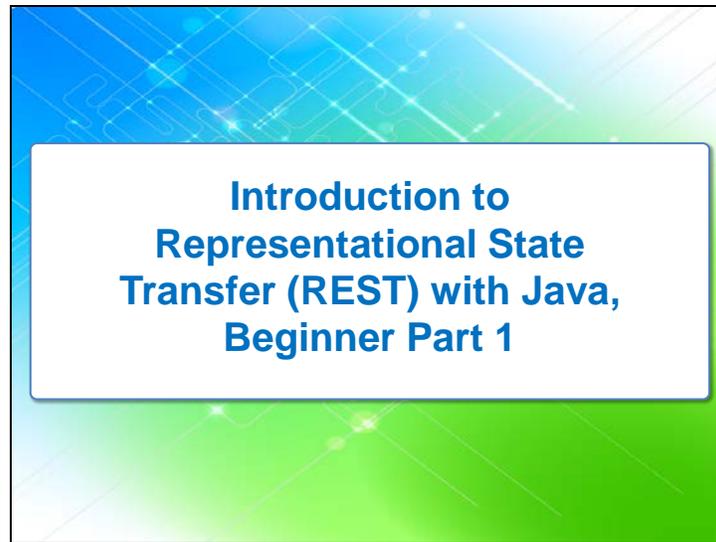


Slide 1

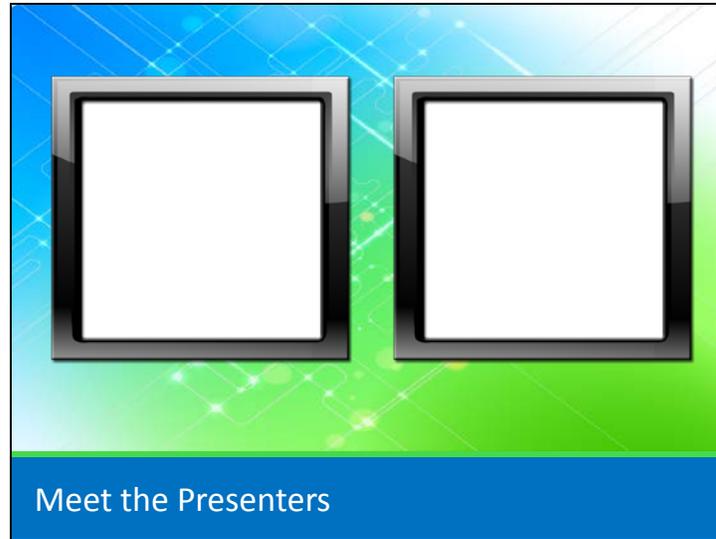


PRESENTER 1 SAY:

On behalf of IT Workforce Development, welcome to today's session on *Introduction to Representational State Transfer (REST) with Java, Beginner Part 1*. This is the first of a three-part series on REST.

Today's course will give you a broad overview of REST concepts. Each course will build on the next and will help you learn how to design, build, and test RESTful web services.

Slide 2



PRESENTER 1 SAY:

We'd like to take a moment to introduce ourselves. I'm PRESENTER 1.

PRESENTER 1 SAY:

And I'm PRESENTER 2.

We're looking forward to providing you with information on today's topic. But before we get started, let's talk about a few Lync items.

Slide 3

Chat/Questions: Use the Chat window
Dial-in: Use the VANTS line in the invitation
Participation Credit: Submit TMS self-certification
Group Participation: Email vaitwd@va.gov
Handouts: Download via the paperclip icon



Lync Information

PRESENTER 1 SAY:

We want this to be an engaging session. We'll be asking you questions throughout this session, so please use the Chat window to respond. You can also use the Chat window if you have questions for us. We'll answer your questions either during the session or during Q&A at the end of the session.

We want to make sure you get credit for attending today's session. We'll provide you with instructions on how to complete the self-certification at the end, so be sure to stick around. If you are attending as a group, please email the VA ITWD mailbox and let us know. We'll be able to ensure you all receive completion credit.

PRESENTER 2 SAY:

To access today's handouts, select the paperclip icon in the Lync toolbar above the list of participants. Then, select the right arrow next to the file in the Attachments pop-up menu. From there, you can select Open or Save As. If you chose Save As, you can select where you want to save the downloads. You can also access the downloads from your keyboard, using control-F.

Slide 4

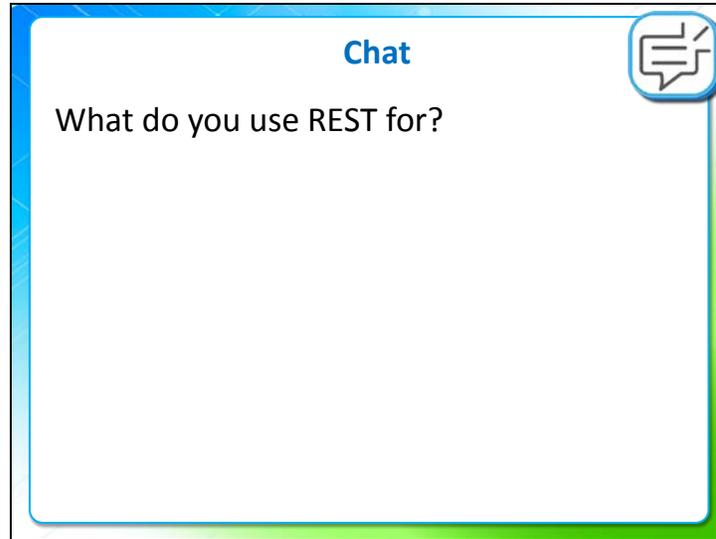


PRESENTER 2 SAY:

In this first course, we will provide a broad overview of REST. We'll begin by discussing some key concepts that you'll need to be familiar with for the entire series. Next, we'll talk about the basic components of REST, including data elements and connectors. After that, we'll talk about programming RESTful web services as well as programming in Java. Toward the end of the session, we'll walk you through a live demo of how to connect to a REST API. We'll also provide you with the opportunity to practice what you learn today through a self-paced homework exercise that you'll complete after this class is over. We'll get into the details of that at the end of today's training.

