QAComplete REST API, который имеет следующий формат: http(s)://{ваш сервер}/rest-api/service/api/{version}/{resource}. Этот API использует HTTP-методы GET, POST, PUT, PATCH и DELETE.

Пример кода ниже запрашивает получение дефекта по его ID в проекте в QAComplete, используя QAComplete REST API. Фрагмент кода написан в JSON и использует метод HTTP, GET:

```
GET http://yourserver.com/rest-api/service/api/v1/projects/11873/defects/17 HT
Host: yourserver.com
Connection: keep-alive
Accept: application/json
Authorization: Basic am9obkBleGFtcGxlLmNvbTpwQHNzd29yZA==
```

Сервер вернет код состояния HTTP и информацию о дефекте. В приведенном ниже примере есть код состояния HTTP 200, показывающий, что запрос был успешным, а также объект JSON с информацией о дефекте:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 1758

{
"id": 17,
"title": "Floating toolbar improvements",
"status": "Resolved",
"__permissions": {
"acl": 7
},
"act_finish": "2015-06-31T00:00:00.0000000",
"act_hrs": 0,
"act_start": "2015-06-17T00:00:00.000000",
"actual_results": "The design of the floating toolbar is inconsistent.",
"assigned_to_name": "Smith, John",
"assignee_user_id": 27942,
"closed_by": 27942,
"custom_fields": [
{
"Id": "Custom1",
"Name": "Components",
"Value": "UI"
},
{
"Id": "Custom2",
```

```
"Name": "Fixed in build",
"Value": "1.0.13"
}
],
"date_closed": "0001-01-01T00:00:00.0000000",
"date_created": "2015-05-26T10:54:21.0930000",
"date_opened": "2015-05-27T11:42:29.1700000",
"date_resolved": "2015-06-24T09:55:25.4500000",
"date_updated": "2015-06-24T09:55:25.4500000",
"description": "The design of the floating toolbar needs improvement",
"est_finish": "0001-01-01T00:00:00.0000000",
"est_hrs": 0,
"est_hrs_remaining": 0,
"est_start": "0001-01-01T00:00:00.0000000",
"expected_results": "This should work",
"folder_id": 52359,
"folder_name": "UI Defects",
"functional_area_code": "",
"how_found_code": "Code Review",
"import_id": 0,
"issue_code": "Code Defect",
"nbr_events": 0,
"nbr_files": 1,
"nbr_notes": 1,
"nbr_tasks": 0,
"opened_by": 27942,
"opened_by_company": "EDGB",
"opened_by_email": "smith.john@edgb.com",
"opened_by_name": "Smith, John",
"original_id": 0,
"owner_name": "Smith, John",
"owner_user_id": 27942,
"pct_complete": 0,
"priority_code": "1-Fix ASAP",
"project_id": 11873,
"resolution": "Toolbar is redesigned",
"resolution_code": "Fixed",
"resolved_by": 27942,
"resolved_by_name": "Smith, John",
"severity_code": "1-Crash",
"software_version_code": "1.0",
"steps_to_repro": "step 1rnstep 2rnstep 3rnstep 3.1rnstep 3.2rn",
"update_user_id": 27942,
"user_name": "Smith, John"

}
```


Обратите внимание, что этот ответ не такой длинный, как ответ, полученный от SOAP API. Поскольку REST API обычно отправляют ответы с меньшим количеством данных и в формате JSON, они требуют меньшей пропускной способности, чем SOAP API.

Когда использовать SOAP, а когда REST

Общее правило при выборе между SOAP и REST для создания API: если вам нужна стандартизация и повышенная безопасность, используйте SOAP. Если вам нужна гибкость и эффективность, используйте REST. О конкретных случаях использования SOAP и REST читайте далее.

Когда следует использовать SOAP

Разработка частных API, особенно для крупных предприятий.

SOAP позволяет передавать данные в децентрализованной, распределенной среде. Он также имеет множество механизмов веб-безопасности. Эти качества делают его идеальным для корпоративных решений.

Работа с операциями с состоянием.

В отличие от вызовов REST API, вызовы SOAP API являются государственными. Это означает, что сервер хранит информацию о клиенте и использует ее в серии запросов или цепочке операций. Хотя это требует больше ресурсов сервера и пропускной способности, это важно при выполнении повторяющихся или цепных задач, например, банковских переводов.

Использование базового транспортного протокола, отличного от HTTP.

SOAP не зависит от базового транспортного протокола, поэтому вам не обязательно использовать HTTP. Вместо этого вы можете использовать SMTP (Simple Mail Transfer Protocol), JMS (Java Messaging Service) или другой транспортный протокол, в зависимости от вашего приложения.

Когда следует использовать REST

Разработка общедоступных API.

Многие считают, что REST API проще в использовании и принятии, чем SOAP API, что делает их идеальными для создания публичных веб-служб. В REST также отсутствуют некоторые встроенные функции безопасности, которые есть в SOAP –

но они и не нужны при работе с публичными данными и сервисами.

Работа с ограниченными ресурсами сервера и пропускной способностью.

Все вызовы REST API должны быть без статических данных. Это означает, что каждое взаимодействие является независимым, поэтому каждый запрос и ответ содержат всю информацию, необходимую для завершения взаимодействия. Поскольку сервер воспринимает каждый запрос как новый, он не хранит информацию о прошлых запросах. Это значительно сокращает объем необходимой памяти сервера.

Это повышает производительность, так как серверу не нужно предпринимать дополнительные действия или получать прошлые данные при выполнении запроса. Поскольку REST не имеет статического характера, данные можно кэшировать, что также экономит ресурсы сервера и пропускную способность. Наконец, API REST могут использовать различные форматы данных, например, JSON, который легче XML. Это делает их более быстрыми и эффективными, чем большинство SOAP API.

Создание мобильных приложений.

Поскольку REST отличается легкостью, эффективностью, отсутствием статичности и возможностью кэширования, он идеально подходит для создания мобильных приложений.

Выбор между REST и SOAP

Не существует золотого правила для создания API. Выбор между SOAP и REST при создании API зависит от множества факторов, в том числе от того, какой язык программирования вы используете и сколько времени у вас есть на его создание.

Дата Создания
26.05.2023