1. Contract-first web service = a web service that starts with the XML Schema/WSDL contract first followed by the Java code second.
2. In contract-first web services, it’s XML that’s sent across the wire, and the fact that Java is used to implement the Web service is an implementation detail.
3. Once we have some sense of the XML data we will use, it makes sense to formalize this into a schema. This data contract defines the message format we accept. There are four different ways of defining such a contract for XML, one of which is XML Schema (XSD)
4. A service contract is generally expressed as a WSDL file. In spring-WS, writing the WSDL by hand is not required. Based on the XSD and some conventions, Sping-WS can create the WSDL for you. You can still write it by hand, though.

Youtube: Spring-WS Tutorial

* Spring-ws is designed to support a technique called contract-first development.
* Spring-WS is a separate Spring project to give us the tools to write endpoints and expose hand-written WSDLs. It exists specifically to allow you to write contract-first web services
* An xsd is a document that describes the valid content of an xml document
* In addition to defining the actual types of data, we’ve also got to create the types for the inputs and outputs for our service in the WSDL
* Think of namespaces as a unique label that we associate with our xsd that keeps us from having conflicts with other definitions that might have the same names as we used in our xsd. Very similar to java packages & the naming scheme we use for those
* The next step is writing an endpoint. The endpoint receives the xml document that the client sends to our service. Because we’re using sping WS, you receive a representation of the raw XML at the Endpoint. Really it’s still a Java Object, and we can choose what the representation is going to be. So when we’re writing our endpoints, we have a choice: use a low-level XML library and have fine-grain control, or do it higher-level or even let SpringWS generate a java object for us without ever seeing the XML
* Use the @Endpoint annotation to annotate your endpoint class
* Need to autowire your service
* The way we associate the methods in our endpoint is via the PayloadRoot annotation. Effectively says that the incoming data to this method will be an xml document, and with this annotation we’re asking spirngWS to convert the XML to a java object that we can handle