I want to emphasize that today's course will serve as an introduction to topics that will be discussed in greater depth in courses two and three. It's important to take all three of these courses in sequence, as the information builds on itself from one course to the next.

Slide 5



PRESENTER 1 SAY:

Before we get started I want to ask a question. What do you use REST for? We know that you may know nothing about REST right now, and that's just fine. You're in the right place. We're going to start from the beginning in this course, and we have two other courses for you after this one that will get into more advanced concepts. But if you are working on anything REST related, we'd like to know about it and how you're using it.

PRESENTER 2 SAY:

This session will give you a nice overview of REST and RESTful web services and will help prepare you for subsequent courses.

Thank you all for your participation.

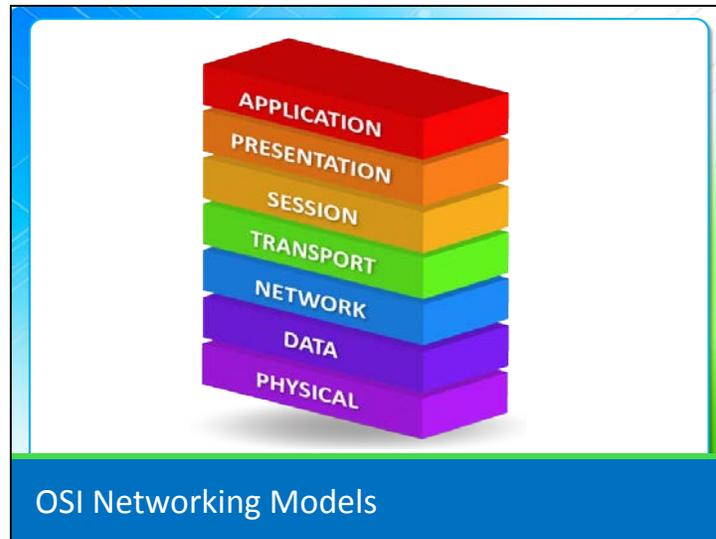
Slide 6



PRESENTER 1 SAY:

Before we talk about the basics of REST, let's discuss some key web architecture concepts that you'll need to be familiar with as we move through the series. Some of these concepts you may already be familiar with, and that's OK. If you are familiar, then this can be a good refresher for you.

Slide 7



PRESENTER 1 SAY:

Computers communicate over networks. In the early 1980s, when computer communication was in its infancy, there was no model for how hardware and software was developed and connected. As the need for hierarchical communications grew with the expanding number of devices on networks, the International Standards Organization developed the Open Systems Interconnection Model, called the O-S-I Model.

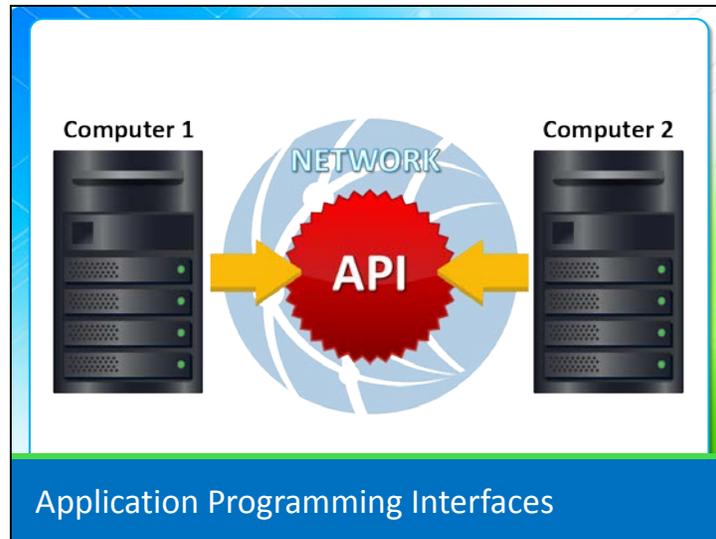
The OSI model is a conceptual model that standardizes and characterizes computer communication by separating it into abstraction layers. It partitions networks into seven layers, which, from bottom to top, include Physical, Data link, Network, Transport, Session, Presentation, and Application.

Each layer contains a group of similar functions that provide services to the layer above it and receives service from the layer below it.

Think about it as if you're sending a letter—it's a process. It requires the actual writing, putting the letter into an envelope, addressing the envelope, and mailing. From there, the letter is handled in even more steps by the postal service. Eventually, the letter reaches its destination. Computer networks function in a very similar way. Each step, or layer, serves a specific function in achieving the overall goal.

So where does REST fit in this model? REST happens over HTTP, which is an application in the OSI model. This allows the REST developer to focus on the application, while relying on the OSI model to take care of the lower data transport. This can lead to efficient and effective coding.

Slide 8



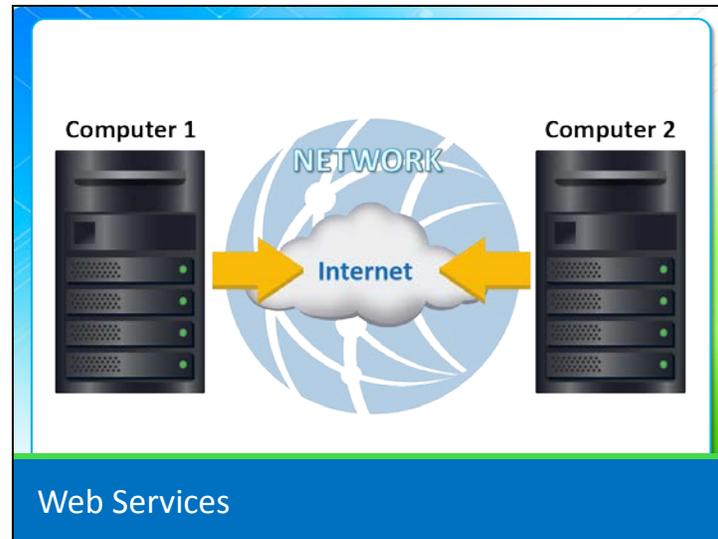
PRESENTER 2 SAY:

The basic REST application is a client and server connecting and sharing information. The interface used between them is what we call an Application Programming Interface, or APIs. In computer programming, an API specifies how software components interact with each other. For example, an API can come in a library of routines, data structures, object classes or, as in the case of REST, remote calls that communicate with other computers or networks to execute a procedure—or any combination. It is basically a structured, advertised, or agreed way of communicating.

An API specifies what should be accomplished or how to interact with specific components. It's exactly what it sounds like—how programming applications interface with each other.

As we move through these courses, we'll discuss APIs in greater detail.

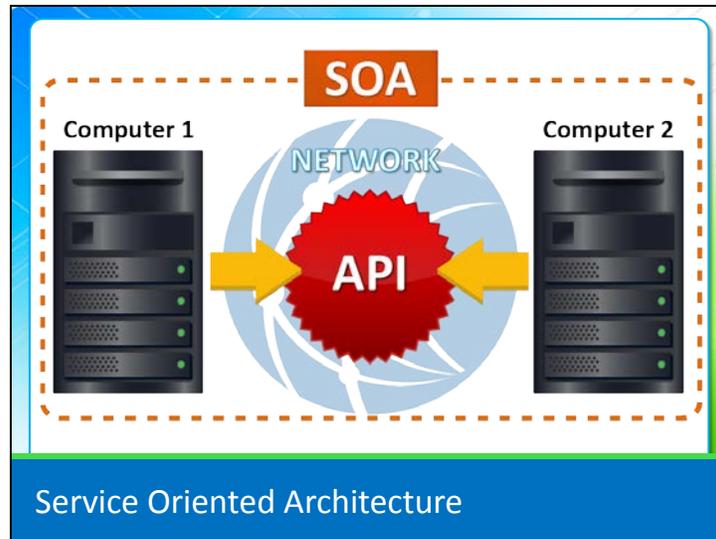
Slide 9



PRESENTER 1 SAY:

A web service is a type of API. A web service allows communication between software systems to exchange data over the **Internet**. Of course, software is built using many programming languages, so there is a need for a data exchange method that doesn't depend upon a specific language. Web services are communicated over HTTP, or hypertext transfer protocol. Additionally, many applications understand XML, or Extensible Markup Language. For example, web services can use XML for data exchange.

Slide 10

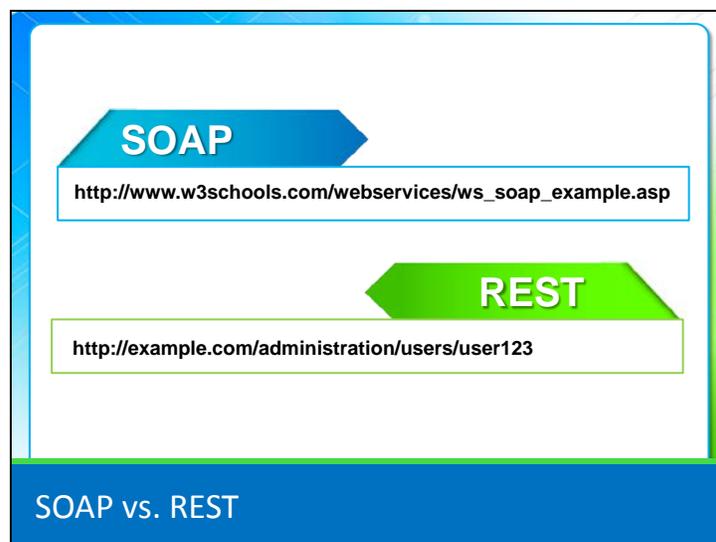


PRESENTER 2 SAY:

Service-oriented architecture, or SOA , is an architecture based on the principle that applications provide services to other applications, usually promoting loose coupling (for example, communicating over REST).

An everyday illustration of this occurs when you use Facebook or a website. Both of them communicate with the Facebook servers using REST APIs. In fact, the servers in Facebook also communicate with each other through REST APIs. For example, when you upload a picture to Facebook, one service looks for faces to tag, another resizes the picture, and another generates the URL for the picture, placing it into a database, and yet another service posts the picture to your wall. All of these services grouped together constitute a service-oriented architecture, where one application is providing some service to another.

Slide 11



PRESENTER 1 SAY:

Some of you may have heard of SOAP. SOAP is an acronym for Simple Object Access Protocol. In the past, many programmers used SOAP as the protocol when developing web services. It consists of the envelope, which defines what the message is and how to process it, encoding rules, and procedure call and response conventions, all of which could be advertised via WSDL, or Web Services Description Language. However, in recent years, there has been a push toward using REST as the framework instead. Why is that? Overall, REST is considered easier to use compared to SOAP, which can be very complicated. Also, REST focuses on the data transfer, which allows for more of a black-box approach to programming.

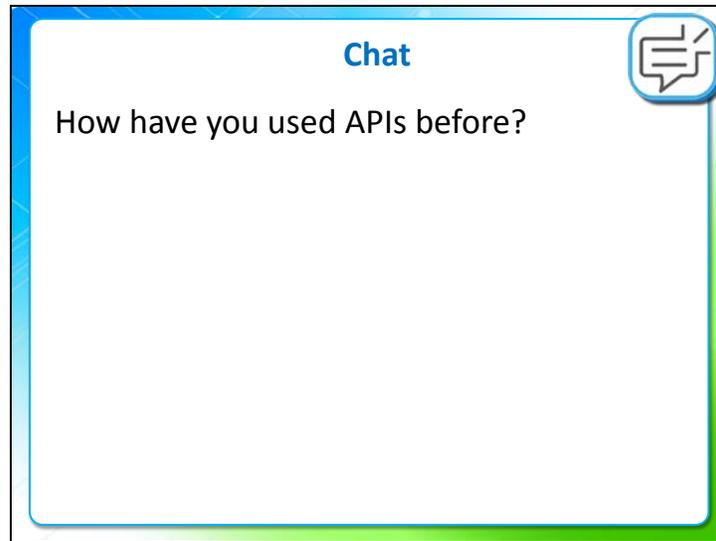
Since REST applications focus on the data transfer instead of the internal transactions, they are often simpler to implement than SOAP applications. First, REST permits different data formats like XML and JSON, or Javascript Object Notation, while SOAP only permits XML. REST is also more scalable, meaning it can handle a greater number of requests with fewer potential glitches.

PRESENTER 2 SAY:

Another way that REST is simpler is that it's what is known as more "human readable" than the SOAP coding. If you look at the image displayed on your screen, you'll see what we're talking about. The REST code is shorter and more clearly understandable than the SOAP code.

Many big companies have switched to REST for these reasons. For example, Google used SOAP until 2006, when they switched to REST, completely eliminating the SOAP APIs. Other big names like Yahoo, Twitter, and Facebook quickly followed. This has contributed to REST emerging as a dominant web service platform.

Slide 12



PRESENTER 1 SAY:

Now that we've talked about key concepts in web architecture, we'd like to hear from you. So tell us, how have you used APIs before? Type your responses in the Lync Chat window on the left.

PRESENTER 2 SAY:

I know there are several uses for APIs. Human resource management software like PeopleSoft use APIs to streamline new hire processes. When new employees are entered into PeopleSoft's systems they are signed up for an email account, ID badge, and registered with remote conferencing services. A welcome email is also generated with their ID and a temporary password for scheduling and joining conferences is sent. Using an API in this way streamlines the hiring process and makes it more efficient for new employees to hit the ground running.

PRESENTER 1 SAY:

That's a great example! Let's move on to discuss REST basics.

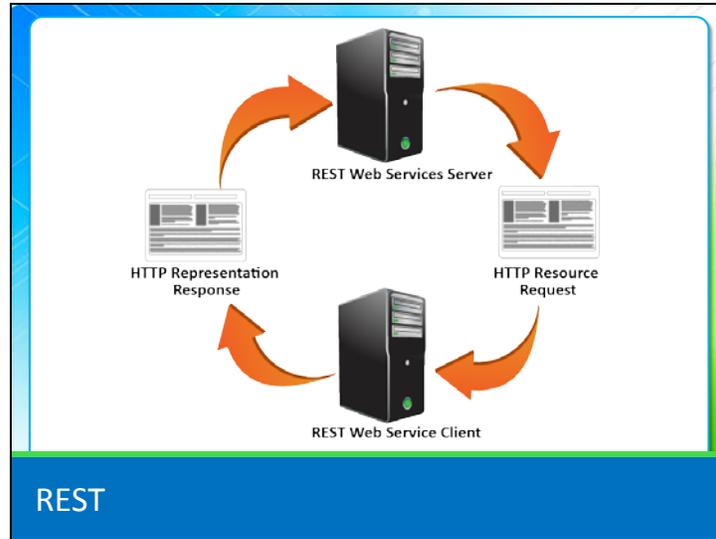
Slide 13



PRESENTER 2 SAY:

Now that we've gone over some key concepts of web architecture, let's talk about the basics of REST, including data elements and connectors.

Slide 14

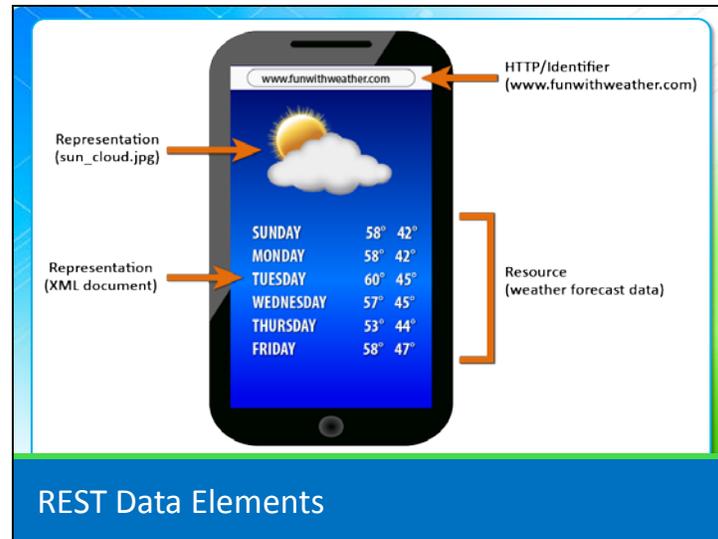


PRESENTER 2 SAY:

REST is an acronym for Representational State Transfer. It was introduced by computer scientist Roy Fielding in 2000, in his doctoral dissertation *Architectural Styles and the Design of Network-based Software Architectures*.

REST is NOT a software program, but an architectural style with principles that define how web standards, like identifiers and HTTP, can be used.

Slide 15



PRESENTER 2 SAY:

The data elements of REST include resources, resource identifiers, representations of resources, and HTTP. Let's break these down one at a time.

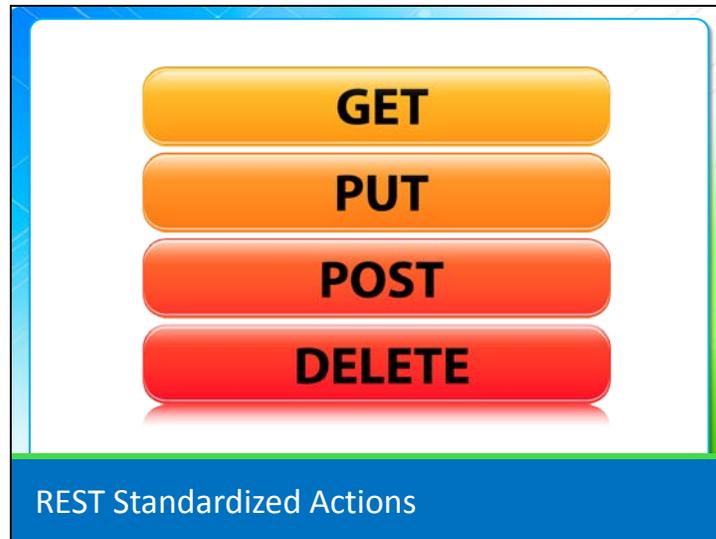
At the core of REST is the concept of a **resource**. Any information that can be named is a resource. For example, a document, an image, and today's weather forecast are all resources.

Resources have **identifiers** that are a string of characters used to identify them. These are called uniform resource identifiers, or URIs. URIs can be classified as a universal resource locator, URL, or as a uniform resource name, URN.

Every resource has a **representation**. The representation is intrinsic, meaning it's not "assigned" but "identified." For example, a document can be represented as an HTML document, an image can be represented as a JPEG image, and weather data can be represented in an XML document.

In order to get a representation of a resource from a server to your computer, **HTTP** is used.

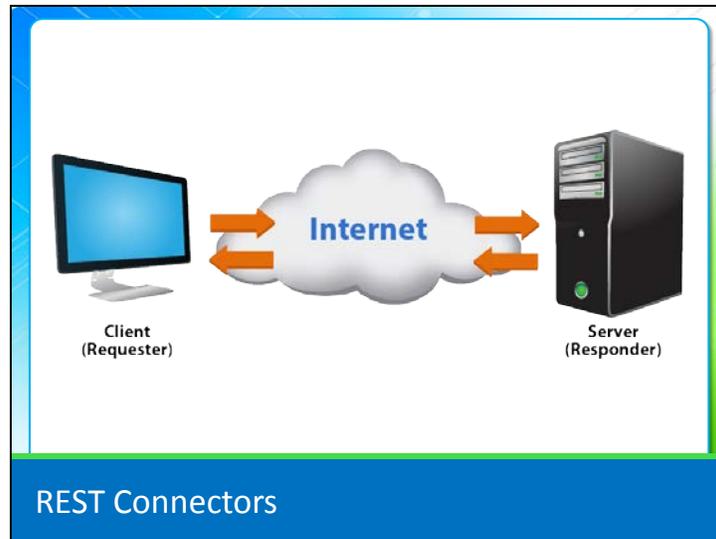
Slide 16



PRESENTER 1 SAY:

REST uses HTTP to define the action to be taken with the URL. The four standardized actions are GET, PUT, POST, and DELETE. Each of these four actions completes a different activity. A quick breakdown is as follows: GET retrieves a resource, PUT changes the state of a resource, POST creates a resource, and DELETE removes a resource. It's important to note that GET, PUT, POST, and DELETE are HTTP messages utilized by REST, but they are not REST-specific. We'll talk about the four standardized actions in much more detail during the second course. For the purpose of today's session, we just want to make sure you are aware of them and what each one of them does.

Slide 17



PRESENTER 2 SAY:

REST connectors include clients, servers, and tunnels for communication.

REST uses a client-server model. The client is the requester, and server is the responder. The client asks for something, and the server responds. For example, the client might ask, “What time is it?” and the server would respond with the correct time.

The tunnels for communication provide an interface for communication that hides the implementation details of the communication. For example, when you tune your radio dial to your favorite station, your station automatically plays, and you don’t have to worry about the inner workings of the radio. All you have to do is adjust the dial.

Slide 18



PRESENTER 1 SAY:

REST is stateless, meaning the server treats each request as an independent transaction that is unrelated to any other request. Therefore, each request and interaction stands on its own.

For example, when you send out a request for information, the browser requests the web page you are searching for from a web server. The web server responds by delivering the information you requested back to your computer and then returning to a stateless condition, where it has no memory of the request or the information that was in it, therefore, making the interaction stateless.

Each individual request contains all the information it needs to understand and complete that one single interaction. Since each request stands on its own, the results of one request will not affect the results of another request, which allows the web service to operate more efficiently.

Slide 19

Poll

Which of the following options are examples of **data elements** in REST?

- a. XML and JSON
- b. Resource, resource identifiers, representations of resources, and HTTP
- c. Clients, servers, and tunnels for communication
- d. GET, PUT, POST, and DELETE

PRESENTER 2 SAY:

Now that we've talked about REST basics, we'd like to see what you've learned about data elements. So based on our discussion, which of the following options are examples of data elements in REST?

Your options are:

- a. XML and JSON
- b. Resource, resource, identifiers, representations of resources, and HTTP
- c. Clients, servers, and tunnels for communication
- d. GET, PUT, POST, and DELETE

PRESENTER 1 SAY:

Take your time and think about the examples of data elements we just discussed.

If you guessed b. Resource, resource identifiers, representations of resources, and HTTP, you're correct!

Let's move on to discuss programming RESTful web services.

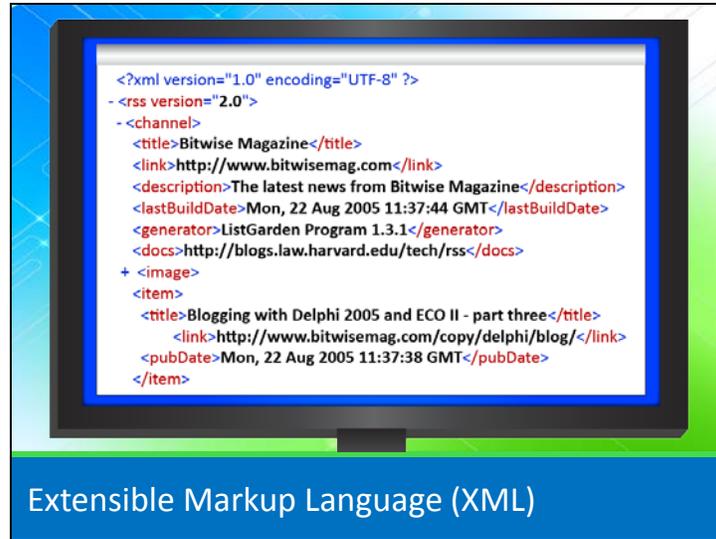
Slide 20



PRESENTER 2 SAY:

Now that we've discussed the basics of REST, let's take a look at how to program RESTful web services. Any information that uses the REST architectural style is considered a RESTful web service. First, we're going to look at two data formats: Extensible Markup Language and JavaScript Object Notation.

Slide 21

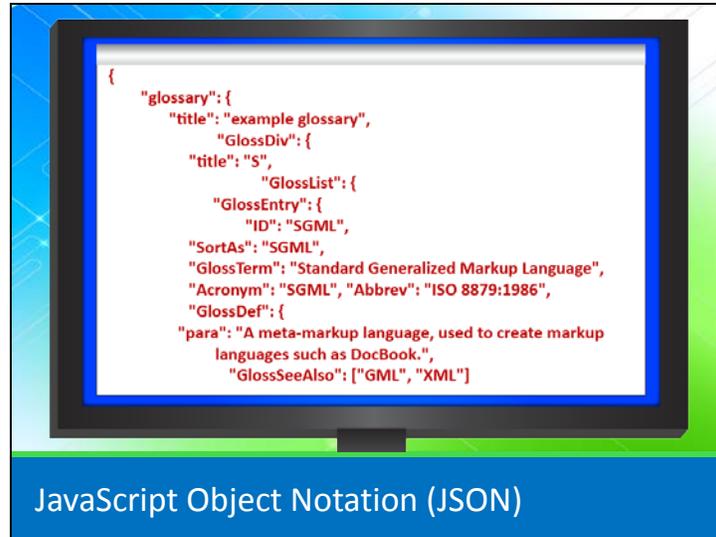


PRESENTER 2 SAY:

When you use your standardized actions, the data returned by RESTful web services can be in any format; the most common are XML or JSON. Therefore, XML and JSON are alternatives to one another. Although you can use either, the decision on which to use usually comes from another entity, maybe the system you have to interface with or a hardware requirement. For example, if you are trying to communicate with an existing system that uses XML, you would create a system that uses XML.

XML is a markup language similar to Hypertext Markup Language, or HTML. XML differs from HTML because it was created to carry or transfer data from one server to another, while HTML was designed to display data on from a website. Additionally, XML tags aren't predefined like HTML tags. Tags are used to mark the beginning and ending of a specific piece of code. These XML tags make your documents more versatile, allowing for systems to handle large and diverse with multiple types of interrelationships.

Slide 22



PRESENTER 2 SAY:

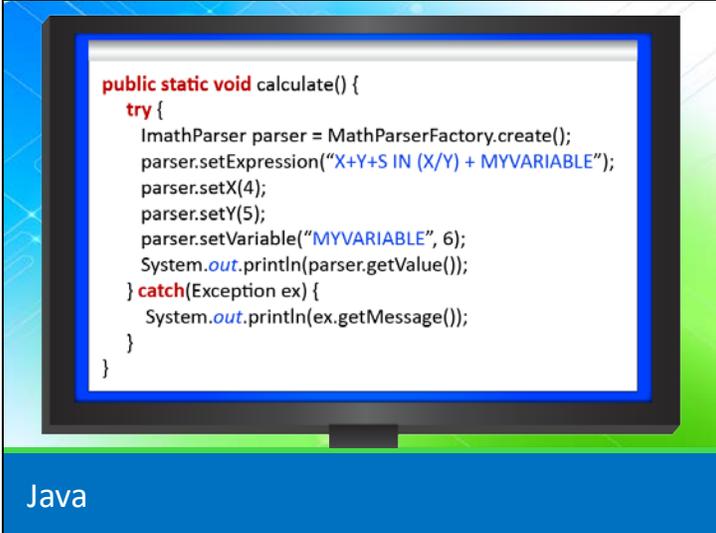
JavaScript Object Notation, or JSON, is a similar markup language to XML. For many years, XML was the most popular data transfer format; however, in recent years there's been a slight shift towards JSON. Although XML is still used most often, JSON is quickly catching up. For example, social media sites, like Twitter, have switched from XML to JSON only formatting.

JSON tends to be faster and easier for computers to parse or analyze. JSON requires a less complicated parsing process when converting it from text back to data in the client or server. And the simpler format of JSON is also smaller in size, which may lead to better network utilization.

JSON is typically used for systems that handle a single type of data or for communicating smaller sets of data back-and-forth.

Now that we've talked about markup languages, let's take a look at Java.

Slide 23



```
public static void calculate() {  
    try {  
        ImathParser parser = MathParserFactory.create();  
        parser.setExpression("X+Y+S IN (X/Y) + MYVARIABLE");  
        parser.setX(4);  
        parser.setY(5);  
        parser.setVariable("MYVARIABLE", 6);  
        System.out.println(parser.getValue());  
    } catch (Exception ex) {  
        System.out.println(ex.getMessage());  
    }  
}
```

Java

PRESENTER 1 SAY:

Java is the programming language that is commonly used when creating RESTful web services. It's currently one of the most commonly used programming languages in the world, specifically for client-server web applications, for which we will use Java.

Java is intended to "write once, run anywhere," meaning you write the code once and it can deploy on any operating system, which makes it a great platform for web development.

During the second course, we're also going to learn about the Java Jersey library, which you'll be using to complete your homework assignment for that course. The Jersey library is a Java library that helps programmers write RESTful web services. The majority of the time, you'll use Java libraries to help you write code, so you don't have to manually write low-level code yourself, which takes large amounts of time and can be very taxing, and potentially inefficient. These Java libraries have been written by other programmers, and can be used to help you save time when writing and implementing RESTful web services yourself. You can download the Jersey library and other Java libraries off the Internet, which we'll learn more about during the next course. Jersey is approved in the TRM, or Technical Reference Model, which means you can use it on VA systems, provided it remains patched. It is important to make sure any technology used complies with the TRM or has the appropriate waivers approved.

Slide 24



PRESENTER 2 SAY:

Shortly, we're going to show you how to connect to a REST API on data.gov and a commercial API, but before we do that we want to share with you some examples of REST in the real world.

Examples of REST APIs are all over the web. Some examples you may be familiar with include Amazon Web Services, Google Maps, and NOAA Weather data.

REST APIs are also popular for social media sites like Twitter, Facebook, and Flickr.

REST APIs are used by federal agencies. Websites like data.gov/developers/apis have an extensive list of APIs from across the government. We suggest that you take some time to look through the list when you have some free time. We'll provide that link in the guidebook handout.

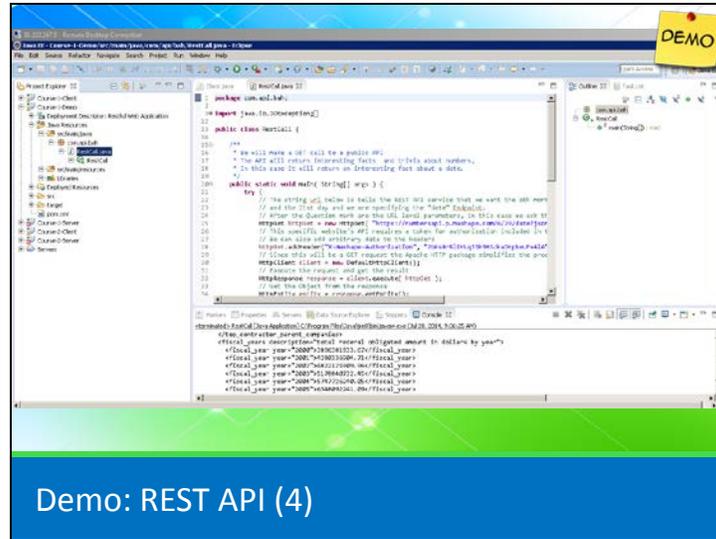
Slide 25



PRESENTER 2 SAY:

Now, we're going to show you how to connect to a live REST API. One is a commercial REST API and the other is a government API we found on data.gov. One API provides interesting info based on numbers, specifically a date that we input. The other API is the government spending API. We're going to find the sub-award summary for the state of Illinois.

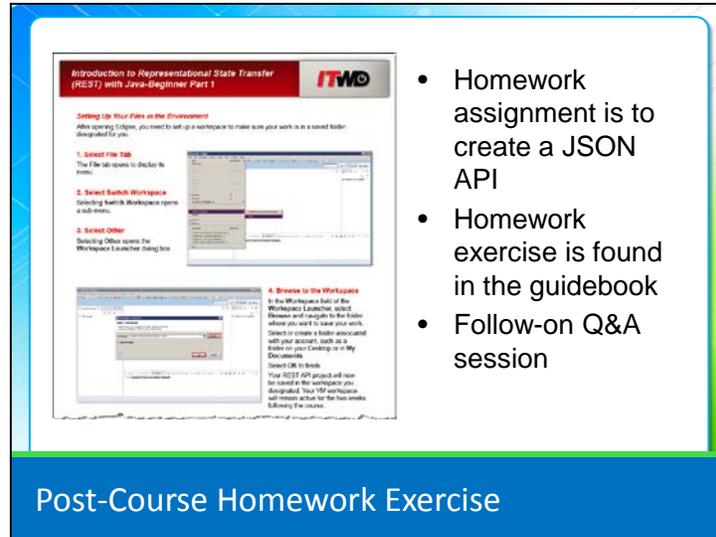
Slide 28



PRESENTER 2 SAY:

As part of our demo, we've also made a call to usaspending.gov to find a data summary about government spending from 2000 to present. The data summary is in XML format. So when we make that call, we get back the XML data from usaspending.gov. You can see the fiscal year spending here, lower down in the response. Note that for usaspending.gov, we don't need an API key.

Slide 29



- Homework assignment is to create a JSON API
- Homework exercise is found in the guidebook
- Follow-on Q&A session

PRESENTER 2 SAY:

Now that we've shown you a REST API, as we mentioned at the beginning of the class today, we have a homework exercise for you to try. For the exercise, you're going to practice creating an API. You'll do this on your own after this course is over. It's a self-paced exercise set up for you in a virtual environment, and we've provided you with a guidebook that will step you through everything. In the guidebook, you'll find sets of step-by-step directions for you to access VMware, where we have set up this homework assignment, and complete the exercise. We've provided all of the software tools in VMware that you need to complete the assignment.

For the assignment, you'll create an API using JSON. For those of you who haven't done this before, it may sound a little scary, but the guidebook will walk you through it. We would like for you to complete this assignment before the *REST Beginner Part 2* session because we will build upon what you have learned in this course.

If you have any issues or questions about the assignment, we'll have a separate Q&A session that you can dial into and ask questions.

Slide 30



PRESENTER 1 SAY:

So wrapping up our session today, we covered some key concepts in web architecture, including the OSI model, HTTP, XML, JSON and how they all relate to REST. We discussed the basics of REST, including data elements like resources and resource representations, the stateless nature of REST, how REST utilizes HTTP standard actions, and how it can simplify communications between computers. We also talked about Java and how many REST applications use it, and we showed you a demo of how to connect to a REST API. At this point, we have introduced the basics to you and will build upon this foundation in the upcoming courses, including providing more in depth information on developing REST applications in Java using the Jersey library.

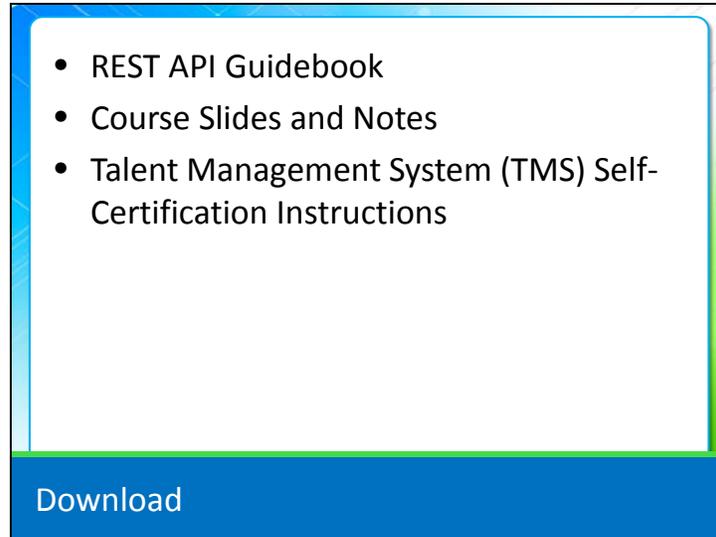
Slide 31



PRESENTER 2 SAY:

Now we want to hear from you. What questions do you have? First, we'll review the questions we received in the Lync Chat window. Please type in any more questions you have.

Slide 32



- REST API Guidebook
- Course Slides and Notes
- Talent Management System (TMS) Self-Certification Instructions

Download

PRESENTER 1 SAY:

We've got three handouts for you to download today. We've got a guidebook for your self-paced homework assignment, the full course transcript, and the directions for self-certifying in the Talent Management System, or TMS, that you took this course.

To access the downloads, select the paperclip icon in the Lync toolbar above the list of participants. Next, select the right arrow next to the file in the Attachments pop-up menu. From there you can select Open, or select Save As to navigate to a location where you can save the download.

Slide 33



PRESENTER 1 SAY:

Thank you for joining us in today's session. As we mentioned earlier, this is the first in a series of courses on REST. In the second course, we'll discuss some of these topics in more depth, as well as fundamental data types and REST design guidelines.

Be sure to check the training calendar on the ITWD portal to find out when these courses will be offered, so you don't miss them.

Slide 34

TMS Self-Certification

- Log in to the TMS <http://www.tms.va.gov>
- Enter **3878052** in the Search Catalog field on your TMS home page and select the **Go** button
- Select the **Start Course** button
- Select the **Yes** button
- Select the **OK** button
- Select the **Close Window** button
- **NEW!** Complete the course evaluation survey that is on your TMS To-Do List

For assistance, contact the **TMS Help Desk:**
vatmshelp@va.gov or **1-866-496-0463**

PRESENTER 2 SAY:

Take a minute to self-certify to get credit for your participation in this training. TMS self-certification instructions are provided on the slide.

Once you're self-certified, you will receive a link to complete an evaluation for this training

We will leave the instructions up for a couple minutes to allow you time to self-certify.

This concludes *Introduction to Representational State Transfer (REST) with Java, Beginner Part 1*. We'll be adding more training so keep your eyes open for those announcements. Thank you for your participation.